Flexibilisierung verteilter Prozessausführung

Zur dynamischen Verteilung, Überwachung und Steuerung individueller Prozessinstanzen auf Anwendungsebene

Dissertation

zur Erlangung des Doktorgrades
an der Fakultät für Mathematik, Informatik und Naturwissenschaften,
der Universität Hamburg

eingereicht beim Fachbereich Informatik von
Dipl.-Inf. Sonja Zaplata
Freie und Hansestadt Hamburg

Erstgutachter: Prof. Dr. Winfried Lamersdorf, Universität Hamburg
Zweitgutachter: Prof. Dr. Frank Leymann, Universität Stuttgart

Tag der Disputation: 24. Oktober 2012

Zusammenfassung

Die fachliche Modellierung, technische Umsetzung und computergestützte Ausführung von Prozessen ermöglichen in der Softwareentwicklung eine flexible Abbildung und Unterstützung von Anwendungsvorgängen mittels Informations- und Kommunikationstechnologie. Insbesondere bei *verteilt ausgeführten Prozessen* ist dabei aufgrund des komplexen Zusammenspiels zwischen heterogenen autonomen Systemen, der Veränderlichkeit der Systemumgebung und einer zunehmenden Mobilität von Systemkomponenten häufig eine Anpassung der Ausführung an veränderte Kontexte erforderlich. Eine wichtige Art dieser Anpassung besteht in der Partitionierung von Prozessen und der Zuweisung von Prozessabschnitten zu prozessausführenden Systemen. Für *individuelle Prozessinstanzen* ist jedoch bislang eine dynamische Anpassbarkeit dieser Parameter oft nicht ohne unverhältnismäßig hohen Aufwand möglich, da die hierfür notwendige Flexibilität aufgrund starrer Verteilungsstrukturen und unzureichender Transparenz der verteilten Prozessausführung in der Regel nicht gegeben ist.

Diese Arbeit leistet einen Beitrag zur Flexibilisierung verteilt ausgeführter Prozesse, indem konzeptionelle Grundlagen und technische Mechanismen untersucht bzw. erarbeitet werden, die eine in Hinblick auf die Verteilung *fortwährend anpassungsfähige* Ausführung von Prozessinstanzen durch mehrere über Netzwerke permanent oder zeitweilig verbundene Ausführungseinheiten erlauben. Als Weiterentwicklung der verteilten Prozessausführung auf Basis dienstorientierter Architekturen wird dabei das Prinzip des *Process-Management-as-a-Service* nutzbar gemacht, um die Entscheidung über die Verteilung eines (fachlich determinierten) Prozesses, dessen verteilte Ausführung, dessen Überwachung und dessen dynamische Anpassung auf der Basis benutzerdefinierter Rahmenbedingungen zu ermöglichen. Dabei werden als Hauptbeiträge dieser Arbeit ein Verfahren zur nicht-invasiven Verteilung eines Prozesses durch Migration von Prozessinstanzen und ein Rahmenwerk zur Überwachung verteilter Prozesse auf der Basis von Prozessmanagementsystemen als verwaltbare Ressourcen vorgestellt. Da eine dynamische Verteilung die Betrachtung des Ausführungskontextes erfordert, werden außerdem ein erweitertes Konzept zur dynamischen Darstellung von Benutzerinteraktionen sowie ein Prognoseverfahren zur proaktiven Anpassung der Prozessausführung an antizipierte Kontexte vorgeschlagen. Als praktischer Beitrag dieser Arbeit werden die hiermit verbundenen Lösungsvorschläge so in eine komponentenbasierte Middleware integriert, dass die verteilte Ausführung von Prozessinstanzen über mehrere Prozessmanagementsysteme hinweg zur Laufzeit möglich wird.

Die Ergebnisse dieser Arbeit zeigen, dass eine fortwährende Anpassungsfähigkeit der verteilten Prozessausführung erreicht werden kann, wenn das fachliche Prozessmodell hinreichend von technischen Anweisungen zur Verteilung entkoppelt ist und durch eine geeignete Abstraktion die Verteilungstransparenz als inhärente Eigenschaft verteilter Systeme auch für prozessorientierte Anwendungen adäquat umgesetzt wird. Als Konsequenz können prozessorientierte Anwendungen weitestgehend unverändert mit bekannten Softwarewerkzeugen entwickelt und ausgeführt werden und trotzdem einzelne Prozessinstanzen bei Bedarf dynamisch in der Art ihrer Ausführung an veränderte Systemumgebungen angepasst werden. Die erarbeiteten Konzepte wurden im Kontext von aktuellen Standardprozessbeschreibungssprachen wie XPDL, WS-BPEL und BPMN sowie entsprechenden Prozessmanagementsystemen für den stationären und mobilen Einsatz umgesetzt. Die prototypischen Implementierungen hierfür bestätigen, dass mit der neu erreichbaren Flexibilität eine dynamische Verteilung von individuellen Prozessinstanzen möglich ist, ohne dass die Repräsentation der Prozesse hierfür syntaktisch, inhaltlich oder strukturell angepasst werden muss.

Abstract

Business process modelling, technical implementation, and computerized execution of process-oriented applications constitute important techniques for flexibly supporting structured business cases with information and communication technology. Facing a complex interplay of heterogenous and autonomous systems, highly dynamic system environments, and increasing mobility of devices, especially *distributed processes* are often subject to adaptation. In respect of distributed process execution, an important type of adaptation is given by dynamic partitioning of processes and assignment of resulting process parts to selected process execution systems. However, for *individual process instances*, a continuous adaptability of those parameters can often not be achieved as the necessary flexibility of process execution does not exist due to rigid distribution structures and due to an insufficient transparency of execution.

This thesis contributes to increasing flexibility of the execution of distributed processes by examining and developing conceptual foundations and technical mechanisms in order to allow for an adaptable distribution of processes executed by a set of permanently or temporarily connected systems. Based on distribution by service-oriented architectures, the principle of *Process-Management-as-a-Service* is leveraged in order to allow for the dynamic decision on the distribution of a (functional and structural determined) process, its distributed execution, its adaptable monitoring, and its dynamic adaptation based on user-defined requirements. As major contributions, this thesis presents approaches for a non-invasive distribution of processes by means of process instance migration and for monitoring and controlling distributed processes based on process management systems as manageable resources. Since dynamic distribution also requires reconsideration of the execution context, enhanced concepts for the dynamic representation of user interfaces and for the prediction of application-specific context are proposed additionally. As practical contribution, the proposed strategies are integrated into a component-based middleware allowing for an ad-hoc distributed execution of (running) process instances across multiple process management systems.

The findings of this thesis indicate that a continuous adaptability of distributed process execution can be achieved if the functional business process is sufficiently decoupled from technical instructions for distribution and management and, in consequence, distribution transparency as an inherent characteristic of distributed systems is also realized for distributed process-oriented applications. As a result, processes can still be developed with existing process modelling tools and can be run on existing process execution systems, but individual process instances can also be adapted to dynamically changing system environments by customizing their way of execution on demand at runtime. The presented concepts have been applied in the context of standard process execution languages such as XPDL, WS-BPEL and BPMN and respective stationary and mobile process execution systems. The prototypical implementations demonstrate that increased flexibility allows for a dynamic distribution of individual process instances without the need for syntactical, functional, or structural modification of the processes' representation.

Danksagung

Besonderer Dank gilt meinem Doktorvater Herrn Prof. Dr. Winfried Lamersdorf für die Betreuung meines Dissertationsprojektes und die Gelegenheit, im Umfeld des Arbeitsbereichs *Verteilte Systeme und Informationssysteme* (VSIS) und des europäischen Exzellenznetzwerks *Software Services and Systems Network* (S-Cube) mit großer Freiheit wissenschaftlich tätig sein zu dürfen.

Einen großen Anteil an dieser Arbeit hat zudem Herr Dr. Christian P. Kunze, der mir als Betreuer meiner Diplomarbeit das „Tor zur Welt der Wissenschaft“ geöffnet und darüber hinaus durch viele gemeinsame Projekte zum inhaltlichen Kontext dieser Arbeit beigetragen hat.

Desweiteren möchte ich mich herzlich bei meinen Kollegen und Weggefährten vom Arbeitsbereich VSIS bedanken, insbesondere bei Herrn Dirk Bade, Herrn Kristof Hamann und Herrn Ante Vilenica, die mir nicht nur stets durch fruchtbare inhaltliche Diskussionen und gemeinsame Forschungsarbeiten, sondern auch durch zwischenmenschlichen Rückhalt und durch ihre große Hilfe beim Korrekturlesen dieser Arbeit zur Seite standen. Ebenso möchte ich mich auch bei meinem guten Freund Herrn Frank Fürstenwerth für seine außergewöhnliche Hilfsbereitschaft und Unterstützung beim Korrekturlesen dieser Arbeit bedanken.

Großer Dank für weitere inhaltliche Beiträge gilt den vielen Studierenden, welche in Lehrveranstaltungen und im Rahmen ihrer Bachelor- und Diplomarbeiten maßgeblich zu meinem Dissertationsprojekt beigetragen haben, insbesondere Herrn Viktor Dreiling, Herrn Matthias Meiners und Herrn Kristian Kottke, welche darüber hinaus auch als studentische Mitarbeiter an weiterführenden Inhalten und gemeinsamen Publikationen mitgewirkt haben.

Sonja Zaplata

Inhaltsverzeichnis

1 Einleitung

Die Entwicklung prozessorientierter Softwaresysteme stellt einen wichtigen Teilforschungsbereich der verteilten Softwaresystemtechnik zur Abbildung von automatisierbaren systemübergreifenden Vorgängen der Realwelt und zu deren Unterstützung durch Informations- und Kommunikationstechnologien dar. Insbesondere die zunehmenden Anforderungen an diese Systeme in Hinblick auf ihre Anpassbarkeit an dynamische Veränderungen der Systemumgebung motivieren die Betrachtung einer möglichen Flexibilisierung solcher Systeme als Untersuchungsgegenstand dieser Arbeit. Dieses einführende Kapitel zeigt diese Anforderungen auf und ordnet sie in den Kontext der Entwicklung verteilter Systemsoftware und in den Rahmen des aktuellen Forschungsstands auf diesem Gebiet ein. Im Zuge dessen wird die Problemstellung dieser Arbeit konkretisiert und die gewählte Herangehensweise zur Erarbeitung von Lösungsvorschlägen vorgestellt. Das Kapitel schließt mit einer Zusammenfassung der erzielten Ergebnisse und der eigenen Beiträge zum Stand der Forschung auf diesem Gebiet.

1.1 Motivation

Das Internet und die Mobilkommunikation haben das wirtschaftliche und gesellschaftliche Umfeld in den letzten Jahren stark verändert. Die steigende Leistungsfähigkeit und stetig sinkende Kosten für Kommunikation und Infrastruktur tragen dazu bei, dass Informationen und Dienste nahezu jederzeit an beliebigen Orten verbreitet und abgerufen werden können. Die daraus resultierenden Möglichkeiten zur lokalen oder globalen Vernetzung erlauben es dabei einer Vielzahl von Organisationen, Personen und Computersystemen, spontan miteinander in Verbindung zu treten. Dies fördert sowohl *Kooperationen* über Unternehmens- oder Landesgrenzen hinweg (zum Beispiel im Rahmen *virtueller Unternehmen* [Bor06, Rie08] oder des *E-Government* [SKH03, May04a]), als auch die Integration von lokalen Ressourcen im persönlichen Umfeld (wie zum Beispiel im Bereich des *Mobile Computing* [Kun08] oder der *Home Automation* [CCC07]).

Der Einsatz von modernen Informations- und Kommunikationstechnologien zur Realisierung *verteilter Systeme* unterstützt dabei sowohl die Selbständigkeit einzelner, spezialisierter Komponenten, als auch die Integration und den Austausch von Funktionalitäten nach marktlichen oder nutzerspezifischen Kriterien. Eine wichtige Methodik zur Realisierung von systemübergreifender Vorgängen umfasst dabei eine möglichst technologieunabhängige Formulierung der auszuführenden fachlichen Anwendungslogik, deren Zerlegung in einzelne Schritte und die Verteilung der resultierenden

Teilaufgaben an geeignete Ausführungseinheiten. Im Unternehmenskontext wird diese Art der Umsetzung als Teil des *Geschäftsprozessmanagements* (*Business Process Management*) wahrgenommen, indem relevante Wertschöpfungsketten identifiziert, auf Geschäftspartner wie Hersteller, Händler, Lieferanten, Kunden oder Transportdienstleister abgebildet und durch *Prozessmanagementsysteme* kontrolliert ausgeführt werden. Aber auch außerhalb dieses betrieblichen Kontextes entsteht durch die maschinen- bzw. computergestützte Ausführung von beliebigen Vorgängen eine zunehmende Automatisierung, so dass auch Aufgaben im medizinischen, wissenschaftlichen oder privaten Nutzungsbereich zunehmend schneller und weitestgehend ohne manuelle Intervention erledigt werden können. Die Entwicklung *prozessorientierter Softwareanwendungen* stellt daher im Allgemeinen eine der wesentlichen Herangehensweisen zur Abbildung, Ausführung und Unterstützung natürlich verteilter Abläufe und kooperativer Ressourcennutzung durch Informations- und Kommunikationstechnologie dar, bei welcher weitreichende Vorteile wie die Reduktion von Kosten, Leistungssteigerungen, eine höhere Fehlertoleranz und eine bessere Skalierbarkeit erzielt werden können [Ses10].

Aufgrund der zunehmenden Dynamik des Umfeldes, dem steigenden Bedarf von Nutzern nach Individualisierung sowie der wachsenden Mobilität von Komponenten kann jedoch oft bei der Initialisierung solcher (insbesondere lang andauernder) prozessorientierter Anwendungsfälle eine spätere Entwicklung noch nicht abgesehen werden. In vielen Fällen ist eine schnelle Reaktion *während der Ausführung* erforderlich, um die Wettbewerbsfähigkeit von Organisationen zu erhalten oder zu erhöhen, das aktuelle, sich schnell weiterentwickelnde Potential neuer Technologien zu nutzen oder auf unerwartete Situationen angemessen zu reagieren [KKJ07]. Dabei kommt es insbesondere bei *verteilt ausgeführten Prozessen* aufgrund des komplexen Zusammenspiels zwischen heterogenen autonomen Systemen und der potentiellen Veränderlichkeit der Systemumgebung zu einem hohen Bedarf an *Anpassungsfähigkeit*.

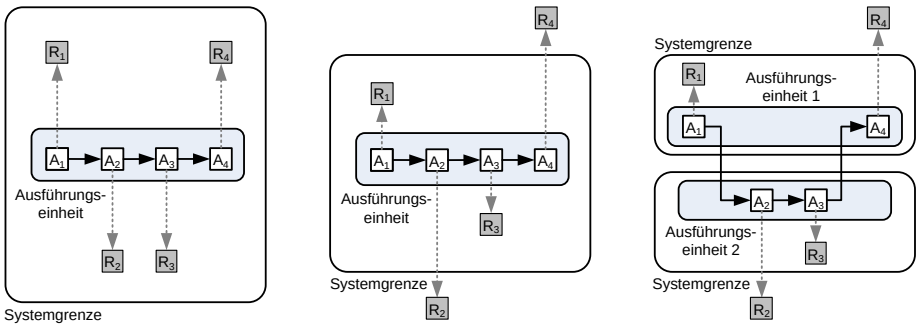
Voraussetzung für die Entwicklung und Ausführung jeglicher anpassungsfähiger Systeme ist ein ausreichendes Maß an *Flexibilität*. Die Überwindung vorherrschender starrer Strukturen für die flexible Ausführung prozessorientierter Anwendungen stellt dabei eine wesentliche Herausforderung dar. Im Fall von verteilten Systemen erschwert zusätzlich das Überschreiten von Systemgrenzen, die eingeschränkte Transparenz und das Verlassen eines einheitlichen Einflussbereichs ein agiles Handeln und damit die Durchführung notwendiger Anpassungen. Insbesondere lang andauernde organisationsübergreifende Geschäftsprozesse sind zur Zeit von diesem Problem betroffen, denn die meisten aktuell praktisch eingesetzten Produkte lassen nach der Initiierung der automatisierten Bearbeitung eines Anwendungsfalls keinerlei Änderungen mehr daran zu [Wes07]. Die Steigerung der Flexibilität ist daher derzeit eines der wichtigsten Ziele bei der Einführung neuer oder der Ergänzung bestehender Softwaresysteme [Mar10].

Im Gegensatz zur (fachlichen und technischen) Anpassung traditioneller, nicht-verteilter Prozessabläufe sind Möglichkeiten zur *Flexibilisierung verteilt ausgeführter Prozesse* in der praktischen Informatik bislang nicht hinreichend untersucht worden. Aufgrund der oben genannten aktuellen und fortschreitenden Entwicklung in Bezug auf Vernetzung, Wettbewerb und Kooperation kommt jedoch gerade solchen verteilten prozessorientierten Anwendungen und deren flexibler Ausführung eine große Relevanz zu. Diese Arbeit befasst sich daher mit der Untersuchung solcher *dynamisch verteilt ausgeführter* prozessorientierter Anwendungsfälle, identifiziert relevante Flexibilisierungsmöglichkeiten und zeigt Konzepte und technische Umsetzungsmöglichkeiten für eine Anpassung verteilt ausgeführter Prozesse an dynamische Kontexte auf.

1.2 Gegenstand der Forschung

Im Mittelpunkt dieser Arbeit steht die verteilte Ausführung von strukturierten Aufgaben und Vorgängen auf Anwendungsebene, welche sich in einer ausführbaren Repräsentation beschreiben, in mehrere Teilschritte zerlegen und so in einer bestimmten zeitlichen und kausalen Abfolge durch ein oder mehrere Softwaresysteme abarbeiten lassen. Beispiele hierfür sind Geschäftsprozesse, aber auch beschreibbare Abfolgen von automatisierten oder computergestützten manuellen Aktivitäten, die anderen als kommerziellen Zwecken dienen. Als Verallgemeinerung wird im Folgenden eine solche Abfolge von Aktivitäten als *Prozess* bezeichnet (vgl. Abschnitt 2.1).

Im Gegensatz zu *zentral ausgeführten* Prozessen, welche auf lokale oder verteilte Ressourcen wie Softwareanwendungen oder Personen zugreifen, dabei jedoch durch eine zentrale Instanz verwaltet werden, zeichnen sich *verteilt ausgeführte Prozesse* dadurch aus, dass während der Ausführung des Prozesses verschiedene *Ausführungseinheiten* für die Bearbeitung des Kontrollflusses und für den Aufruf von Ressourcen verantwortlich sind. Abbildung 1.1 verdeutlicht diese Unterscheidung anhand eines einfachen Prozesses mit den Aktivitäten A_1, A_2, A_3, A_4, durch deren Ausführung ein sequentieller Zugriff auf die Ressourcen R_1, R_2, R_3, R_4 vorgenommen wird. Während die Ausführung des Prozesses in Abbildung 1.1a die Grenzen eines lokalen Ausführungssystems nicht überschreitet (z. B. für die Bearbeitung eines ausschließlich organisationsinternen Anwendungsfalls), werden bei der Ausführung des Prozesses in Abbildung 1.1b auch entfernte Ressourcen genutzt (z. B. zur Integration externer Anwendungsfunktionalitäten). In beiden Fällen liegt die Kontrolle über die Prozessausführung jedoch bei einer einzigen Ausführungseinheit, d. h. bei einer einzigen *Prozess-Engine*. Abbildung 1.1c zeigt eine mögliche *verteilte Ausführung* dieses Prozesses, z. B. zur Realisierung einer Kooperation zwischen verschiedenen Organisationen, welche ihrerseits verschiedene lokale und entfernte Ressourcen zugreifen können. Entsprechend aktueller Anforderungen können die beteiligten Ausführungseinheiten dabei sowohl *zentral* als auch *dezentral* verteilt organisiert sein, *stationäre* oder *mobile* Teilnehmer



(a) Zentral ausgeführter Prozess mit Aufruf lokaler Ressourcen

(b) Zentral ausgeführter Prozess mit Aufruf lokaler und entfernter Ressourcen

(c) Verteilt ausgeführter Prozess mit Aufruf lokaler und entfernter Ressourcen

Abbildung 1.1: Varianten der Prozessausführung (Beispiele)

umfassen sowie als Einheit *autonom* sein oder einer größeren Organisationseinheit, wie etwa einem Unternehmen oder einem Rechenzentrum angehören. Die Variante solcher *verteilt ausgeführter Prozesse* (vgl. Abbildung 1.1c) bildet den Forschungsgegenstand dieser Arbeit.

Wie in Abschnitt 1.1 motiviert wurde, besteht für eine bedarfsgerechte Ausführung von prozessorientierten Anwendungen ein steigender Bedarf, Entscheidungen über die konkrete Ausführung erst zur Laufzeit des Prozesses treffen oder während der Laufzeit des Prozesses anpassen zu können. Eine solche Anpassung (*Adaption*) kann grundsätzlich auf Initiative des Systems selbst geschehen (*Adaptivität*) oder auf Initiative des Benutzers erfolgen, was in beiden Fällen die Eigenschaft der *Anpassungsfähigkeit* (*Adaptierbarkeit*) voraussetzt [Kla04]. Eine solche Anpassungsfähigkeit kann nur dann realisiert werden, wenn bei der Entwicklung von Softwaresystemen starre Strukturen vermieden werden und die für den gewünschten Anpassungsspielraum notwendige *Flexibilität* geschaffen wird bzw. bis zum Zeitpunkt der Anpassung oder sogar darüber hinaus erhalten bleibt [DKN+06, VMSS07]. In Bezug auf die oben genannte verteilte Ausführung von Prozessen bedeutet dies, dass die Art und Weise der verteilten Ausführung (d. h. insbesondere die Granularität und die Zielorte der Verteilung) nicht zur Entwicklungszeit des Prozesses statisch vorgegeben werden dürfen, sondern auf Basis der auszuführenden bestehenden Prozesse, der involvierten Prozessausführungssysteme und des zur Ausführung relevanten Kontextes zur Laufzeit bestimmt werden müssen.

Um eine möglichst hohe Anpassungsfähigkeit zu erreichen, kommt dabei zunächst der *Flexibilität der Verteilung* eines Prozesses eine besondere Bedeutung zu. Dabei ist es in vielen Anwendungsfällen nicht mehr ausreichend, ein *Prozessmodell* an verschiedene Ausführungseinheiten zu verteilen und alle davon abgeleiteten Prozessinstanzen in derselben Art und Weise auszuführen, sondern es besteht darüber hinaus die Notwendigkeit, auch die von einem Prozessmodell abgeleiteten *Prozessinstanzen* jeweils individuell verteilt ausführen zu können:

- Vor der Ausführung ist nicht immer sicher, ob ein Prozess überhaupt zentral oder verteilt ausgeführt werden wird. Erst aktuelle Parameter oder auftretende Ereignisse zur Laufzeit einzelner Prozessinstanzen können eine Auslagerung einzelner Schritte an andere Ausführungseinheiten vorteilhaft oder sogar erforderlich machen. Ein Beispiel hierfür sind virtuelle Unternehmen, die sich nur zur Ausführung eines einzelnen Vorgangs mit Kooperationspartnern zusammenschließen und sich danach wieder auflösen (vgl. Abschnitt 4.2.1).

- Auch bei einer im Voraus beabsichtigten Verteilung des Prozesses können teilnehmende Systeme häufig erst während der Ausführung eines Vorgangs bestimmt werden. Dies ist zum Beispiel bei der Ausführung eines organisationsübergreifenden Prozesses der Fall, wenn ein Teilnehmer zur Laufzeit zwischen verschiedenen Optionen wählen kann, welche die Integration verschiedener anderer Organisationen oder Organisationseinheiten beeinflussen (vgl. Abschnitt 4.2.2).

- Bedingt durch die wachsende Mobilität und geographische Verteilung von Ressourcen stellt in einigen Fällen auch der *Kontext der Ausführung* ein wichtiges dynamisches Auswahlkriterium dar. Ein Beispiel hierfür ist die Ausführung computergestützter Abfolgen von Befragungen im Rahmen der Marktforschung, welche nur an einem bestimmten Ort den gewünschten Zweck erfüllen. Bei der verteilten Ausführung benutzerzentrischer Prozesse auf heterogenen (mobilen) Endgeräten ist zudem das Problem einer adäquaten Interaktion mit dem Benutzer und eine angemessene Darstellung der Benutzungsschnittstelle zu adressieren (vgl. Abschnitt 4.2.3).

- Oftmals sind die für die Ausführung des Prozesses benötigten Ressourcen nur in einer lokalen Ausführungsumgebung zugreifbar. Dies gilt insbesondere für Softwareanwendungen, die nur über proprietäre Schnittstellen verfügen oder durch besondere Sicherheitsmechanismen geschützt sind, so dass ein entfernter Zugriff erschwert oder verhindert wird. Aber auch für die Ausführung relevante Personen oder Maschinen sind gegebenenfalls nur von einem bestimmten Ausführungssystem aus erreichbar (vgl. Abschnitt 4.2.4).

- Weitere sich oftmals ändernde Bedingungen stellen qualitative Eigenschaften von Ausführungseinheiten dar (*Quality-of-Service*), welche zum Beispiel auf der aktuellen Auslastung von Systemen oder Personen beruhen können. Dabei kann einerseits durch die Bevorzugung leistungsstarker oder kostengünstiger Systeme die Prozessausführung optimiert werden, und andererseits die zur Verfügung stehende Infrastruktur ressourcenschonend genutzt werden. Ein Beispiel hierfür ist die Vermeidung einer mehrfachen Übertragung großer Datenmengen durch die Verlagerung der Prozessausführung zum Ursprungsort der Daten (vgl. Abschnitt 4.2.5).

- Nicht zuletzt kann auch der Ausfall einer zuvor selektierten Ausführungseinheit einen Austausch und damit verbunden eine Anpassung der Zuweisung einzelner Prozessabschnitte zu verantwortlichen Prozessteilnehmern oder sogar eine gänzliche Umverteilung notwendig machen. Im Rahmen von sogenannten *Process-Management-as-a-Service*-Szenarien kann zudem die Ausführung von ganzen Prozessen mehr oder weniger spontan ausgelagert werden, z. B. falls eine Organisation nicht über die erforderlichen Softwaresysteme zur Prozessausführung verfügt (vgl. Abschnitt 4.2.6).

Eine wichtige Voraussetzung, um zur Laufzeit eines verteilten Prozesses flexibel reagieren und sich an Änderungen des Ausführungskontextes anpassen zu können, ist die Konzeption und Bereitstellung von *geeigneten Mechanismen zur Überwachung und Steuerung* der entfernt ausgeführten Prozesse bzw. Prozesspartitionen. Die Entwicklung von Konzepten und die Umsetzung von Technologien zur Überwachung einer flexiblen verteilten Prozessausführung und die Ausführung von adäquaten Reaktionen wird jedoch im Kontext verteilter Prozesse durch das Überschreiten von Organisations- und Systemgrenzen zur Laufzeit erschwert. Die Autonomie der Teilnehmer, eine hohe Heterogenität bei Prozessmanagementsystemen, die potentielle Diskonnektivität mobiler Systeme und fehlende Standards für systemübergreifende Managementfunktionalitäten stellen weitere große Herausforderungen dar.

Eine zufriedenstellende Unterstützung der genannten Szenarien und Anforderungen ist durch den derzeitigen Stand der Forschung bislang nicht gegeben. Eine Verteilung prozessorientierter Anwendungen kann derzeit auf Basis von *dienstorientierten Architekturen* durch die Kapselung von Ressourcen als *elektronische Dienste* erreicht werden. Die in den Abbildungen 1.1a und 1.1b dargestellten prozessorientierten Abläufe lassen sich damit z. B. im Rahmen einer *Orchestrierung* als Kompositionen solcher Dienste repräsentieren und ausführen (vgl. Abschnitt 3.4.4). Eine Verteilung von Dienstkompositionen auf mehrere Organisationen bzw. Ausführungseinheiten (vgl. Abbildung 1.1c) kann dabei durch die erneute Kapselung der durch einen bestimmten Teilnehmer zu erbringenden Aktivitäten als Dienst und die Spezifikation des zur Übergabe des Kontrollflusses notwendigen Nachrichtenaustauschs als *Choreographie* (vgl. Abschnitt 3.4.5) realisiert werden. Dies setzt jedoch in der Regel voraus, dass die Art der Verteilung, die potentiell beteiligten Parteien und die für die Kommunikation benötigten Schnittstellen bereits *vor der Initiierung* eines Prozesses feststehen. Angemessene Konzepte, welche die oben geforderte Flexibilität zur Anpassung der verteilten Prozessausführung *während der Laufzeit* ermöglichen, fehlen jedoch noch. Zudem werden durch die jetzigen Lösungsansätze die fachliche Anwendungslogik und die technische Logik zum Management der Verteilung in unerwünschter Weise vermischt und durch die Kapselung der von einer Partei zu erbringenden Leistung die dringend notwendigen Maßnahmen zur Überwachung und Steuerung der Prozessausführung

erschwert oder teilweise sogar unmöglich gemacht. Eine geeignete Abstraktion prozessorientierter Anwendungen in Hinblick auf ihre Verteilung ist daher bisher kaum durchsetzbar.

Zusammengefasst ergeben sich in Hinblick auf die angestrebte Flexibilisierung sowie aus den oben genannten Forderungen nach Kooperation, Transparenz und Dynamik besondere Rahmenbedingungen und Herausforderungen an einen geeigneten Verteilungsansatz und eine möglichst technologieunabhängige und anwendungsneutrale Softwareplattform, welche als *Middleware* die verteilte Ausführung von Prozessen, deren Überwachung und Steuerung über mehrere Prozessmanagementsysteme hinweg bedarfsorientiert unterstützt. Relevante Aufgaben sind hierbei insbesondere die

- Überwindung der Heterogenität und der Abgeschlossenheit von Prozessausführungssystemen,
- Bereitstellung von Mechanismen zum Austausch von Informationen über mehrere Prozessausführungssysteme hinweg sowie zur Steuerung eigener entfernt ausgeführter Prozesse,
- Erkennung dynamischer Veränderungen und Ermöglichung möglichst zeitnaher Reaktionen hierauf,
- Unterstützung der Entscheidung über die Verteilung und ggf. über die Partitionierung und die Zuweisung eines Prozesses zur Laufzeit,
- Integration von Mechanismen zur Selektion von geeigneten Ressourcen (d.h. von Organisationen, Personen, Hard- oder Softwaresystemen),
- Modellierung von benutzerspezifischen Restriktionen zur zielgerichteten Beeinflussung der Verteilung,
- Konzeption des Transfers von (Teil-) Prozessen zwischen Prozessmanagementsystemen sowie Bereitstellung der benötigten Kommunikations- und Koordinationsprotokolle und
- Berücksichtigung und Integration verschiedener Benutzungsschnittstellen und Endgeräte (z. B. Mobilgeräte).

Das Paradigma der *dienstorientierten Architekturen* (vgl. Abschnitt 3.4) hat sich in diesem Zusammenhang als geeignetes Konzept erwiesen, um verteilte Ressourcen flexibel in Prozesse einbinden und sogar zur Laufzeit des Prozesses austauschen zu können. Einer flexiblen Ausführung und Anpassung *verteilt ausgeführter Prozesse* bzw. der Individualisierung einzelner *Prozessinstanzen* steht es jedoch in seiner jetzigen Art der Anwendung eher im Wege. Die benötigte Flexibilität für die spontane Realisierung von Kooperationen und die Integration mobiler Systeme sowie die gewünschte Transparenz zur Ermöglichung von zeitnahen, angemessenen Reaktionen und damit das Erreichen wichtiger Wettbewerbsvorteile in einem immer dynamischeren Umfeld können somit zur Zeit nicht ausreichend zur Verfügung gestellt werden.

1.3 Zielsetzung

Diese Arbeit beschäftigt sich mit der Untersuchung dynamisch verteilt ausgeführter prozessorientierter Anwendungen, der Identifikation relevanter Flexibilisierungsmöglichkeiten und der Entwicklung geeigneter Konzepte und technischer Umsetzungsmöglichkeiten, um eine Anpassung der Ausführung verteilter Prozesse an dynamische Kontexte zu erlauben. Konkretes Ziel dieser Arbeit ist es daher, als Ergänzung bestehender dienstorientierter Architekturen zur Verteilung und flexiblen Nutzbarmachung von Ressourcen, ein Konzept zu entwickeln, um die *verteilte Ausführung von Prozessen* selbst weitestgehend zu flexibilisieren. Unter *Flexibilisierung* wird hierbei eine Erhöhung der Anpassungsfähigkeit verstanden, so dass die Aufteilung des Prozesses, die Zuweisung von Ausführungssystemen, die Überwachung der Prozessausführung sowie eine Reaktion auf potentielle Ereignisse während der Ausführung zur Laufzeit des Prozesses und unter Beachtung etwaiger benutzerdefinierter Rahmenbedingungen vorgenommen werden können. Dazu soll auf Basis dienstorientierter Architekturen eine unterstützende Systemplattform zur Erweiterung bestehender Middleware-Systeme konzipiert und entwickelt werden, welche die verteilte Ausführung von Prozessinstanzen über mehrere Prozessmanagementsysteme hinweg zur Laufzeit unterstützt. Besondere Rahmenbedingungen entstehen durch die Berücksichtigung aktueller Ausprägungen verteilter Systeme, d.h. insbesondere durch die Integration heterogener stationärer und mobiler Systeme, der Berücksichtigung entsprechender dynamischer Systemumgebungen sowie den Anforderungen aus organisatorischer und wirtschaftlicher Sicht.

1.4 Vorgehensweise

Um den Bedarf an Flexibilisierung verteilter Prozessausführung zu erfassen, eine möglichst durchgängige und umfassende Flexibilisierung zu unterstützen und die Anpassung auf die Bedürfnisse von Nutzern und die Eigenschaften möglicher Infrastrukturen zu ermöglichen, geht dieser Arbeit eine Analyse voraus, in welcher Prozesse und Prozessmanagementsysteme im Allgemeinen untersucht und anhand von verschiedenen Fallstudien Anforderungen an die Anpassungsfähigkeit von zentral ausgeführten (nicht-verteilten) und verteilt ausgeführten Prozessen abgeleitet werden. Die zum Verständnis dieser Zusammenhänge notwendigen Grundlagen allgemeiner prozessorientierter Anwendungen werden in Kapitel 2 dieser Arbeit dargestellt.

Eine nachfolgende Untersuchung und Bewertung bestehender Möglichkeiten zur Flexibilisierung von Prozessen im Allgemeinen und eine gesonderte Betrachtung von bestehenden Ansätzen zur Flexibilisierung verteilt ausgeführter Prozesse im Speziellen legt dar, auf welche Weise und wie weit dem ermittelten Bedarf an Anpassungsfähigkeit bisher entsprochen werden kann. Auf der Grundlage des Studiums verschiedener Anwendungsfälle für verteilt ausgeführte Prozesse und deren Bedarf für eine dynamische An-

passbarkeit der Verteilung, der Ableitung allgemeiner Ziele und Eigenschaften einer solchen Verteilung und der Betrachtung klassischer Verteilungsmodelle werden dazu relevante Anforderungen und in diesem Kontext zu beachtende Rahmenbedingungen für den Lebenszyklus solcher Prozesse erarbeitet. Die resultierende Anforderungsdefinition bildet dabei den Ausgangspunkt für die Untersuchung des aktuellen Forschungsstands auf diesem Gebiet. Für die Beschreibung der Untersuchung werden die wichtigsten Ansätze zur Flexibilisierung prozessorientierter Anwendungen im Allgemeinen in Kapitel 3 zusammengefasst. Die Analyse dynamisch verteilt ausgeführter Prozesse im Speziellen mit den hierzu betrachteten Fallstudien und der erarbeiteten Anforderungsdefinition sind Inhalt von Kapitel 4. Der aktuelle Stand der Forschung sowie die Bewertung der betrachteten Ansätze in Bezug auf die gesammelten Anforderungen werden in Kapitel 5 beschrieben.

Auf Basis der gewonnenen Erkenntnisse folgt die Erarbeitung eines Konzepts, um die an der Erstellung, Ausführung und Überwachung von dynamisch verteilt ausgeführten Prozessen beteiligten Personen, Rollen und Systeme möglichst weitgehend zu unterstützen, so dass unter benutzerdefinierten Einschränkungen eine möglichst hohe Anpassungsfähigkeit der verteilten Prozessausführung zur Laufzeit erreicht werden kann. Hierzu erfolgt zunächst eine Zerlegung der Aufgabe in Teilprobleme, deren Einordnung sowie die Identifikation ihrer Interdependenzen und Auswirkungen auf die Flexibilisierung der Prozessausführung. Die als wesentliche Teilansätze resultierenden Strategien zur Flexibilisierung verteilter Prozessausführung sind dementsprechend in ein nicht-invasives Verfahren zur dynamischen Verteilung von laufenden Prozessinstanzen und ein grundlegendes Modell zur Überwachung und Steuerung entfernt ausgeführter Prozesse bzw. Prozesspartitionen gegliedert. Darauf aufbauend erfolgt eine Lösung für die aus der dynamischen Verteilung resultierenden Teilprobleme einer angemessenen Interaktion mit dem Benutzer durch die kontextbasierte Darstellung abstrakter Benutzungsschnittstellen sowie für eine proaktive Anpassung der Verteilung durch die Vorhersage von Ausführungskontexten mittels strukturierter Kontextdatenprognose. Das integrierte Konzept erlaubt somit eine Erkennung von Änderungsbedarf, die Ermöglichung von (echt-) zeitnahen Reaktionen und proaktiven Anpassungen sowie eine weitestgehende Automatisierbarkeit des Anpassungsvorgangs von verteilt ausgeführten Prozessen. Eine Übersicht sowie eine detaillierte Beschreibung der vorgeschlagenen Teilkonzepte erfolgen in Kapitel 6 dieser Arbeit.

Die Anwendbarkeit der erarbeiteten Ansätze wird durch die prototypische Implementierung von auf den oben genannten Konzepten basierenden Middleware-Komponenten überprüft, welche zunächst im Rahmen einer Durchführbarkeitsanalyse auf bestehende Prozessmanagementsysteme und Prozessbeschreibungssprachen angewendet wird. Die Unterstützung verschiedener Verteilungskonfigurationen und Anwendungsszenarien wird durch den praktischen Einsatz in einer Auswahl der genannten Fallstudien erprobt. Eine Betrachtung des Flexibilitätsgewinns im Vergleich zu bestehenden Ansätzen

sowie die Entwicklung von szenario- und benutzerspezifischen Anwendungen zur Nutzung der (neu) bereitgestellten Flexibilität stellen dabei sicher, dass sowohl die Ziele der Arbeit erreicht werden als auch die gewonnenen Ergebnisse praktisch relevant und einsetzbar sind. Die Architektur und Umsetzung der prototypischen Implementierung der wesentlichen Konzepte dieser Arbeit sind in Kapitel 7 zusammengefasst, während die darauf aufbauende Anwendung und Bewertung im darauf folgenden Kapitel 8 beschrieben ist.

Die Arbeit schließt mit der inhaltlichen Abgrenzung des Beitrags dieser Arbeit und der Identifikation von Erweiterungs- und Entwicklungsmöglichkeiten in verwandten Forschungsbereichen. Die entsprechende Zusammenfassung und der Ausblick sind Inhalt von Kapitel 9.

1.5 Ergebnisse und Beiträge

Die vorliegende Arbeit zur Flexibilisierung verteilter Prozessausführung zeigt, dass für die Ausführung prozessorientierter Anwendungen in Bezug auf ihre verteilte Ausführung ein hohes Flexibilisierungspotential besteht, welches bislang aufgrund der Heterogenität und der Abgeschlossenheit existierender Systeme nicht genutzt wird. Dabei kann festgehalten werden, dass unerwünschte starre Strukturen für die Verteilung, die Überwachung, die Repräsentation und die Anpassung der Prozessausführung vermieden werden können, wenn der auszuführende funktionale Prozess nicht selbst für die Beschreibung derartiger technischer und organisatorischer Aufgaben verändert wird, sondern diese angemessen von den ausführenden Prozessmanagementsystemen zur Laufzeit des Prozesses erfüllt werden. Durch die Kooperation verschiedener Prozessmanagementsysteme sowie die Spezifikation und den Austausch von ausführungsrelevanten Informationen kann eine Anreicherung des Prozesses mit statischer anwendungsfremder Logik vermieden und während der gesamten Laufzeit eine bedarfsgerechte Anpassung der Prozessausführung sowohl von Seiten menschlicher Benutzer (Adaptierbarkeit) als auch von Seiten des Systems selbst (Adaptivität) ermöglicht werden. Bestehende prozessorientierte Anwendungen können somit weitestgehend unverändert mit bekannten Softwarewerkzeugen modelliert, ausgeführt und unter benutzerdefinierten Rahmenbedingungen bei Bedarf zur Laufzeit in der Art ihrer Ausführung an sich ändernde Systemumgebungen angepasst werden.

Auf Basis dieser Erkenntnisse lassen sich verschiedenartige Verteilungsmodelle unterstützen, wobei sich das Konzept der *Migration einzelner Prozessinstanzen* als Lösung mit der höchsten Flexibilität erwiesen hat, um sowohl die Verteilung, die Partitionierung und die Zuordnung von Prozessen bzw. Prozessfragmenten zur Laufzeit bestimmen und auch fortlaufend anpassen zu können. Als Erweiterung zum traditionellen (Geschäfts-)-Prozessmanagement, bei dem ein Prozessmodell lediglich eine Schablone für die homogene Ausführung hiervon abgeleiteter Prozessinstanzen darstellt, wird durch den hierbei neu erreichten Grad an Flexibilität eine hohe *Individualisierbarkeit* der Ausführung einzelner Prozessinstanzen in Bezug auf ihre

Verteilung ermöglicht. Besondere Unterstützung wird dabei für die Klasse bestehender Prozessmodelle erreicht, von denen kurzfristig einzelne Prozessinstanzen verschiedenartig verteilt ausgeführt werden sollen. Im Fokus stehen daher insbesondere solche Prozesse, welche in einem dynamischen Umfeld entstehen oder in einem solchen ausgeführt werden. In Hinblick auf die Praxis werden dadurch insbesondere verteilte Anwendungen ermöglicht, welche sich erst spontan zur Laufzeit ergeben (wie zum Beispiel *Ad-hoc-Kooperationen*), welche von einer hohen bzw. dynamischen *Quality-of-Service* profitieren (wie zum Beispiel *verteilte Geschäftsprozesse*) oder deren Ausführung möglichst zeitnah an sich ändernde Umgebungsbedingungen angepasst werden muss (wie zum Beispiel Anwendungen aus den Bereichen des *Mobile Computing*). Desweiteren wird die entfernte Ausführung von Prozessen nach verschiedenartigen Benutzervorgaben unterstützt. Dies erlaubt die flexible Umsetzung neuartiger und innovativer systemübergreifender und individualisierbarer Produkte, wie zum Beispiel zur Realisierung von *Process-Management-as-a-Service*-Angeboten.

Zusammengefasst gehen folgende Ergebnisse und Beiträge zu den oben genannten Forschungsbereichen aus dieser Arbeit hervor:

- ein Konzept für die logische Partitionierung der Prozessausführung durch die nicht-invasive Migration von Prozessinstanzen, um unter benutzerdefinierten Einschränkungen zur Laufzeit eines Prozesses eine ungeplante dynamische Verteilung der Prozessausführung auf mehrere Ausführungseinheiten zu ermöglichen, und die entsprechende Architektur eines *Migrationsdienstes* als Umsetzung des *Process-Management-as-a-Service*-Modells für verteilt ausgeführte Prozesse [ZKML10, ZHKK10],
- einen Ansatz zur anbieterseitigen Bereitstellung von Überwachungs- und Steuerungsfunktionalitäten auf Basis eines grundlegenden Referenzmodells zur Abbildung eines Prozessmanagementsystems als *verwaltbare Ressource* und dessen Umsetzung mit WSDM [Zap07, vRZ07, ZBH+10],
- einen Ansatz zur Beschreibung *abstrakter Benutzungsschnittstellen* und zu deren kontextbasierter Transformation in konkrete Interaktionselemente, welche die konkreten Eigenschaften der Ausführungsumgebung und den situativen Kontext des individuellen menschlichen Prozessteilnehmers berücksichtigen [ZVBK09, ZBV09],
- eine Softwarearchitektur für das dynamische Anbieten und Nutzen von Diensten in heterogenen Systemumgebungen, insbesondere unter Berücksichtigung dynamischer lokaler Netze und mobiler Dienstnutzer und -konsumenten [ZDL09a, ZDL09b],
- einen Ansatz zur *Kontextdatenprognose* für dynamisch verteilt ausgeführte Anwendungen und deren Nutzbarmachung für die Anpassung der Verteilung prozessorientierter Anwendungen in Form der Vorhersage von Dienstverfügbarkeiten in dynamischen lokalen Netzen [MZL10, ZML11],

- auf Basis der vorgestellten Strategien zur Flexibilisierung der verteilten Prozessausführung eine Untersuchung der dynamischen Verteilung von XPDL- [ZHKK10], WS-BPEL [ZKML10] und BPMN-Prozessen [BDH+12] mittels Migration laufender Prozessinstanzen,
- auf Basis der vorgestellten Strategien zur Flexibilisierung der verteilten Prozessausführung einen Vorschlag zur Optimierung des Ansatzes der *kontextbasierten Kooperation* [ZKL09a, ZKL09b],
- auf Basis der vorgestellten Strategien zur Flexibilisierung der verteilten Prozessausführung ein Konzept zur Realisierung eines kommerziell anwendbaren *Process-Management-as-a-Service*-Szenarios für die dynamische Verteilung von Prozessen zum Einsatz in hybriden Infrastrukturen aus mobilen und stationären Ausführungseinheiten [ZL10].

Durch eine dienstorientierte Architektur können die genannten Beiträge als unabhängige Komponenten realisiert und so als anpassbare Middleware umgesetzt werden. Eine prototypische Implementierung einer solchen Middleware-Plattform stellt das Ergebnis des praktischen Teils dieser Arbeit dar. Die entstandenen Komponenten können wiederum als Teil bestehender Middleware-Infrastrukturen zur Verwaltung von dienstorientierten Architekturen eingesetzt oder auch als eigenständige Dienste integriert und genutzt werden.

2 Grundlagen prozessorientierter Anwendungen

Bedingt durch den steigenden Bedarf an Anpassungsfähigkeit, die Notwendigkeit von Kostensenkungen und den Wunsch nach einer gleichzeitig möglichst hohen Softwarequalität hat sich die Anwendungsentwicklung insbesondere im Bereich der verteilten Systeme in den letzten Jahren stark verändert. So erfolgt die Entwicklung von komplexen Anwendungen nicht mehr nur monolithisch, sondern es wird verstärkt Wert auf die systematische Wiederverwendbarkeit von ausgereiften Komponenten und autonomen Anwendungen gelegt, die für ihren speziellen Einsatzzweck individuell entwickelt und getestet werden [Szy02, And04, Ses10].

Betrachtet man die Vielzahl der durch Softwaresysteme zu unterstützenden Anwendungsaufgaben in der realen Welt, so ist festzustellen, dass viele von ihnen schon in natürlicher Weise aus mehreren Teilschritten aufgebaut sind, welche einem individuellen Ablauf folgen. Beispiele sind Arbeitsabläufe und Produktionsprozesse in Unternehmen, medizinische Untersuchungs- und Behandlungsvorgänge, Verwaltungsaufgaben in öffentlichen Einrichtungen oder das private Anfertigen, Überprüfen und Versenden von Dokumenten, wie z. B. einer Steuererklärung. Dabei kann die Bearbeitung derartiger Vorgänge in der realen Welt durchaus unter Beteiligung verschiedener Systeme und Personen erfolgen, die sich auch an geographisch unterschiedlichen Orten befinden oder verschiedenen Organisationen angehören können.

Die optimale Abbildung von systemübergreifenden Vorgängen der Realwelt auf Basis lose gekoppelter Anwendungen und Benutzer ist Gegenstand der Entwicklung *prozessorientierter Softwaresysteme* [ODvdA+09]. Ein wichtiges Mittel zur Integration geeigneter Softwarekomponenten, die in einer bestimmten Abfolge nacheinander, parallel oder voneinander bedingt ausgeführt werden sollen, ist die Beschreibung des *Kontroll-* und *Datenflusses* zwischen den beteiligten Systemen. Diese kann entweder statisch innerhalb des Programmcodes einer integrierenden Anwendung oder als eigenständig interpretierbare Kompositionsanweisung erfolgen [MR09], wobei nicht einzelne Anwendungsfunktionalitäten selbst beschrieben werden, sondern deren komplexes Zusammenspiel im Ablauf abgebildet wird [CRW98]. Die aus einer losen Kopplung der Systeme resultierende Flexibilität ermöglicht dabei eine hohe Anpassungsfähigkeit der prozessorientierten Anwendung im Ganzen als auch der beteiligten Teilsysteme. Auf diesem Entwicklungsansatz beruhende Softwaresysteme sind daher gut für ablauforientierte Anwendungsfälle geeignet, welche einer hohen Systemdynamik in Hinblick auf einen kontinuierlichen Verbesserungsprozess unterliegen.

In diesem Kapitel werden die wesentlichen Eigenschaften und Ausprägungen allgemeiner prozessorientierter Anwendungen vorgestellt. Dazu

wird zunächst der dieser Arbeit zugrunde liegende Begriff des *Prozesses* eingeführt und die Reichweite seiner Anwendbarkeit identifiziert. Es folgt eine genauere Betrachtung der Modellierung, Beschreibung und technischen Ausführung von Prozessen sowie die integrierte Darstellung des allgemeinen Lebenszyklus prozessorientierter Anwendungen.

2.1 Einführung und Begriffsklärung

Der für diese Arbeit zentrale Begriff des *Prozesses* wird in der Literatur und in der Praxis je nach Fachgebiet und Blickwinkel teilweise sehr unterschiedlich verstanden und verwendet. Den aktuellen Verwendungsformen des Begriffes sowohl im wissenschaftlichen als auch im umgangssprachlichen Bereich unterliegt jedoch die allgemeine und intuitive Bedeutung eines *gerichteten, inhaltlich abgeschlossenen Vorgangs* oder *Geschehens* mit einer gewissen *zeitlichen Dynamik* und der Eigenschaft, potentiell *mehrere Teilschritte* oder *Phasen* zu umfassen [FH93, VB96, BR98, All05].

In der Informatik kann bei der Einordnung des Prozessbegriffs zwischen einer systemnahen Sicht und einer erweiterten Sicht auf Anwendungsebene unterschieden werden. Die *systemnahe Sicht* setzt den Fokus auf den Ablauf einzelner Operationen als eine partiell geordnete Menge von Prozessschritten [FH93]. So wird z. B. die durch ein Betriebssystem verwaltete Ausführung eines Softwareprogramms mit seinen konkreten Ein- und Ausgaben sowie seinem Zustand als (Betriebssystem-) Prozess bezeichnet [Thu04, Tan03]. Eine *erweiterte Sichtweise* des Prozessbegriffs hat sich im Rahmen der Softwareentwicklung etabliert, wo die Betrachtung und Abbildung von Unternehmensvorgängen, Arbeitsabläufen, Handlungsvorschriften oder allgemeinen Vorgehensweisen im Mittelpunkt steht [Joe00] und somit eine *anwendungsorientierte Verwendung* des Begriffes *auf fachlicher Ebene* der Software impliziert.

Ausgehend von dieser anwendungsorientierten Sicht hat insbesondere der Begriff des *Geschäftsprozesses* bzw. der englische Begriff des *Business Process* die Bedeutung des Prozessbegriffs in der Informatik und Wirtschaftsinformatik geprägt. Hierbei wird jedoch oft diskutiert, ob der Präfix „Geschäft-“ bzw. das englische Wort „Business“ im engeren betriebswirtschaftlichen Kontext oder in der allgemeineren Bedeutung eines durch Informations- und Kommunikationstechnologie abzubildenden *fachlichen Anwendungsfalls* der Realwelt (engl. *Business Case*) zur Abgrenzung rein technischer Vorgänge aufzufassen ist [All05, Gad10, SWG+07, FRH10]. Abbildung 2.1 zeigt eine Übersicht zur Einordnung dieser Sichtweisen und der nachfolgenden Diskussion.

Ein *Geschäftsprozess* im strengen betriebswirtschaftlichen Sinne bildet das *Kerngeschäft* eines betrachteten Wirtschaftsunternehmens ab. Abhängig vom Fokus verschiedener Autoren ist ein solcher Geschäftsprozess daher immer auf ein übergeordnetes Unternehmensziel hin ausgerichtet, stellt einen Teil der Wertschöpfungskette des Unternehmens dar [Mül05], erzeugt einen Kundennutzen [Eur09, FRH10] und/oder weist Schnittstellen zu externen Marktpartnern auf [VB96]. Weniger streng wirtschaftlich orientierte Begriffsdefini-

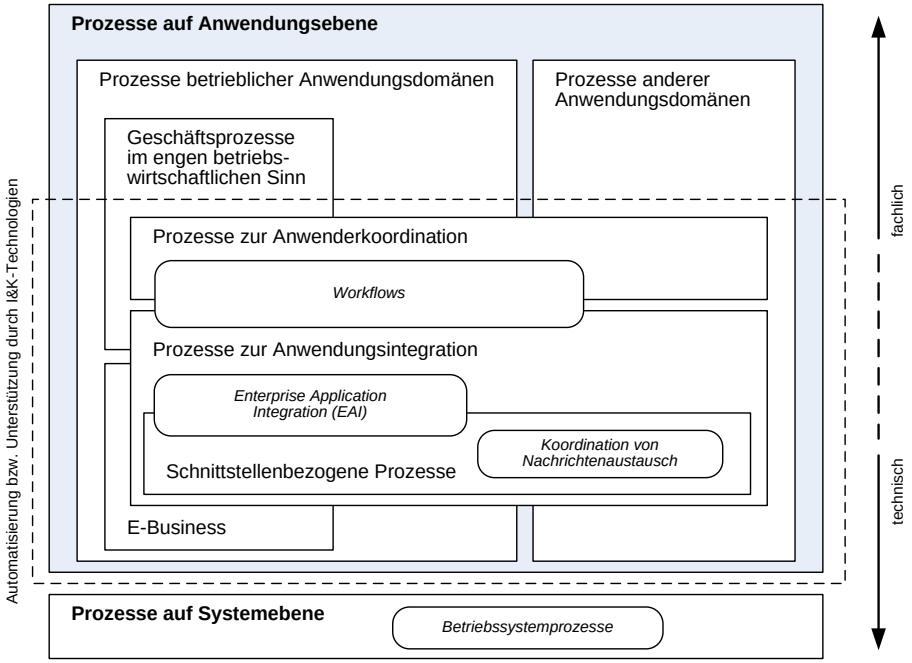


Abbildung 2.1: Darstellung und Einordnung des Prozessbegriffs in der Informatik

tionen fordern lediglich, dass es sich bei einem (Geschäfts-)Prozess um eine zusammenhängende, abgeschlossene, zeitliche und sachlogische Abfolge von Tätigkeiten handelt, die zur *Erfüllung einer betrieblichen Aufgabe* erledigt werden [All05, Sta06] oder zur *Bearbeitung eines betriebswirtschaftlich relevanten Objekts* notwendig sind [VB96]. In Hinblick auf dieses betriebswirtschaftlich relevante Objekt gibt der Autor STAUD [Sta06] zu bedenken, dass die Ebene, auf der die Prozesse und Aufgaben betrachtet werden, ein subjektiver Faktor ist, der durch den Modellierer oder den Zweck der Modellierung festgelegt werden kann und auch die Länge der Prozesse bzw. deren Granularität beeinflusst [Sta06]. ALLWEYER [All05] fügt seiner Analyse über die (allgemeinere) betriebswirtschaftlich orientierte Verwendung des Begriffes hinzu, dass eine *Leistung* in Form von Material- und oder Informationstransformation erbracht wird, sowohl Prozesse innerhalb eines Unternehmens als auch unternehmensübergreifende Prozesse erfasst werden und diese sowohl *durch Computersysteme automatisiert* als auch manuell ausgeführt werden können. Dabei reicht das Spektrum von der Gesamtbetrachtung umfassender Geschäftsstrategien bis zur einzelnen Betrachtung kleinerer Unterstützungsprozesse [All05]. Unabhängig davon beschreibt auch ÖSTERLE [Öst95] den Geschäftsprozess als eine Abfolge von Aufgaben, die über mehrere *organisatorische Einheiten* verteilt sein können und deren Ausführung von informationstechnologischen Anwendungen unterstützt wird. Dabei ist ein Prozess sowohl Produzent und Konsument von Leistungen und verfolgt von der Prozessführung gesetzte Ziele. Da der Geschäftsprozess als spezielle Form der Ablauforganisation die Geschäftsstrategie konkretisiert und sie mit einem Informationssystem verknüpft, sieht ÖSTERLE den Geschäftsprozess als Binde-

glied zwischen der Unternehmensstrategie und der Systementwicklung bzw. den unterstützenden Informationssystemen [Öst95].

Im engen Zusammenhang dazu haben sich ebenfalls die Begriffe des *Workflows* und der *Enterprise Application Integration (EAI)* etabliert, wobei jeweils im Vordergrund steht, dass ein Geschäftsvorgang in einem Wirtschaftsunternehmen (ggf. auch nur teilweise) technisch abgebildet und mit Hilfe von Computersystemen unterstützt dargestellt, ausgeführt bzw. automatisiert wird [VB96, LR00, Joe00, DGH03, Mül05]. Historisch gesehen entspringt jedoch der Begriff des *Workflows* dem Dokumentenmanagement bzw. der computergestützten Gruppenarbeit (*Computer Supported Cooperative Work*) [ACKM04, Mül05, Str08] und stellt aufgrund dieses Ursprungs in der Regel die Koordination menschlicher Prozessteilnehmer entlang eines (personalintensiven) Ablaufs in den Vordergrund (*Anwenderkoordination*) [Str08]. Systemintegrierende Prozesse hingegen haben ihren Ursprung im Bereich der *Anwendungsintegration* mit dem Schwerpunkt der prozessbasierten Koordination heterogener Anwendungssysteme entlang einer (anwendungsintensiven) Prozesskette [Str08]. Bedingt durch die wachsende Bedeutung und Verbreitung von Software und die zunehmende Verschmelzung beider Subkategorien [Ses10] werden in vielen Werken die Begriffe Geschäftsprozess, Workflow und/oder Prozess sogar einander gleichgesetzt oder auf eine explizite Unterscheidung verzichtet [Joe00, vdAvH02, All05, Str08, FRH10].

Zudem wird deutlich, dass im betriebswirtschaftlichen Kontext ein zunehmend starker Zusammenhang zwischen betrieblichen Abläufen und deren Unterstützung durch Informations- und Kommunikationstechnologien gesehen wird und somit in Hinblick auf die Informatik auch eine *automatisierungsbezogene Sichtweise* des Prozesses gerechtfertigt ist [All05]. Im Rahmen des *E-Business* kann dieser Bezug zu Informations- und Kommunikationssystemen sogar als zentraler Aspekt betrachtet werden, da hier ein Austausch elektronischer Dokumente zwischen den Informationssystemen beteiligter Partner im Vordergrund steht und der Fokus des Geschäftsprozesses auf der Darstellung des hierzu relevanten Datenflusses liegt (*schnittstellenbezogene Verwendung* des Prozessbegriffs) [All05]. Im Gegensatz zur Abbildung von globalen Geschäftstrategien und zur Unterstützung einer fachlichen Anwenderkoordination liegt hierbei eine sehr feingranulare technische Sichtweise des Prozessbegriffs vor, welche *konversationsorientierte Systeme* durch die Modellierung des Ablaufs von Sprechakten unterstützt [Sch00].

Die rein auf den Unternehmenskontext bezogene Definition von Prozessen erweist sich bei genauerer Betrachtung der anwendungsorientierten Verwendung des Begriffes in der Informatik als sehr restriktiv. So stellt ein Unternehmen lediglich einen speziellen Betriebstyp in marktwirtschaftlichen Systemen dar, welcher über die Merkmale des erwerbswirtschaftliche Prinzips (d. h. Streben nach Gewinnmaximierung), dem Prinzip des Privateigentums und des Autonomieprinzips (d. h. Selbstbestimmung des Wirtschaftsplans) verfügt [DS08]. Dies schließt Anwendungsdomänen aus anderen Bereichen, wie öffentliche Betriebe und Verwaltungen, Privathaushalte und

Körperschaften des Privatrechts aus, welche den Prozessbegriff zur Abbildung und Automatisierung von relevanten Vorgängen auf ihren Anwendungskontext übertragen. E-Government-Prozesse [SKH03, SOB+03, May04a, Hac05], Abläufe im Gesundheitswesen [Müh04] oder Prozesse zur Kooperation zwischen persönlichen mobilen Systemen [Kun08] stellen nur wenige Beispiele dar, die anderen als rein kommerziellen Zwecken dienen.

In Hinblick auf ein sehr weites Spektrum von Anwendungsszenarien stellen LEYMANN und ROLLER [LR00] fest, dass die Frage, was ein (Geschäfts-)Prozess ist, davon abhängt, was das Geschäft bzw. der Anwendungsfall des jeweiligen Initiators des Prozesses ist. Daher definieren sie einen (Geschäfts-)Prozess nur abstrakt anhand seiner Eigenschaften als Menge von Aktivitäten, die in einer bestimmte Reihenfolge ausgeführt werden [LR00]. VAN DER AALST und VAN HEE [vdAvH02] verfolgen eine ähnlich freie Sichtweise auf den Prozessbegriff, indem sie den Geschäftsprozess als Bearbeitung eines (Anwendungs-) Falles (engl. *Case*) definieren, wobei ein solcher (Anwendungs-) Fall ein konkretes oder abstraktes Objekt darstellen kann. Ein jeder solcher (Anwendungs-) Fall hat demnach einen Anfang und ein Ende und enthält einen Vorgang, der aus einer Menge von auszuführenden Aufgaben und einer Menge von Bedingungen besteht, welche die Reihenfolge zwischen diesen Aufgaben vorgeben [vdAvH02].

Um alle relevanten Anwendungsgebiete abzudecken, soll in dieser Arbeit der Prozessbegriff möglichst neutral und abstrakt gehalten werden und verwendet daher den allgemeineren Begriff der *Organisation* bzw. der *Organisationseinheit* anstelle des Unternehmens. Dabei stellen *Geschäftsprozesse* bzw. *Workflows* eine besondere Untermenge von Prozessen dar [VB96], bei der die betrachtete Organisation ein Wirtschaftsbetrieb ist bzw. bei denen neben einer Anwendungsintegration auch die Koordination von Anwendern im Vordergrund steht. Angelehnt an die Definitionen von GADATSCH [Gad10] und WESKE [Wes07] ist ein Prozess hier demnach wie folgt definiert:

> *Ein **Prozess** [auf Anwendungsebene] ist eine zielgerichtete, zeitlich-logische, zusammenhängende und abgeschlossene Abfolge festgelegter Aktivitäten, welche arbeitsteilig von mehreren Organisationen oder Organisationseinheiten unter Nutzung von Informations- und Kommunikationstechnologien ausgeführt werden können.*

Eine *Aktivität* stellt dabei eine logische Einheit der Bearbeitung dar, welche auch in sich strukturiert und somit weiter detailliert sein kann [Gad10]. Eine Aktivität wird als *maximal detailliert* oder als *atomar* bezeichnet, wenn die damit verbundene fachliche Funktion als Ganzes in einem Zug von einer Ressource erfüllt werden kann [vdAvH02, Gad10]. Eine *Ressource* ist ein generisches Konstrukt für eine verantwortliche Person, eine Maschine oder Softwarekomponente oder eine Gruppe von Personen oder Maschinen bzw. Softwarekomponenten, welche für die Ausführung von Aktivitäten benötigt wird [vdAvH02, Sta06, Eur09, FRH10]. Personen und Maschinen können dabei

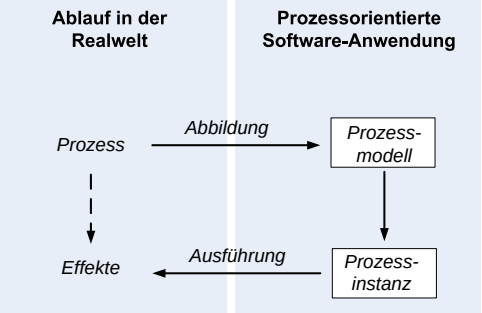


Abbildung 2.2: Zusammenhang zwischen Abläufen der Realwelt und prozessorientierten Anwendungen (in Anlehnung an (LR00))

durchaus auch hinter einer Softwarekomponente gekapselt sein [Thu04]. Folglich kann eine Aktivität entweder manuell, teil-automatisiert oder automatisiert erfüllt werden [Sta06], wobei sowohl die konkreten Ressourcen als auch die Art der Ausführung nicht zwingend im Voraus festgelegt sein müssen, sondern über abstrakte *Rollen* definiert sein können [Mül05]. Ressourcen können hierarchie- und standortübergreifend sein [Mül05] und somit verschiedenen Organisationen oder Organisationseinheiten angehören. Die Ausführung eines Prozesses wird daher zusammenfassend von drei Dimensionen bestimmt: der *Prozesslogik*, die beschreibt, welche Aktivitäten in welcher Reihenfolge ausgeführt werden sollen, dem *organisatorischen Kontext*, in dem ein Prozess eingebunden ist, und der am Prozess beteiligten *Informations- und Kommunikationssysteme* [LR00, Joe00].

Ein so definierter Prozess besitzt mindestens einen definierten Anfang (*Auslöser des Prozesses* bzw. *Ereignis*) und ein definiertes Ende (*Endzustand*) und kann zwischen Anfang und Ende einen unterschiedlichen Verlauf nehmen [Mer95, Mül05, All05]. Ein *Prozessmodell* stellt dazu ein Muster für die Struktur aller relevanten Ablaufvarianten in dem betrachteten Abschnitt der Realwelt dar [LR00, Thu04] (vgl. Abbildung 2.2). Der Verlauf eines individuellen Prozesses (d. h. einer *Prozessinstanz*) wird durch die Eingaben (*Input*) von konkreten Objekten sowie von internen und externen Ereignissen und die Zuordnung zu konkreten Ressourcen beeinflusst. Jede Aktivität wird dabei auf Grundlage einer (ggf. leeren) Eingabe ausgeführt, wobei die Eingabe durch jeweils eine geeignete Ressource verarbeitet wird. Mögliche Ergebnisse dieser Aktionen sind *externe Effekte* oder die Ausgabe einer Aktivität (*Output*), welche als erneute Eingabe für weitere Aktivitäten zur Verfügung stehen kann [Mül05]. Betrachtet man eine (Teil-)Folge von Aktivitäten, so kann zwischen *informationsflussbasierten* und *materialflussbasierten* Vorgängen [Joe00] bzw. einer *Informations-* und *Materialtransformationen* [All05] unterschieden werden, je nachdem ob als Objekte Daten oder physische Materialien in die Aktivitäten eingehen bzw. von ihnen ausgehen. In dieser Arbeit werden in erster Linie informationsflussbasierte Vorgänge betrachtet, wobei nicht explizit ausgeschlossen werden soll, dass hierdurch auch Materialien erzeugt oder verar-

beitet werden. Für den Prozessinitiator wird durch die Ausführung des Prozesses in jedem Fall eine *Leistung* bzw. ein *Mehrwert* erzeugt [All05].

Prozesse auf Anwendungsebene stellen eine wichtige konzeptuelle Basis für die Integration und Koordination von lokalen oder verteilten Ressourcen dar, bei der die Abbildung von relevanten strukturierten und arbeitsteilig ausgeführten Vorgängen der Realwelt und deren geeignete Umsetzung mit Hilfe von Informations- und Kommunikationstechnologien im Mittelpunkt der Betrachtung steht. Durch die lose Verknüpfung von Hard- und Softwarekomponenten und der Interaktion mit menschlichen Akteuren in einer für den jeweiligen Anwendungsfall geeigneten Form entsteht eine höherwertige Anwendung, welche sich nicht durch die Komplexität und Starrheit monolithischer Anwendungen auszeichnet, sondern den Kriterien der Wiederverwendbarkeit, Flexibilität und Zuverlässigkeit als zentrale softwaretechnische Anforderungen nachkommt. Nach diesem Paradigma erstellte Anwendungen werden als *Workflow-based Applications* [LR00], *Process-oriented Software Systems* [ODvdA+09] oder dem allgemeinen Prozessbegriff dieser Arbeit entsprechend als *prozessorientierte Anwendungen* bezeichnet.

2.2 Klassifizierung von Prozessen

Unabhängig vom speziellen Anwendungskontext treten Prozesse in der Realwelt in unterschiedlichen Ausprägungen auf, welche sich anhand von allgemeinen Eigenschaften kategorisieren lassen und Aufschluss über die technische Unterstützbarkeit der Anwendungsfälle und die damit verbundenen Anforderungen und Herausforderungen geben. Tabelle 2.1 zeigt einen Überblick über die wesentlichen Eigenschaften zur Klassifizierung von Prozessen.

Ein wesentlicher Aspekt bei der Entwicklung prozessorientierter Anwendungen besteht darin, inwieweit der betrachtete Vorgang überhaupt im Voraus planbar ist und ob er in festen Strukturen beschrieben werden kann. Der *Strukturgrad* gibt dabei an, ob die Details aller möglichen Abläufe im Voraus festgelegt werden können (*strukturierter Prozess*), ob es sich um einen eher kreativen Prozess handelt, bei dem einige Details offen sind (*semistrukturierter Prozess*) oder ob bei dem Vorgang weitestgehend alle Zusammenhänge und Schritte im Voraus unbekannt sind (*unstrukturierter Prozess*) [BS98, All05, Wes07].

Die genaue Kenntnis der Struktur eines Vorgangs in der Realwelt wird vor allem durch seine Wiederholung bzw. durch die Reproduzierbarkeit von Ergebnissen beeinflusst. Dabei muss die Wiederholung nicht zwingend zu genauen Zeitpunkten erfolgen, sondern kann auch unregelmäßig aufgrund von bestimmten Ereignissen auftreten (*Auftretensart*). In Zusammenhang mit der *Häufigkeit* des Auftretens ist die Wiederholung von strukturierten Vorgängen – auch aus den Gesichtspunkten der Automatisierung – ein wesentlicher Aspekt, da oftmals erst häufig in gleicher oder ähnlicher Weise wiederholte Vorgänge den Aufwand der Anschaffung und des Unterhalts von Systemen zu deren Automatisierung rechtfertigen [LR00, All05, Wes07]. Bei strukturierten,

Klassifizierung	Prozessart	Beschreibung
Strukturgrad	strukturiert	vollständig vorherbestimmt
	semi-strukturiert	enthält Elemente, die nicht im Voraus beschreibbar sind
	unstrukturiert	strukturelle Entscheidungen können nicht im Voraus getroffen werden
Auftretensart	zyklisch	regelmäßiges Auftreten zu bestimmbaren Zeitpunkten
	wiederholt	unregelmäßiges Auftreten
	einmalig	Vorgang ohne Wiederholung (Projekt)
Häufigkeit	häufig	häufig auftretende Vorgänge
	situativ	je nach Anfragesituation
	selten	selten oder einmaliges Auftreten
Automatisierungsgrad	vollautomatisiert	keine manuellen Aktivitäten
	teilautomatisiert	teilweise manuelle Aktivitäten
	manuell	ausschließlich manuelle Aktivitäten
Dauer	kurz	Abwicklung im Sekunden- oder Minutenbereich
	mittel	Vorgänge mit einer Dauer von einigen Stunden bis wenige Tage
	lang	Vorgänge über Wochen, Monate oder sogar Jahre
Abstraktionsgrad	hoch	Strategische, stark verdichtete Vorgänge
	gering	Operativ detaillierte Vorgänge
	technisch	technisch umsetzbar bzw. ausführbar
Verteilungsgrad	intern	Bearbeitung innerhalb einer abgeschlossenen Organisationseinheit
	verteilte Ressourcen	Beteiligung der Ressourcen mehrerer Organisationseinheiten
	verteilte Ausführung	Steuerung des Prozesses unter Beteiligung mehrerer Organisationseinheiten

Tabelle 2.1: Klassifizierung von Prozessen (nach (LR00, All05, Gad10, Lie07, Wes07, Ham09))

aber dennoch selten oder sogar nur einmalig auftretenden zeitlich begrenzten Vorgängen spricht man auch häufig anstelle eines Prozesses von einem *Projekt* [All05]. Als Beispiel kann man die Bestellung in einem Versandhaus oder das Durchführen einer Banküberweisung als häufig bzw. in gleicher Art und Weise wiederholt klassifizieren, während der Bau einer Brücke oder das Anlegen eines Gartens einen individuellen Vorgang mit Projektcharakter darstellt [All05].

Damit zusammenhängend stellt der *Automatisierungsgrad* von Prozessen eine weitere Perspektive dar, welche beschreibt, in welchem Maße der Prozess unter Beteiligung menschlicher Akteure ausgeführt wird. Dabei wird vor allem betrachtet, ob der Prozess manuelle Aktivitäten aufweist, nicht aber, ob Menschen zur Steuerung des Prozesses eingesetzt werden. Prozesse mit ausschließlich manuellen Aktivitäten werden daher als *manuell*, Prozesse ohne manuelle Aktivitäten als *(voll-)automatisiert* und Mischformen als *teilautomatisiert* bezeichnet [LR00, Wes07]. Dabei können auch Prozesse mit ausschließlich oder überwiegend manuellen Tätigkeiten durch Computersysteme verwaltet bzw. gesteuert werden, was der klassischen Bedeutung eines *Workflows* entspricht [Mül05]. *Automatisierungsgrad* und *Automatisierbarkeit* von Prozessen in Bezug auf deren computergestützte Ablaufsteuerung bedürfen daher einer klaren Abgrenzung [Gad10].

Dauer und *Umfang* stellen weitere wichtige Eigenschaften eines Prozesses dar, wobei die Dauer oft abhängig von der Anzahl der Aktivitäten und insbesondere von der Beteiligung menschlicher Akteure ist [All05]. Sehr *kurze Prozesse* mit einer Dauer im Sekundenbereich können in der Regel nur

durch einen hohen Automatisierungsgrad erreicht werden. Beispiele hierfür sind elektronische Banküberweisungen und Flugbuchungen. *Prozesse mit einer mittleren Dauer* im Bereich von Minuten, Stunden oder Tagen sind oft durch manuelle Tätigkeiten oder durch das Warten auf temporär nicht verfügbare Ressourcen gekennzeichnet. Ein Beispiel stellt ein typischer Prozess in einem Versandunternehmen von der Auftragsannahme im Versandhaus bis zur Ankunft der Ware beim Besteller dar. *Lang andauernde Prozesse* umfassen schließlich Vorgänge der Realwelt, welche sich unter Umständen sogar über Wochen, Monate oder Jahre erstrecken können und oft komplexen Zusammenhängen unterlegen sind. Beispiele sind das Studium eines Studenten an einer Universität oder der zuvor genannte Bau eines Schiffes. Oft kommt es insbesondere bei mittleren und langen Prozessen zu der Notwendigkeit, Änderungen aufgrund externer Ereignisse vornehmen zu müssen. Die Anzahl der gleichartigen Abläufe ist daher oft geringer und es ist nur selten eine vollständige Automatisierung möglich.

Der *Abstraktionsgrad* eines Prozesses beschreibt, auf welcher Ebene der jeweilige Anwendungsfall betrachtet wird. Ein hoher Abstraktionsgrad drückt aus, dass der Prozess *stark verdichtet* dargestellt wird, durch eine informelle oder semi-formale Beschreibung spezifiziert und nur durch eine weitere Verfeinerung (automatisiert) ausgeführt werden kann. Hierzu gehören Prozesse auf organisatorischer Ebene wie die oben beschriebenen Geschäftsprozesse im strengen betriebswirtschaftlichen Sinn [Wes07, Gad10]. Verfeinerte Prozesse mit einem geringeren Abstraktionsgrad stellen in der Regel *operative Vorgänge* mit einer detaillierten Beschreibung dar. *Technische Prozesse* enthalten bereits oft Informationen über die technische und organisatorische Umgebung, die zu ihrer Ausführung notwendig sind und sind aufgrund ihrer detaillierten formalen Beschreibung in der Regel direkt ausführbar [Wes07]. Davon abzugrenzen sind rein technische Vorgänge, welche keinerlei organisatorische Informationen zum eigentlichen Anwendungsfall enthalten, sondern ausschließlich *Unterstützungsprozesse* beschreiben, um die Automatisierung oder Strukturierung von höherwertigen Vorgängen zu ermöglichen. Beispiele sind Vorgänge zur Übersetzung einer Nachricht oder zur Transformation von komplexen Datenstrukturen [Wes07].

Relativ unabhängig von den zuvor genannten Kategorien kann schließlich zwischen *organisationsinternen* und *organisationsübergreifenden* Prozessen in Hinblick darauf unterschieden werden, ob eine oder mehrere Organisationeinheiten an der Erstellung und Ausführung eines Prozesses beteiligt sind [Wes07]. Dabei kann bei einer organisationsübergreifenden Verteilung des Prozesses differenziert werden, ob lediglich die Aktivitäten des Prozesses von Ressourcen unterschiedlicher Organisationseinheiten ausgeführt werden (*verteilte Ressourcen*), oder ob die Steuerung und damit die Kontrolle über die Prozessausführung selbst verteilt ist (*verteilte Prozessausführung*) [SO04, FRH10]. Die Automatisierbarkeit organisationsübergreifender Prozesse birgt in jedem Fall große Herausforderungen. In den folgenden Kapiteln

wird die besondere Ausprägung von verteilt ausgeführten Prozessen als zentraler Aspekt dieser Arbeit genauer betrachtet und weiter verfeinert.

2.3 Prozessmodellierung und -beschreibung

Die Identifikation, Ableitung und Beschreibung der Ablaufstruktur eines Anwendungsfalls sind Gegenstände der *Prozessmodellierung*. Ein Modell dient als Abstraktion bzw. Reduktion eines Objekts der realen Welt mit dem Zweck, die für die Durchführung einer bestimmten Aufgabe wesentlichen Eigenschaften dieses Objektes explizit abzubilden. Als Ergänzung dieser grundlegenden und allgemeinen Bedeutung steht in der Informatik insbesondere die geeignete Nachbildung des relevanten Gegenstandsbereichs als *Grundlage für eine maschinelle Verarbeitung* im Vordergrund [VB96, Flo97]. Die Prozessmodellierung dient daher in der Informatik in der Regel nicht nur als Mittel zu einem besseren oder gemeinsamen Verständnis eines prozessorientierten Anwendungsbereichs und damit als Ausgangspunkt für mögliche Verbesserungen, sondern stellt bei einer hinreichend formalen Repräsentation des Vorgangs auch eine wichtige Basis für die Unterstützung des Vorgangs durch Informations- und Kommunikationstechnologie dar [Joe00].

Angelehnt an das *ARIS*-Konzept (*Architektur integrierter Informationssysteme*) werden prozessorientierte Anwendungen in der Regel durch ein mehrstufiges Vorgehen entwickelt (vgl. [Mül05, Gad10, Str08, FRH10]). Eine anwendungsneutrale Übersicht der verschiedenen Ebenen der Prozessmodellierung ist in Abbildung 2.3 visualisiert. Dabei wird zunächst der abzubildenden Anwendungsfall analysiert und durch eine zumeist graphische, nichttechnische Beschreibung auf einer fachlich-konzeptionellen Ebene dargestellt [Gad10, Str08]. Es können dabei sowohl Ist- als auch Sollprozesse modelliert bzw. ineinander überführt werden [Mül05]. Bei komplexen Prozessen bietet es sich dazu an, die Modellierung mit einer groben, ergebnisorientierten Darstellung zu beginnen, welche ein grundsätzliches Verständnis des Prozesses ermöglicht und zur Kommunikation zwischen Prozessbeteiligten dienen kann (*Ebene 1*). Diese erste Darstellung enthält in der Regel nur wenige, stark verdichtete Aktivitäten, zeigt keine oder nur wenige mögliche Ablaufvarianten und abstrahiert von möglichen Fehlern oder anderen Besonderheiten [FRH10]. In betriebswirtschaftlicher Hinsicht entspricht dieser erste Schritt der Identifikation und expliziten Formulierung der Geschäftsstrategie in Form von Kerngeschäftsprozessen [Gad10], weshalb man auf dieser Ebene auch von *strategischen Prozessmodellen* spricht.

In einem zweiten Schritt werden die so gewonnenen Prozessmodelle verfeinert, so dass alle relevanten Details der tatsächlichen Abwicklung dargestellt werden können (*Ebene 2*) [FRH10]. Durch eine hierarchische Dekomposition werden dabei die verdichteten Aktivitäten solange in weitere Teilschritte zerlegt, bis atomare Aktivitäten vorliegen, die sich fachlich nicht weiter aufteilen lassen [Wes07]. Die Modellierung sollte dennoch an dieser Stelle nur Aktivitäten enthalten, die für den fachlichen Ablauf der Prozesse erforderlich sind

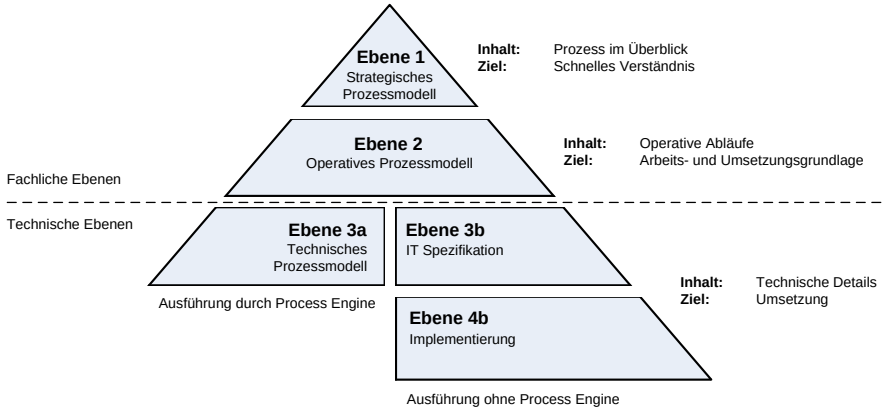


Abbildung 2.3: Ebenen der Prozessmodellierung (FRH10)

[Mül05]. Das resultierende *operative Prozessmodell* dient somit als genaue Anleitung zur Durchführung des Prozesses und somit als geeigneter Ausgangspunkt für eine technische Umsetzung.

Mögliche Darstellungsformen für Prozessmodelle der fachlich-konzeptionellen Ebenen sind natürlichsprachliche Beschreibungen als Text, tabellarische Formen oder anschauliche graphischen Darstellungen mit oder ohne bestimmte Notationen [All05]. Mathematisch formale Modelle oder gar formale Programmiersprachen sind auf diesen Ebenen der Prozessmodellierung nicht üblich. Operative Prozessmodelle können daher in der Regel noch nicht direkt maschinell verarbeitet werden und sind somit im Allgemeinen noch nicht ausführbar.

Für die computergestützte Ausführung von Prozessen mittels eines Prozessmanagementsystems ist die technische Beschreibung der Prozesse in einer hinreichend detaillierten und konkretisierten Form erforderlich. Fehlende, nicht exakte oder widersprüchliche Arbeitsanweisungen müssen vermieden werden, da sie in der Regel nicht automatisch ergänzt bzw. korrigiert werden können. Die Prozesse müssen daher in einer streng festgelegten Syntax modelliert werden, welche von einer Ausführungseinheit mit eindeutig festgelegter Ausführungssemantik interpretiert werden kann [Thu04]. Die genaue weitere Vorgehensweise der Modellierung in Richtung einer technischen Umsetzung ist dabei von der gewünschten Art der Umsetzung und der vorliegenden technischen Infrastruktur abhängig. Abbildung 2.3 verdeutlicht auf den technischen Ebenen zwei mögliche Arten der Umsetzung:

- *Technische Prozessmodelle* können einerseits mit Hilfe einer speziell zur Prozessmodellierung entworfenen, meist deskriptiven, höheren Programmiersprache definiert werden, welche im Rahmen eines *Metamodells* konkrete Sprachkonstrukte zur Beschreibung von Prozessen zur Verfügung stellt und somit eine formale Notation besitzt. Eine solche Sprache wird im Folgenden als *Prozessbeschreibungssprache* bezeichnet. Dabei kann ein ausführbares Prozessmodell spezifiziert werden, welches die notwendigen Verfeinerungen in Hinblick auf eine computer-

gestützte Ausführung des Prozesses enthält und eine Zuordnung von Aktivitäten zu Funktionen der technischen Ressourcen oder zu den Arbeitsplätzen der Mitarbeiter ermöglicht [Mül05]. Die Ableitung des technisch ausführbaren Prozesses entspricht im betriebswirtschaftlichen Kontext der Modellierung eines *Workflows* [Gad10] (vgl. Abschnitt 2.1). Das durch eine Prozessbeschreibungssprache spezifizierte technische Prozessmodell (*Ebene 3a*) wird in diesem Fall von einem spezialisierten, aber dennoch prozessunabhängigen Ausführungssystem (*Prozess-Engine* bzw. *Workflow-Engine*) interpretiert, welches bei einer Instantiierung auf Basis der gemachten Eingaben den Kontrollfluss des Prozesses steuert, die Daten des Prozesses verwaltet und den Aufruf von Ressourcen vornimmt.

- Alternativ kann die Prozesslogik des operativen Prozessmodells auch als statischer Programmcode einer herkömmlichen Programmiersprache ausgedrückt werden oder im Rahmen eines anpassbaren *Enterprise Resource Planning (ERP)* Systems umgesetzt werden [FRH10]. Hierfür ist in der Regel zunächst eine weitere Spezifikation der technischen Systeme nötig, z. B. durch die Erstellung eines DV-Konzepts oder eines Pflichtenheftes (*Ebene 3b*) [FRH10]. Mit der Implementierung folgt dann bei dieser Variante die tatsächliche technische Umsetzung des Prozesses für die gewählte Plattform (*Ebene 4b*) [FRH10].

Neben technischen Prozessmodellen, die direkt zur Ausführung mit Hilfe von Informations- und Kommunikationssystemen verwendet werden, gibt es zudem Beschreibungssprachen, die auf einer Algebra zur formalen Beschreibung von Prozessen basieren, wie beispielsweise dem λ-Kalkül [Bar84] oder dem π-Kalkül [Mil99]. Da die Ausführung eines Prozesses durch die Beschreibung einer mathematisch-formalen Semantik eindeutig definiert ist, werden derartige Ansätze oft zur *Verifikation* von komplexen Prozessmodellen verwendet [Puh06, Puh07]. Oft zeigt sich jedoch erst in der Praxis oder bei einer Simulation des realen Einsatzes, ob die über Struktur und Ablauf getroffenen Annahmen mit der Realität übereinstimmen und die gewünschten Ergebnisse und Effekte bei einer angemessenen Qualität und zu akzeptablen Kosten erreicht werden. Es sind daher häufig Änderungen an der Prozesslogik notwendig, welche bei der zweiten genannten Umsetzungsmöglichkeit (Ebenen 3b bzw. 4b) mit einem hohem Aufwand verbunden sind [Mül05]. Die effektive Reorganisation von Prozessmodellen ohne Programmieraufwand, die Verwaltung vieler (ggf. auch nur im Detail) unterschiedlicher Prozessmodelle und der Vorteil einer plattformunabhängigen und austauschbaren Prozessrepräsentation sind entscheidende Aspekte, die für die Erstellung technischer Prozessmodelle auf Basis eines festgelegten Metamodells sprechen. Die proprietäre Umsetzung von prozessorientierten Anwendungen als klassische Softwareanwendung wurde daher weitestgehend verdrängt. Im Rahmen dieser Arbeit wird im Folgenden daher das resultierende, direkt ausführbare technische Prozessmodell als explizite und formale Repräsentation des relevanten Teils des betrach-

teten Anwendungsbereichs in der Realwelt als Ausgangspunkt weiterer Betrachtungen verwendet (vgl. *Ebene 3a* in Abbildung 2.3).

Eine Überführung von Prozessmodellen von der fachlich-konzeptionellen auf die technische Ebene ist in der Regel mit vielen Herausforderungen verbunden. Die unterschiedlichen Modellierungssprachen und -werkzeuge sind oft nicht ausreichend kompatibel und die Transformation von Prozessmodellen von einer übersichtlichen graphischen Darstellung in eine textuelle, mathematisch-formale oder programmiersprachliche Repräsentation ist oft nicht verlustfrei möglich. Zudem sind oft weitere Informationen und technische Zwischenschritte erforderlich, um die gewünschten Ergebnisse oder Effekte zu erzielen, welche mit der technischen Umsetzung des prozessorientierten Anwendungsfalls beabsichtigt werden [Str08]. Des Weiteren sind auf den verschiedenen Ebenen häufig unterschiedliche Benutzergruppen an der Modellierung des Prozesses beteiligt [Gad10]. Der Entwicklungsprozess von der Analyse bis zur Fertigstellung eines ausführbaren Prozessmodells auf technischer Ebene ist daher insbesondere bei komplexen und lang andauernden Prozessen sehr aufwendig und nimmt oft einen großen Zeitbedarf in Anspruch. Daher ist es sehr unvorteilhaft, wenn durch Änderungen an technischen Systemen oder an organisatorischen Begebenheiten Änderungen am Prozessmodell notwendig werden, die nicht durch eine fachliche Änderung der Prozesslogik motiviert sind. Auch technische Prozessmodelle sollten daher weitestgehend implementierungsunabhängig ausgedrückt werden, d. h. keine Konstrukte von Programmiermethoden oder Produkten verwenden, welche diese Unabhängigkeit nicht gewährleisten können. Umgekehrt sollen auch ablauforganisatorische Modifikationen nicht dazu führen, dass Änderungen an den unterliegenden Softwarekomponenten notwendig werden [Mül05]. Die folgenden Abschnitte werden grundlegenden Konzepte aufzeigen, mit denen eine hinreichend lose Kopplung erreicht wird, um diese Forderungen zu erfüllen.

2.4 Kontrollfluss von Prozessen

Während der Modellierung eines Prozesses kann die Darstellung und die Beschreibung des Prozessmodells unterschiedliche Formen und Detaillierungsgrade annehmen. Als inhärente Gemeinsamkeit wird durch das Prozessmodell jedoch auf jeder Modellierungsebene das gewünschte oder tatsächliche Verhalten des betrachteten Systems ausgedrückt, welches durch eine zeitliche Abfolge von Zustandsübergängen definiert ist. Um die angestrebte Entkopplung auszuführender Aufgaben von konkreten Ressourcen zu erreichen, wird hierzu das abstrakte Konzept der *Aktivität* verwendet, welches in der Regel nur die fachliche Funktion, nicht aber das ausführende Individuum beschreibt. Die Gesamtheit aller möglichen Zustandsübergänge zwischen den Aktivitäten eines Prozesses und deren zeitlich oder logisch bedingten Abhängigkeiten werden im Kontext prozessorientierter Anwendungen als *Kontrollfluss* bezeichnet (vgl. [LR00, All05, Sta06, Gad10]).

Unabhängig von einer graphischen oder textuellen Darstellungsform können Kontrollflusselemente von Prozessbeschreibungen entweder in einer Graphstruktur oder in einer Blockstruktur aufgebaut sein [MLZ06, FKT08, Hün08]. Auf einer *Graphstruktur* basierende Strukturen definieren den Kontrollfluss über *Transitionen* (auch *Kontrollflusskonnektoren* [LR00]), welche verbindende Übergänge zwischen Aktivitäten darstellen, um eine potentielle Reihenfolge für deren Ausführung festzulegen. Dabei stellen die Transitionen die gerichteten Kanten und die Aktivitäten die Knoten des Graphen dar. Aktivitäten können durchaus mehrere *eingehende* und *ausgehende* Transitionen besitzen, welche auf diese Art und Weise spezifizieren, ob Aufgaben sequentiell oder parallel ausgeführt werden sollen. Besitzt eine Aktivität keine eingehenden Transitionen, so ist sie als Startaktivität definiert. Aktivitäten ohne ausgehende Transitionen signalisieren das Ende eines Prozesses [LR00]. Technische Prozessbeschreibungen in Graphstruktur erlauben eine relativ direkte Abbildung von den auf strategischen und operativen Ebenen entwickelten Prozessmodellen, welche zum Großteil bereits in einer flussbasierten graphischen Notation vorliegen (vgl. Abschnitt 2.3).

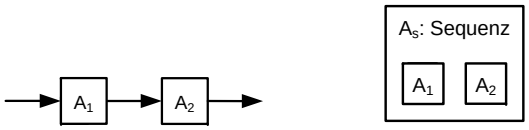
In einer *Blockstruktur* hingegen werden Konstrukte der Prozessbeschreibung geschachtelt und spezielle strukturierte Elemente stehen als Behälter zur Verfügung, um eine Ordnung zwischen den enthaltenen Aktivitäten herzustellen (*Hierarchisierung*) [Hün08, Ses10]. Hierzu wird zwischen *atomaren* Aktivitäten und *strukturierten* Aktivitäten unterschieden, wobei die strukturierte Aktivität einen Block beinhaltet, welcher den Ablauf der untergeordneten Elemente definiert (vgl. [LR00, MLZ06, FKT08]). Auf der obersten Ebene gibt es dabei genau einen Block, mit dem die hierarchische Struktur beginnt [Hün08]. Aufgrund der nicht intuitiven Darstellung für Benutzergruppen ohne Programmierkenntnisse werden Prozessbeschreibungen in reiner Blockstruktur auf den strategischen und operativen Ebenen der Prozessmodellierung in der Regel nicht verwendet. Es gibt jedoch Prozessbeschreibungen auf technischer Ebene, welche die Blockstruktur aufgrund ihrer syntaktischen Ähnlichkeit zu imperativen Programmiersprachen und damit zu der Ausführung von Maschinen-Code nutzen [Tha01, And03]. Zudem existieren eine ganze Reihe von Prozessbeschreibungssprachen, welche Kontrollflussstrukturen in Graph- und Blockstruktur miteinander kombinieren, um die Vorteile beider Strukturvarianten nutzbar zu machen [NMS05, OAS07].

Unabhängig von einer konkreten Repräsentation können grundsätzliche Entwicklungsmuster für Kontrollflusskonstrukte und -strukturen identifiziert werden, welche auf Basis einer Zusammenstellung von *Kontrollflussmustern* der Autoren VAN DER AALST et al. und Erweiterungen anderer Autoren im Folgenden beschrieben werden (vgl. [vdAvH02, vdATHKB03, RHM06]). Zu deren Visualisierung wird in dieser Arbeit eine einfache graphische Notation verwendet, wobei die grundlegenden Kontrollflussstrukturen zur Veranschaulichung sowohl in Graphstruktur als auch in Blockstruktur gezeigt werden, sofern beide Repräsentationen möglich bzw. üblich sind. Die Abbildungen dienen

dabei ausschließlich zur Veranschaulichung der allgemeinen Kontrollflussmuster und stellen keine vollständige formale Prozessbeschreibungssprache dar.

2.4.1 Grundlegende Kontrollflussstrukturen

Der einfachste Anwendungsfall für den Ablauf eines Prozesses besteht darin, dass eine unbedingte *sequentielle Ausführungsreihenfolge* zwischen zwei Aktivitäten vorliegt [vdAvH02, vdATHKB03]. Sind zwei Aktivitäten A_1 und A_2 durch eine Transition miteinander verbunden oder einer sequentiellen Blockstruktur untergeordnet, so kann Aktivität A_2 in der Regel erst ausgeführt werden, wenn Aktivität A_1 beendet worden ist (vgl. Abbildung 2.4).

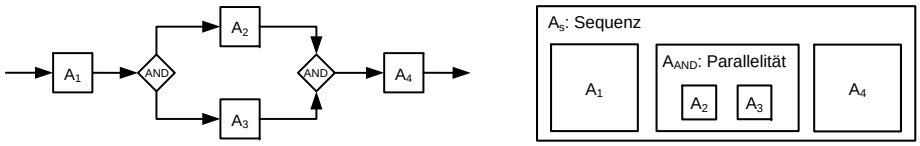


(a) Graphstruktur **(b)** Blockstruktur

Abbildung 2.4: Einfache Sequenz der Aktivitäten A_1 und A_2

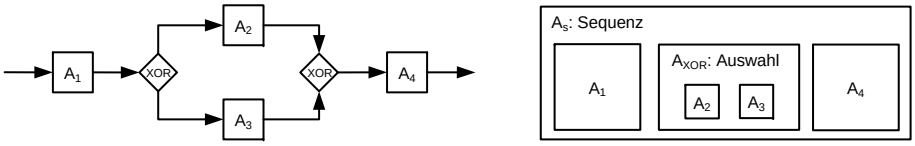
Voneinander unabhängige Aktivitäten können prinzipiell in beliebiger Reihenfolge bearbeitet und zur Beschleunigung der Prozessausführung parallel zueinander ausgeführt werden. Eine parallele Ausführung erfordert die Aufteilung des Prozesses in eine Menge voneinander unabhängiger *Kontrollflusspfade*. Die Struktur, die eine solche Aufteilung in parallele Pfade einleitet, wird als *AND-Split* bezeichnet [vdAvH02]. Parallele Pfade eines Prozesses können unabhängig voneinander terminieren oder an späterer Stelle des Prozesses wieder zusammengeführt werden. In diesem Fall wird an der zusammenführenden Struktur (*AND-Join*) der Kontroll- und Datenfluss der parallelen Pfade synchronisiert. Die dem AND-Join nachgelagerte Aktivität kann im Normalfall erst gestartet werden, wenn der Kontrollfluss innerhalb aller parallelen Pfade den Synchronisationspunkt erreicht hat. Daher muss an dieser Stelle ggf. auf das Fertigstellen paralleler Aktivitäten gewartet werden [vdATHKB03]. Im Fall eines blockstrukturierten Kontrollflusses ist die Art der Weiterführung des Prozesses durch die entsprechend übergeordnete Struktur geregelt. Eine explizite Modellierung der Synchronisation ist hier nicht notwendig, da diese durch die Verwendung der Blockstruktur erreicht wird (vgl. Abbildung 2.5).

In vielen Fällen ist es wichtig, dass Entscheidungen zwischen verschiedenen Varianten der Prozessausführung getroffen werden können. Mögliche Varianten des Kontrollflusses können durch Verzweigungen dargestellt werden, bei denen alle möglichen Kontrollflusspfade angegeben werden, aber zur Ausführungszeit des Prozesses nur ein Pfad ausgewählt und weiter verfolgt wird. Eine solche exklusive Verzweigung in alternative Kontrollflusspfade wird als *XOR-Split* bezeichnet [vdAvH02]. Wie bei der parallelen Ausführung können auch alternative Pfade unabhängig voneinander fortgeführt werden.



(a) Graphstruktur **(b)** Blockstruktur

Abbildung 2.5: Parallele Ausführung von A_2 und A_3 mit anschließender Synchronisation



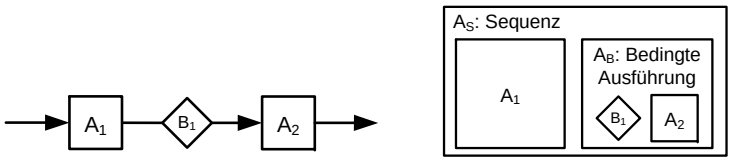
(a) Graphstruktur **(b)** Blockstruktur

Abbildung 2.6: Alternative Ausführung von A_2 und A_3 mit anschließender unsynchronisierter Zusammenführung des Kontrollflusses

Im Fall eines vorangegangenen XOR-Splits ist jedoch auch bei der Zusammenführung der möglichen Kontrollflusspfade in der Regel keine Synchronisation notwendig, da während der Ausführung nur ein einziger Kontrollflusspfad gewählt werden kann. Es ist daher kein Warten erforderlich und der Prozess kann ohne Verzögerung an dieser Stelle fortgeführt werden, sobald die Ausführung des ausgewählten Pfades beendet ist. Man bezeichnet die zusammenführende Kontrollflussstruktur daher als *XOR-Join* oder einfach nur als *Zusammenführung* (*Merge*) [vdATHKB03]. Abbildung 2.6 zeigt die entsprechenden Konstrukte für Graph- und Blockstrukturen.

Um Entscheidungen zur Wahl über Kontrollflusspfade zu treffen, werden geeignete *Bedingungskonstrukte* benötigt. Diese beschreiben die Umstände, unter denen der Übergang von einer Aktivität zur nächsten erlaubt sein soll. In technischen Prozessmodellen sind dies in der Regel Ausdrücke der boolschen Algebra. Sind die spezifizierten Umstände unzutreffend, wird die Bedingung zu *false* ausgewertet und die nachstehende Aktivität wird nicht ausgeführt. Zunächst können einfache sequentielle Ausführungen von einer *Transitionsbedingung* abhängig gemacht werden (vgl. Abbildung 2.7). Als Verfeinerung dieses Konstrukts existieren *Aktivierungsbedingungen*, die anhand bestimmter Vorgaben evaluieren, ob eine Aktivität gestartet werden kann, z. B. eine Zeitvorgabe [LR00]. Die sogenannte *Exit-Bedingung* kann als Nachbedingung der Aktivität angesehen werden. Sie spezifiziert, unter welchen Umständen eine Aktivität nach dessen Ausführung als erledigt anzusehen ist. Zum Beispiel kann auf diese Weise geprüft werden, ob alle Ergebnisdaten in vorgegebener Art und Weise vorliegen [LR00]. Die oben explizit aufgeführten Verzweigungen des Prozesses können ebenfalls durch solche einfachen Bedingungen modelliert werden. Um eine bessere maschinelle Ausführbarkeit zu gewährleisten, bietet es sich hierbei jedoch an, explizite Konstrukte zu verwen-

den [vdATHKB03]. Eine *Split-Bedingung* legt dazu fest, welcher der folgenden alternativen Pfade ausgeführt werden soll. Eine *Join-Bedingung* fungiert an einer eingehenden Transition als Synchronisationspunkt parallel ausgeführter Aktivitäten und überwacht, ob alle spezifizierten Aufgaben so erfüllt worden sind, dass die nächste Aktivität gestartet werden kann. In Blockstruktur wird zur Abbildung von Bedingungen in der Regel das aus vielen Programmiersprachen bekannte *IF...THEN*- bzw. *IF...THEN...ELSE*-Konstrukt verwendet.



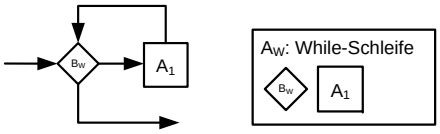
(a) Graphstruktur **(b)** Blockstruktur

Abbildung 2.7: Bedingte Ausführung von A_2

2.4.2 Erweiterte Kontrollflussstrukturen

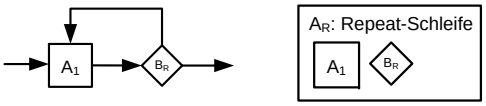
Die vorgenannten Bedingungskonstrukte können in beliebiger Art kombiniert und zu komplexeren Formen zusammengesetzt werden, so dass im Prinzip alle möglichen Wahrheitswertkombinationen abgefragt werden können. Zur Vereinfachung gibt es in der Regel erweiterte Konstrukte, welche die Umsetzung komplexer Kontrollflussstrukturen unterstützen. Hierzu gehören die *Mehrfachauswahl (OR-Split)* mit entsprechenden Möglichkeiten zur Synchronisation oder zur Zusammenführung (*OR-Join*, *Multi-Merge*, *Diskriminator*, *N-out-of-M-Join*) [vdABHK00, vdATHKB03], Konstrukte zur expliziten Beendigung des Prozesses sowie *strukturierte* und *unstrukturierte Schleifen*.

Während der *AND-Join* als Zusammenführung parallel ausgeführter Pfade in jedem Fall eine Synchronisation des Kontrollflusses notwendig macht (vgl. Abschnitt 2.4.1), besitzen die erweiterten Join-Strukturen eine teilweise andere Semantik. Der *OR-Join* macht eine Synchronisation genau derjenigen Pfade erforderlich, welche zuvor an einem *OR-Split* anhand von Bedingungskonstrukten gestartet wurden. Hierzu ist am *OR-Join* entsprechendes Wissen darüber notwendig, wieviele Pfade gestartet wurden, damit für die Synchronisation auf genau diese Anzahl an eingehenden Kontrollflusspfaden gewartet werden kann. Beim *Multi-Merge* ist dies nicht der Fall, da hierbei die dem Join folgenden Aktivitäten für jeden der eingehenden Pfade ausgeführt werden. Es handelt sich hierbei also sozusagen um eine vereinfachte Schreibweise für die parallele Ausführung identischer Folgeaktivitäten (vgl. [vdATHKB03]). Ein *Diskriminator* verhält sich im Wesentlichen wie ein *XOR-Join*, welcher nach Eintreffen des Kontrollflusses des ersten Pfades die Folgeaktivitäten startet und das Eintreffen aller weiteren Pfade ignoriert. Der *N-out-of-M-Join* als allgemeiner Fall des *Diskriminators* fordert die Ausführung der Folgeaktivitäten nach dem Eintreffen einer bestimmten Anzahl von eingehenden Pfaden



(a) Graphstruktur **(b)** Blockstruktur

Abbildung 2.8: Iterative Ausführung von A_1: WHILE...DO...-Schleife



(a) Graphstruktur **(b)** Blockstruktur

Abbildung 2.9: Iterative Ausführung von A_1: REPEAT...UNTIL...-Schleife

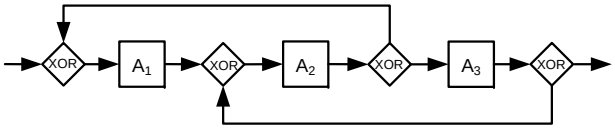


Abbildung 2.10: Unstrukturierte verschachtelte Schleife

[vdATHKB03]. Sogenannte *zustandsbasierte Kontrollflussstrukturen* beschreiben schließlich, dass eine Fortführung eines bestimmten Prozesspfades nur erfolgen soll, falls ein *Meilenstein* auf einem anderen Pfad erreicht wurde bzw. eine *Zeitbeschränkung* noch nicht überschritten wurde [vdATHKB03]. Es ist an dieser Stelle also ggf. eine zusätzliche implizite Synchronisation zwischen den Prozesspfaden notwendig.

Strukturierte Schleifen besitzen nur jeweils eine Eintritts- und eine Austrittsmöglichkeit, welche jeweils mit Bedingungen versehen werden können. Wie in imperativen Programmiersprachen unterscheidet man zwischen kopfgesteuerten bzw. vorprüfenden Schleifen, bei denen die Abbruchbedingung geprüft wird, bevor der Schleifenrumpf durchlaufen wird (*WHILE...DO...*-Schleife) und fußgesteuerten bzw. nachprüfenden Schleifen, bei denen die Abbruchbedingung nach dem Durchlauf des Schleifenrumpfes überprüft wird (*DO...WHILE...*- oder *REPEAT...UNTIL...*-Schleife) [RHM06]. Die Abbildungen 2.8 und 2.9 verdeutlichen die beiden Varianten. Unstrukturierte Schleifen sind vergleichbar mit dem *GOTO*-Konzept und können auf Basis einer Graphstruktur einen beliebigen iterativen sequentiellen Kontrollfluss abbilden. Bei einer Blockstruktur führt jedoch die strikte Trennung in Blöcke dazu, dass Prozessschritte in einem Block keine direkten Auswirkungen auf Prozessschritte in anderen Blöcken (also über Blockgrenzen hinweg) haben können [Hün08]. Daher sind verschachtelte unstrukturierte Schleifen wie in Abbildung 2.10 in Blockstruktur nicht ohne weitere Hilfsmittel abbildbar [Sch07b].

Neben der mehrfachen sequentiellen Ausführung einer Aktivität innerhalb einer Schleife kann es notwendig sein, multiple Instanzen einer Aktivität parallel auszuführen [vdABHK00, vdATHKB03]. Sind diese Notwendigkeit und die Anzahl der auszuführenden Instanzen zur Entwicklungszeit des Prozesses bekannt, ist es möglich, die betreffende Aktivität mehrfach auf jeweils parallelen Pfaden zu modellieren. Ist die Anzahl der Instanzen jedoch erst zur Laufzeit des Prozesses bekannt, sind erweiterte Kontrollflussstrukturen notwendig, um die Aktivität für jedes benötigte Element (z. B. einer Auflistung) durchzuführen (*FOR-EACH*-Schleife). Zudem ist eine Synchronisation multipler Instanzen und das Fortführen des Kontrollflusses notwendig, nachdem alle Instanzen einer Aktivität beendet sind [vdATHKB03].

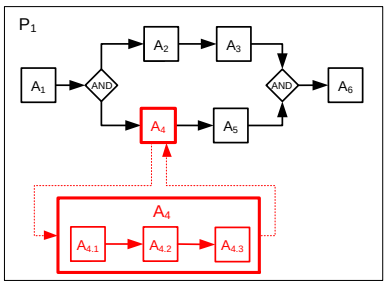
Zur Beschleunigung der Prozessausführung oder zur detaillierten Steuerung kann zudem von der oben definierten sequentiellen Ausführungsreihenfolge, bei welcher der Beginn der zweiten Aktivität zeitlich nach dem Ende der ersten Aktivität liegen muss, abgewichen werden. Im Rahmen einer sequentiellen Ausführung kann ebenso gefordert sein, dass

- die zweite Aktivität nach dem Beginn der ersten Aktivität (unabhängig von deren Beendigung) starten darf (*before-start*),
- unabhängig von dem Start der Aktivitäten nur das Ende der zweiten Aktivität vor dem Ende der ersten Aktivität eintreten muss (*after-end*),
- die zweite Aktivität erst nach dem Start der ersten Aktivität enden darf (*before-end*)
- oder der Start der zweiten Aktivität exakt zum Zeitpunkt der Beendigung der ersten Aktivität erfolgen muss (*meets*) [vdATHKB03].

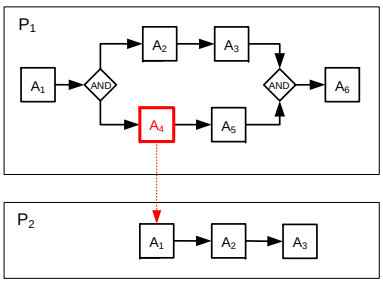
Als spezielle Lockerung der sequentiellen Ausführung kann zudem erlaubt werden, dass die spezifizierten Aktivitäten in beliebiger Reihenfolge, aber im Gegensatz zur parallelen Ausführung nicht gleichzeitig ausgeführt werden dürfen (*Critical Section*) [RHM06]. Zur Vereinfachung soll im weiteren Verlauf dieser Arbeit zunächst die übliche und praktisch relevante Ausführungssemantik vorausgesetzt werden, bei welcher bei einer Sequenz von zwei Aktivitäten stets der Beginn der zweiten Aktivität zeitlich nach dem Ende der ersten Aktivität liegen muss.

2.4.3 Blockaktivitäten und Subprozesse

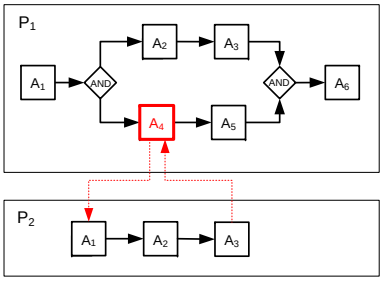
Die Wiederverwendbarkeit von häufig benötigten Abläufen innerhalb eines Anwendungsfalls und die Strukturierung eines Anwendungsfalls in mehrere Komponenten kann durch die Spezifikation von zusätzlichen hierarchischen Strukturen unterstützt werden. Wenn eine aufgerufene Aktivität eine Komposition aus anderen Aktivitäten benutzt, um ihre fachliche Aufgabe zu erfüllen und die Komposition nur innerhalb eines Prozesses sichtbar ist, spricht man von einer *Blockaktivität*. Da die Blockaktivität nur einen Platzhalter für einen



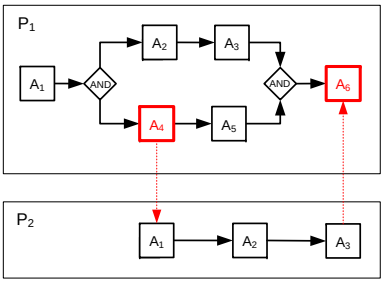
(a) Blockaktivität A_4 als Platzhalter für den Kontrollfluss zwischen $A_{4.1}$, $A_{4.2}$ und $A_{4.3}$



(b) Verkettung des übergeordneten Prozesses P_1 und des Subprozesses P_2



(c) Synchrone Ausführung des übergeordneten Prozesses P_1 und des Subprozesses P_2

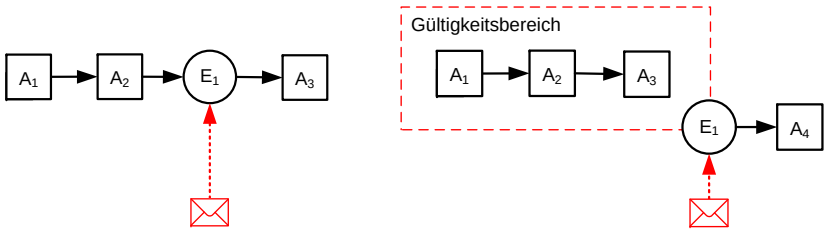


(d) Asynchrone Ausführung des übergeordneten Prozesses P_1 und des Subprozesses P_2

Abbildung 2.11: Hierarchische Kontrollflussstrukturen

komplexeren Kontrollfluss darstellt, muss mit der Ausführung der nachfolgenden Aktivitäten der oberen Ebene stets bis zur Beendigung der Blockaktivität gewartet werden. Im Fall eines ansonsten graphstrukturierten Prozessmodells kommt es hierbei zu einer Vermischung von Graph- und Blockstrukturen. Abbildung 2.11a zeigt ein entsprechendes Beispiel für einen Prozessgraphen mit einer Blockaktivität A_4, welche die sequentielle Ausführung der Aktivitäten $A_{4.1}$, $A_{4.2}$ und $A_{4.3}$ beinhaltet.

Wenn bereits bestehende autonome Prozesse in einem übergeordneten Prozess wiederverwendet werden, stellt der untergeordnete Prozess einen *Subprozess* dar. Während die Blockaktivität in der Regel streng in den umgebenen Prozess eingebettet ist, kann der Grad an Unabhängigkeit zwischen aufrufendem Prozess und Subprozess variieren. LEYMANN und ROLLER [LR00] differenzieren zwischen zwei Modellen von Subprozessen in Form einer einfachen *Verkettung* (*Connected Discrete Model*) und einer *hierarchischen Ausführung* (*Hierarchical Model*). Bei verketteten Subprozessen wird der Subprozess vollkommen autonom gestartet, während der Ursprungsprozess unverändert und ohne Rücksicht auf die Ergebnisse seines Subprozesses fortgeführt wird. Im Extremfall sind die beiden Prozesse einfach nur miteinander verknüpft, indem das Ergebnis von Prozess P_1 die Ausführung von Prozess P_2 anstößt. Subprozess und aufrufender Prozess stellen dadurch im eigentlichen Sinn zwei gleichwertige, verbundene Prozesse dar, welche unabhängig voneinander terminieren können (vgl. Abbildung 2.11b). Im Fall einer hierarchischen



(a) Lokal definiertes Ereignis E_1 **(b)** Global definiertes Ereignis E_1

Abbildung 2.12: Ereignisse

Ausführung liefert der Subprozess ein Ergebnis an die aufrufende Aktivität zurück. Der Ursprungsprozess wird dadurch entweder solange in seiner Ausführung blockiert, bis der Subprozess beendet ist und das Ergebnis seiner Ausführung übergibt (*synchroner Subprozess*, vgl. Abbildung 2.11c), oder beide Prozesse laufen autonom voneinander weiter und werden zu einem späteren Zeitpunkt synchronisiert (*asynchroner Subprozess*, vgl. Abbildung 2.11d). Bei beiden Fällen handelt es sich um eine hierarchische Anordnung von Prozessen, die eine starke Abhängigkeit der Prozesse untereinander beschreibt [LR00].

2.4.4 Ereignisse und externe Anweisungen

Um eine gewisses Maß an Flexibilität während der Ausführung eines Prozesses zu erreichen, soll in vielen Fällen das Eintreten spezifizierter Ereignisse außerhalb des definierten Kontrollflusses berücksichtigt werden [Puh06]. Eine bestimmte Art von Bedingungskonstrukten stellen *lokal definierte Ereignisse* dar, welche direkt in den Kontrollfluss eingebunden sind. Lokal definierte Ereignisse bedingen die Ausführung einzelner Aktivitäten, indem sie die Ausführung der nachfolgenden Aktivität(en) verzögern, solange das Ereignis ausbleibt (z. B. der erwartete Eingang einer Nachricht zur Bestätigung einer Bestellung). Ein solcher Fall ist in Abbildung 2.12a dargestellt. Die Ereignisse können hierbei persistent oder flüchtig sein [RHM06]. *Global definierte Ereignisse* sind vom Kontrollfluss einzelner Aktivitäten losgelöst und einem speziellen *Gültigkeitsbereich* (*Scope*) zugeordnet, welcher im Extremfall auch den ganzen Prozess umfassen kann. Global definierte Ereignisse lösen die Ausführung der als Ereignisbehandlung angegebenen Aktivität bzw. Aktivitäten aus, sobald das vorgegebene Ereignis eintritt (vgl. Aktivität A_4 in Abbildung 2.12b). Die Ausführung der ursprünglichen Aktivitäten innerhalb des definierten Gültigkeitsbereichs kann dabei durch Parallelausführung der Ereignisbehandlung ergänzt, temporär unterbrochen oder gänzlich durch die Ereignisbehandlung ersetzt, d. h. abgebrochen werden. Die Spezifikation von globalen Ereignissen und auszuführenden Reaktionen stellt eine enorme Vereinfachung des Prozessmodells dar, da ansonsten die den Eintritt des Ereignisses spezifizierende Bedingung vor der Ausführung jeder Aktivität überprüft werden müsste. Die Beschreibung der hierzu benötigten Gültigkeitsbereiche kann wiederum mit Blockstrukturen umgesetzt werden (vgl. Abschnitt 2.4.3).

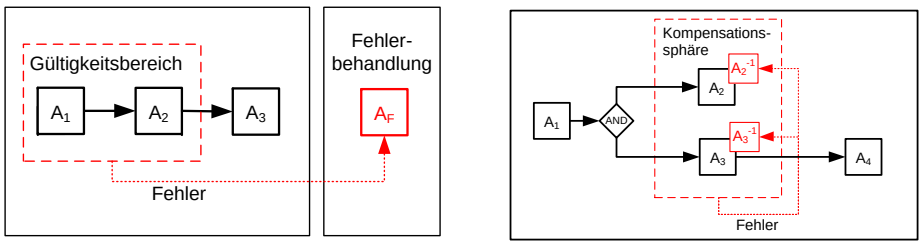
Als Reaktion auf solche Ereignisse kann neben der Ausführung einer Folge von Aktivitäten [RHM06] auch die Ausführung des Prozesses selbst beeinflusst werden, indem eine Abbruchanweisung spezifiziert wird. VAN DER AALST et al. nennen in den von ihnen aufgeführten Kontrollflussmustern zwei Arten von Abbruchanweisungen, welche es erlauben, einen bestimmten Pfad (*Cancel Activity*) oder den ganzen Prozess (*Cancel Case*) zu beenden [vdATHKB03].

2.4.5 Fehlerbehandlung

Eine weitere wichtige Ergänzung zu den von den Autoren VAN DER AALST et al. genannten Kontrollflussmustern stellen Konstrukte zur expliziten Fehlerbehandlung dar [RvdAA06, LCO+10]. Als besondere Art von Ereignissen kann zwischen fachlichen und technischen Fehlern unterschieden werden. *Fachliche Fehler* stellen die erwarteten Varianten des Anwendungsfalls dar, welche vom Regelfall (bzw. den Regelfällen) der Ausführung abweichen. Dies umfasst zum Beispiel die Verletzung von Bedingungen, den Ablauf von Fristen oder die falsche bzw. ausbleibende Ausführung einer Aktivität auf Anwendungsebene [RvdAA06]. Im Gegensatz dazu handelt es sich bei *technischen Fehlern* um infrastrukturelle oder organisatorische Probleme bei der Prozessausführung, welche Verklemmungen, undefinierte Wartezustände, die unplanmäßige Beendigung des Prozesses oder inkorrekte Ergebnisse zur Folge haben können. Beispiele für solche Fehler sind das Ausbleiben von Nachrichten oder Ereignissen bzw. die Überlastung oder Nicht-Verfügbarkeit von Ressourcen [RvdAA06].

Bei der Behandlung beider Arten von Fehlern erweist sich die ausschließliche Verwendung von Graphstrukturen als unvorteilhaft. Zwar können Fehlerbehandlungsmaßnahmen als eine Spezialform der bedingten Verzweigung dargestellt werden, welche explizit einen als Ausnahmebehandlung deklarierten Pfad einleitet. Um die Wiederverwendbarkeit der Fehlerbehandlungsmaßnahmen zu erlauben, werden jedoch in vielen Fällen Verweise auf global definierte Blöcke von Aktivitäten angegeben (vgl. [LCO+10]) und anhand von zusätzlichen *Gültigkeitsbereichen* Fehlerbehandlungsmaßnahmen einer Menge von untergeordneten Kontrollflussstrukturen zugeordnet. So können zum Beispiel bei einem Bestellvorgang für Waren sowohl das Ausbleiben der Lieferung als auch die Lieferung defekter Ware zum Widerruf des Kaufvertrags als wiederverwendbare semantische Fehlerbehandlung führen. Dieses Konstrukt ist vergleichbar mit dem *TRY...CATCH*-Prinzip aus höheren Programmiersprachen (vgl. Abbildung 2.13a).

Eine weitere Maßnahme zur Behandlung von zumeist technischen Fehlern stellen Möglichkeiten zur *Backward-Recovery* dar, um die (Teil-)Ergebnisse von Aktivitäten zurückzusetzen (*Rollback*) oder deren Effekte semantisch zu kompensieren (*Compensation*). Derartige Maßnahmen werden immer dann benötigt, wenn die bereits erfolgte Ausführung einer Aktivität aufgrund des Scheiterns einer anderen Aktivität (ggf. semantisch) rückgängig gemacht werden muss, um *transaktionales Verhalten* zwischen den betreffenden Aktivitäten zu gewährleisten [LR00]. Eine Transaktion stellt allgemein eine logi-



(a) Fehlerbehandlungsmaßnahme A_F **(b)** Kompensationssphäre um A_2 und A_3

Abbildung 2.13: Fehlerbehandlung

sche Einheit von Operationen dar, die entweder vollständig ausgeführt werden muss oder keine Auswirkung haben darf [GR92]. Die Eigenschaften klassischer *ACID-Transaktionen* (d. h. *Atomarität*, *Konsistenz*, *Isolation* und *Dauerhaftigkeit* [GR92]) sind in vielen Fällen bei lang andauernden Prozessen nicht vollständig durchsetzbar. Man spricht daher in diesem Bereich von sogenannten *Long-Running-Transactions (LRTs)*, für die erweiterte Transaktionsmodelle zur Verfügung stehen (vgl. hierzu [Ley97, LR00]).

Um transaktionales Verhalten modellieren zu können, müssen in der Regel sowohl die Schritte einer einzelnen Aktivität als auch die im Sinne einer Transaktion zusammenhängenden Aktivitäten als solche gekennzeichnet werden. Zudem muss angegeben werden, wie sich die Prozessausführung im Fall eines Fehlers verhalten soll. Handelt es sich nicht um eine Ausführung der Aktivitäten als klassische ACID-Transaktion, sondern soll das Konzept der *Kompensation* von Aktivitäten angewendet werden, so muss ebenfalls spezifiziert werden, welche Aktivitäten ausgeführt werden müssen, um die Effekte der zu kompensierenden Aktivität semantisch rückgängig zu machen. Ein Beispiel hierfür ist die Verwendung von *Kompensationssphären* [LR00], bei denen die Transaktion als bestimmter Bereich innerhalb des Prozesses markiert und der Bereich mit einer speziellen Fehlerbehandlungsmaßnahme versehen wird, welche Aktivitäten zur Kompensation dieses Bereichs enthält. Abbildung 2.13b zeigt eine Kompensationssphäre, welche die parallelen Aktivitäten A_2 und A_3 enthält, wobei diese beim Scheitern der Ausführung durch die Kompensationsaktivitäten A_2^{-1} und A_3^{-1} semantisch rückgängig gemacht werden können.

Nicht immer ist die Kompensation von Aktivitäten für die effiziente Ausführung von Prozessen vorteilhaft, da hierbei auf bereits erfolgreich verrichtete Arbeit verzichtet werden muss [ACKM04]. Eine Alternative für die Behandlung von technischen Fehlern bietet die Durchführung von *Forward-Recovery*-Maßnahmen (*Retry*), durch die unter bestimmten Bedingungen ein erneuter Versuch für die Ausführung einer gescheiterten Aktivität vorgenommen werden kann [ACKM04, LCO+10].

2.5 Datenfluss von Prozessen

Zumindest für alle Prozessmodelle auf technischer Ebene ist neben der genauen Spezifikation des Kontrollflusses auch die Festlegung des für die korrekte Bearbeitung der Aktivitäten notwendigen *Datenflusses* notwendig. Daten können grundsätzlich verschiedene Sichtbarkeiten aufweisen, d. h. innerhalb einer Aktivität, innerhalb eines Bereichs von Aktivitäten, innerhalb eines Prozesses, zwischen verschiedenen Prozessen oder zwischen Prozessen und externen Umgebungen verwendet werden [RHE04, RtHEvdA05]. Dabei ist die Menge der während der Prozessausführung benötigten Daten (*Prozessdaten*), welche zur Verarbeitung einzelner Aktivitäten benötigt, von diesen als Ergebnis erzeugt oder zur Evaluation einer Transitionsbedingung verwendet werden, von den *System-* und *Umgebungsdaten* abzugrenzen, welche von der Ausführungsumgebung zur technischen Verwaltung der Prozesse benötigt werden und nur bedingt Einfluss auf den Ablauf eines Prozesses haben [NM02].

Der Datenfluss innerhalb eines Prozesses wird als *Intraprozesskommunikation* bezeichnet. Werden zur Bearbeitung von Aktivitäten innerhalb eines Prozesses Daten oder Dokumente benötigt, wird der Datenfluss implizit über die erwarteten Eingabeparameter der Aktivitäten und deren Ausgabeparameter definiert. Daten, die für die Ausführung einer Aufgabe relevant sind und der Aktivität zugeführt werden müssen, werden daher als *Input-Daten* einer Aktivität bezeichnet [LR00]. Sie werden in einem *Input-Container* der jeweiligen Aktivität gespeichert und bei Bedarf verarbeitet. Dabei werden Daten mit der Umgebung ausgetauscht, z. B. um eine Anwendung aufzurufen, welche die Daten in einer Berechnung verwendet und ggf. Ergebnisse an die aufrufende Aktivität zurückliefert. Stellen die Daten ein Ergebnis einer Aktivität oder gar des Gesamtprozesses dar oder sollen sie von folgenden Aktivitäten weiterbearbeitet werden, so stellen die Daten sogenannte *Output-Daten* der Aktivität bzw. des Prozesses dar. Sie werden im *Output-Container* der Aktivität gespeichert, bis die Aktivität beendet ist oder die Daten von einer anderen Aktivität bezogen werden [LR00]. Abbildung 2.14 stellt diesen grundlegenden Zusammenhang graphisch dar.

Im einfachsten Fall ist die Kontrollstruktur des Datenflusses mit der des Kontrollflusses identisch. Es kann jedoch Vorteile haben, den Datenfluss vom Kontrollfluss zu trennen, insbesondere dann, wenn bestimmte Daten nicht von allen Aktivitäten bearbeitet werden sollen. Dies erhöht die Flexibilität der Ausführung, falls Daten erst an späterer Stelle des Prozesses wieder benötigt werden oder die Mitführung von komplexen Dokumenten über einen langen Zeitraum nicht realisierbar ist. Der Datenfluss bekommt in diesem Fall eigene, von der Kontrollstruktur weitestgehend unabhängige *Datenflusskonnektoren*, welche die Zuteilung von Daten und Dokumenten an Aktivitäten des Prozesses beschreiben [LR00].

Zur Unterstützung von Ein- und Ausgabedaten, die erst zur Laufzeit des Prozesses bekannt werden, werden *Prozessvariablen* bzw. *Datenfelder* verwendet. Hierbei handelt es sich um Behälter im Sinne einer Variablen, welche

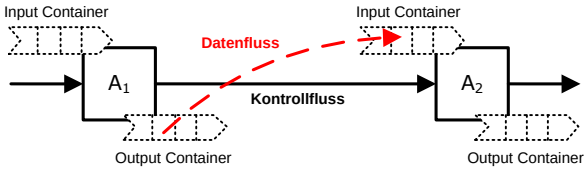
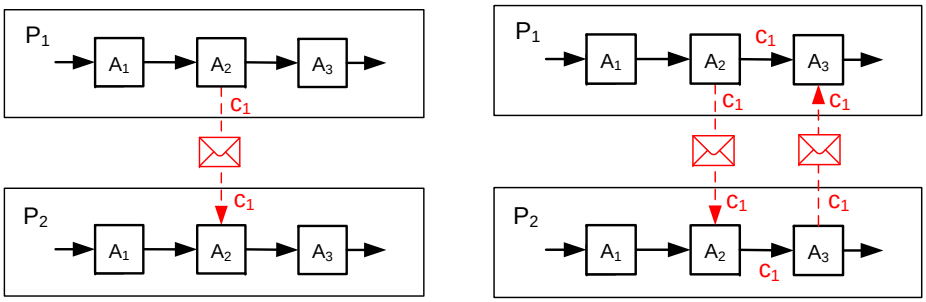


Abbildung 2.14: Intraprozesskommunikation: Datenfluss zwischen zwei Aktivitäten A_1 und A_2 (LR00)

einen Namen und ggf. weitere datenbezogene Informationen (wie z. B. einen Datentyp) enthalten können. Neben der Weitergabe von Daten zwischen atomaren Aktivitäten, ist ein Datenfluss auch auf höherer Ebene möglich und notwendig, um bei der Instantiierung des Prozesses diese Datenfelder zu initialisieren und damit konkrete Werte für die Prozessausführung festzulegen. Im Fall von Blockaktivitäten oder Subprozessen müssen die dort benötigten Daten weitergereicht und nach deren Beendigung ggf. an die übergeordnete Kontrollflussstruktur zurückgegeben werden. Nach Beendigung des gesamten Prozesses können schließlich etwaige Rückgabewerte an den Initiator des Prozesses ausgegeben werden [RHE04, RtHEvdA05].

Daten können jedoch nicht nur zwischen den Kontrollflusskonstrukten eines Prozesses ausgetauscht werden, sondern können auch der Interaktion mit anderen Prozessen dienen. In diesem Fall spricht man von *Interprozesskommunikation*. Die Kommunikation mit anderen Prozessen oder Subprozessen steht in der Regel in engem Zusammenhang mit deren Kontrollfluss. VAN DER AALST et al. [vdABHK00, vdATHKB03] unterscheiden dabei zwischen dem einfachen Senden einer Nachricht an eine andere Prozessinstanz (*Message Communication*) und der komplexeren Koordination von mehreren aufeinander folgenden Nachrichten zwischen einer Reihe von Prozessinstanzen (*Message Coordination*). In beiden Fällen ist es notwendig, die gesendeten bzw. empfangenen Nachrichten einem gemeinsamen Kontext zuzuordnen, um sicherzustellen, dass die eingehende Nachricht der richtigen Prozessinstanz und der richtigen Aktivität darin zugeordnet wird. Wird auf eine eingehende Nachricht eine Antwort gesendet, so muss ebenfalls gewährleistet werden, dass der Konsument der Antwort diese der Prozessinstanz zuordnen kann, welche die Quelle der vorausgegangenen Anfrage war. Für diese Zuordnung müssen geeignete Mittel der *Nachrichtenkorrelation* eingesetzt werden [WS04, Ses10].

Abbildung 2.15 zeigt das Senden und Empfangen einer Nachricht und die Koordination von mehreren Nachrichten. Dabei versendet jeweils Aktivität A_2 des Prozesses P_1 eine Nachricht an Aktivität A_2 des Prozesses P_2. Die beiden Prozessinstanzen teilen den gemeinsamen Kontext C_1, der für die Nachrichtenkorrelation als Identifikator genutzt wird. Bei der Koordination wird der Identifikator ebenfalls genutzt, um die Antwort der vorangegangenen Anfrage zuzuordnen. Dies setzt voraus, dass der Kontrollfluss von Prozess P_1 in der Zwischenzeit zu Aktivität A_3 vorangeschritten ist und hierbei ebenfalls der gemeinsame Kontext C_1 übergeben worden ist.



(a) Einfaches Senden und Empfangen einer Nachricht

(b) Koordination von Nachrichten

Abbildung 2.15: Interprozesskommunikation: Datenfluss zwischen zwei Prozessen P_1 und P_2 mit dem gemeinsamen Kontext C_1

Als erweiterte Form der Interprozesskommunikation kann eine bestimmte Nachricht als *Multicast* gleichzeitig an verschiedene externe Empfänger gesendet werden, wobei die Anzahl und Identität der Empfänger zur Entwicklungszeit oder zur Laufzeit des Prozesses festgelegt werden kann. Entsprechend können mehrere Nachrichten von verschiedenen Sendern zur gleichen Zeit empfangen werden (vgl. [vdABHK00, vdATHKB03]).

2.6 Zuordnung von Ressourcen

Ein weiterer wichtiger, aber dennoch optionaler Bestandteil einer Prozessbeschreibung ist die Verknüpfung der angegebenen Aktivitäten mit den für die Ausführung vorgesehenen Ressourcen. Ressourcen sind alle Personen, Maschinen und Softwarekomponenten, die potentiell an der Ausführung von Aktivitäten oder an der Administration des Prozesses selbst beteiligt sein können [vdAvH02]. Sie werden in der organisationalen bzw. infrastrukturellen Struktur erfasst, welche die personelle und technische Umgebung definiert, in welcher der Prozess ausgeführt werden soll [LR00, All05]. Diese Struktur wird im Folgenden allgemein als *Ausführungsumgebung* bezeichnet.

Eine Zuordnung von Ressourcen im Prozessmodell ist nicht in jedem Fall erforderlich, da eine Zuordnung auch zur Laufzeit des Prozesses durch die entsprechende Ausführungseinheit vorgenommen werden kann (vgl. Abschnitt 2.7). Oft ist jedoch für die korrekte Abbildung eines Anwendungsfalls nicht nur dessen fachliche Funktionalität wesentlich, sondern auch, von welcher Ressource oder von welcher Art von Ressource diese Funktionalität erbracht werden muss. Je nach Bearbeitungsumfang und Granularität können Ressourcen für einzelne Aktivitäten, für eine Menge von Aktivitäten oder für den gesamten Prozess festgelegt werden [RHE05]. In graphischen Prozessmodellen wird zur Darstellung dieser weiteren Dimension neben Kontroll- und Datenfluss eines Prozesses oft eine neue Ebene eingefügt, mit der das Prozessmodell hinterlegt wird. Diese Aktivitäten können in Bahnen, den so genannten *Swimlanes* oder *Lanes* angeordnet werden, um organisatorische Zuständigkeiten

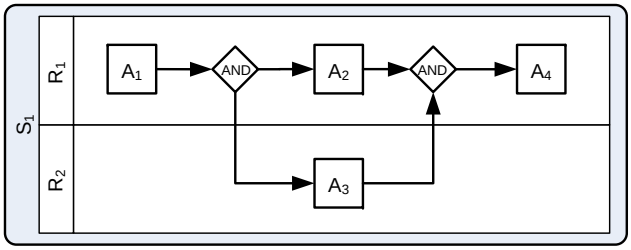


Abbildung 2.16: Zuweisung organisatorischer Zuständigkeiten an die Ressourcen R_1 und R_2 für die Ausführung von Aktivitäten sowie an das System S_1 für die Steuerung des Prozesses

im erzeugten Modell abzubilden [JBo07, FRH10]. Abbildung 2.16 zeigt eine der *Business Process Modeling Notation (BPMN)* [OMG11] entlehnte Darstellung für die Zuweisung von Ressourcen. Dabei ist die Ressource R_1 für die Bearbeitung der Aktivitäten A_1, A_2 und A_4 verantwortlich, während die Ressource R_2 die Bearbeitung der Aktivität A_3 übernehmen soll, damit diese auch tatsächlich parallel zu A_2 ausgeführt werden kann. Die organisatorische Zuordnung von Ressourcen zu einer Ausführungsumgebung wird durch eine umschließende Struktur dargestellt, welche angibt, welche Ressource für die Steuerung des Prozesses verantwortlich ist [FRH10]. Diese Struktur wird als *Pool* bezeichnet [FRH10]. In Abbildung 2.16 ist als Beispiel das System S_1 für die Steuerung des Prozesses zuständig. Die Angabe der Verantwortlichkeit für die Steuerung des Prozesses ist vor allem für die gezielte Ausführung organisationsübergreifender Prozesse relevant.

Eine Zuordnung von Ressourcen zur Entwicklungszeit kann generell einerseits durch Angabe einer fixen Instanz oder durch die Zuordnung von Instanzen zu Rollen und die Angabe einer Rolle in der Prozessbeschreibung erfolgen [RHE05, Wes07]. *Rollen* beschreiben Ressourcen mit hinsichtlich eines bestimmten Kriteriums gleichen oder ähnlichen Eigenschaften. Wird die ausführende Ressource durch eine Rollenbezeichnung identifiziert, so kann flexibel zur Laufzeit festgelegt werden, welche tatsächliche Person, Maschine oder Softwarekomponente zur Ausführung herangezogen werden soll. Dabei können Rollen nicht nur auf fachlichen Kompetenzen, sondern z. B. auch auf dem Besitz von Zugriffsrechten, auf Priorisierung oder auf nicht-funktionalen Anforderungen beruhen. Komplexere Anforderungen an Zuständigkeiten werden oft aus Gründen der Sicherheit, der Robustheit oder der Leistungssteigerung vorgenommen. So kann z. B. definiert werden, dass zwei Aktivitäten nicht von derselben Ressource ausgeführt werden dürfen (*Separation-of-Duties*-Muster), die Verantwortlichkeit für nachfolgende Aktivitäten von der Verantwortlichkeit für eine vorausgegangene Aktivität abhängig ist (*Retain-Familiar*-Muster) oder die Zuweisung aufgrund von zurückliegenden positiven Erfahrungen mit Ressourcen vorgenommen wird (*History-Based-Distribution*) [RHE05]).

Schließlich gibt es bestimmte Arten von Aktivitäten, die keine Zuweisung an Ressourcen erfordern bzw. ermöglichen, da sie automatisch ausgeführt werden. Beispiele sind Wartezustände, Aktivitäten zur Zuweisung von Werten zu Pro-

zessvariablen oder Anweisungen zur Termination des Prozesses [RHE05]. Diese Aufgaben werden in der Regel von der Ausführungseinheit vorgenommen, die für die Steuerung des Prozesses zuständig ist. Bei der Ausführung von prozessorientierten Anwendungen erfolgt diese Steuerung in der Regel über ein spezielles Informations- und Kommunikationssystem. Im folgenden Abschnitt wird die resultierende computergestützte Ausführung von Prozessen näher betrachtet.

2.7 Computergestützte Ausführung von Prozessen

Die Entwicklung, Umsetzung, Durchführung und Steuerung von prozessorientierten Anwendungen kann auf vielfältige und umfassende Weise von Informations- und Kommunikationstechnologien unterstützt werden. Da die für diese Aufgaben eingesetzten Hardware- und Softwaresysteme oft in einem engeren Zusammenhang stehen, werden sie unter dem Sammelbegriff des *Prozessmanagementsystems* oder im betriebswirtschaftlichen Kontext als *Business Process Management System (BPMS)* bzw. als *Workflow Management System (WfMS)* zusammengefasst [Mül05, Str08]. Abbildung 2.17 zeigt eine idealtypische Architektur der wesentlichen Systemkomponenten (vgl. auch [GG99, Gad10] zur Rahmenarchitektur klassischer Workflow-Management-Systeme).

In den Entwicklungsphasen eines Prozessmodells stehen dem Benutzer zunächst Softwareanwendungen zur Verfügung, welche die Erstellung von strategischen, operativen oder technischen Prozessmodellen unterstützen oder den gesamten Entwicklungsprozess begleiten. Hierbei handelt es sich in der Regel um Werkzeuge zur graphischen Modellierung von Prozessmodellen, zu deren Dokumentation [Str08] und ggf. zur Transformation der auf diese Weise entwickelten Prozesse in eine ausführbare Repräsentation. Die hierfür verwendeten Systeme werden daher als Klasse der *Prozessmodellierungswerkzeuge* zusammengefasst.

Die aus der Entwicklungsphase resultierenden konkreten Prozessmodelle oder allgemeinere Prozessvorlagen werden in der Regel als einzelne *Prozessdateien* von einer zentralen Datenhaltungskomponente des Prozessmanagementsystems gespeichert bzw. verwaltet (*Process Repository*) [Str08]. Hierbei kann es sich je nach Anzahl und Entwicklungsstand der Modelle um ein einfaches Dateisystem, eine Datenbank oder ein Versionsverwaltungssystem handeln. Die fertigen technischen und direkt ausführbaren Prozessmodelle können dann zur Ausführung freigegeben werden. Dieser Schritt wird auch als *Deployment* bezeichnet und umfasst alle technischen Vorkehrungen, die zur Vorbereitung der Ausführung des Prozesses notwendig sind. Dazu zählt vor allem die Installation einer geeigneten Schnittstelle, um konkrete Parameter an den Prozess zu übergeben und ihn somit instantiieren zu können.

Die konkrete *Instantiierung* des Prozesses kann danach sowohl von Seiten eines menschlichen Benutzers als auch von Seiten einer Anwendung (verallgemeinert als *Prozessinitiator*) geschehen. Die Prozessbeschreibung wird da-

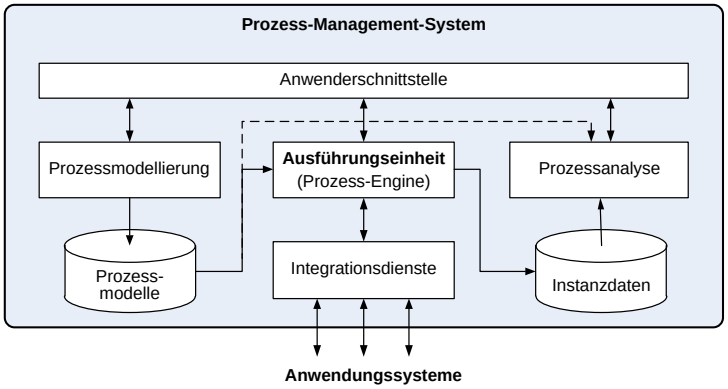


Abbildung 2.17: Idealtypische Architektur eines Prozessmanagementsystems (nach (Str08))

zu von einer auf eine bestimmte Syntax spezialisierte prozessunabhängige *Ausführungseinheit* interpretiert, welche bei einer Instantiierung auf Basis der gemachten Eingaben den Kontrollfluss des Prozesses steuert, die Daten des Prozesses verwaltet und den Aufruf von Ressourcen über verschiedene Anwendungssysteme und Anwendungen hinweg initiiert [Thu04]. Diese Ausführungseinheit wird in der Regel als *Prozess-Engine* bzw. *Workflow-Engine* bezeichnet und stellt die zentrale Komponente eines Prozessmanagementsystems dar (vgl. Abbildung 2.17). Der Aufruf von den in der Prozessbeschreibung spezifizierten Anwendungssystemen kann dabei über interne oder zusätzliche technische Integrationsdienste erfolgen, welche die technische Verbindung zu diesen Systemen herstellen und verwalten. Im Falle von heterogenen Anwendungssystemen kann dies zum Beispiel über geeignete *Adapterkomponenten* geschehen, über die auch die Interaktion mit menschlichen Akteuren oder Maschinen vorgenommen werden kann. Andernfalls kann der Zugriff von menschlichen Akteuren auch über eine zentrale Anwenderschnittstelle geschehen, welche die auszuführenden Aufgaben für den Benutzer speziell aufbereitet und dessen Eingaben entgegennimmt. Diese Art von Präsentationskomponente wird im Kontext des Workflow Managements auch als *Workflow-Client* bezeichnet [GG99, Gad10].

Zusätzlich ist es von Vorteil, die Ausführung von Prozessen aufzuzeichnen und zu analysieren. Zum einen wird unter Umständen ein Nachweis über konkrete Verantwortlichkeiten für ausgeführte Aktivitäten zur leistungsorientierten Bezahlung von Mitarbeitern oder externen Organisationen benötigt, zum anderen können problematische Aktivitäten identifiziert und die Ausführung des Gesamtprozesses in Hinblick auf Kosten und Geschwindigkeit optimiert werden [Pri03]. Aber auch aus Gesichtspunkten einer möglichen Wiederherstellung im Fehlerfall ist eine Speicherung des Prozessverlaufs sinnvoll. Da Prozesse unterschiedliche Laufzeiten von einigen wenigen Sekunden bis im Extremfall zu mehreren Jahren aufweisen können, ist es notwendig, Prozesse jederzeit unterbrechen zu können, ohne die Ergebnisse der bereits bearbeiteten Teilschritte zu verlieren und zu beliebigen späteren Zeitpunkten wieder an den aktuellen Stand der Prozessbearbeitung anknüpfen zu können. Diese

Art von Protokollierung relevanter Prozessereignisse wird als *Audit Trail* bezeichnet [LR00]. Die während der Ausführung einer Prozessinstanz anfallenden Daten werden dazu ebenfalls durch eine geeignete Komponente zur Datenhaltung gespeichert. Hierbei handelt es sich um Daten, die den Status aktuell bearbeiteter Prozessinstanzen determinieren und zur Steuerung des Prozesses dienen. Weiterhin können hier system- und anwenderbezogene Daten und Leistungskennzahlen gespeichert werden. Wichtige Daten sind zum Beispiel Datum und Zeit der Ausführung sowie Identifikatoren von Aktivitäten, Ressourcen und Ereignissen [Pri03, Str08].

Auf Basis der Prozessmodelle und der während der Ausführung von Prozessinstanzen anfallenden Daten kann eine Auswertung der Prozesse vorgenommen werden. Hierfür stehen in der Regel spezielle Anwendungssysteme zur *Prozessanalyse* zur Verfügung, welche Funktionalitäten zur *Simulation* von Prozessabläufen und zum *Monitoring* von Prozessen bereitstellen. Eine einfache Form der Überwachung umfasst eine vom Anwender ausgehende Analyse der Prozessdurchführungsdaten, zum Beispiel durch die Anzeige von Übersichten über die Art und Anzahl laufender und abgeschlossener Prozesse, durch Statusanzeigen aktuell laufender Prozesse, Auflistungen von Fehlersituation, usw. [LR00, Str08]. Komponenten zur Unterstützung des *Business Activity Monitorings (BAM)* stellen erweiterte Funktionalitäten für eine zeitnahe und weitestgehend automatisierte Prozessüberwachung dar, welche auf Basis prozessbezogener Bedingungen und Ereignisse Warnmeldungen generieren oder andere für den Anwendungsfall relevante Reaktionen auslösen können [Cha07, Str08].

In dieser Arbeit steht vor allem die Ausführung von prozessorientierten Anwendungen im Vordergrund. Die Einordnung und Funktionsweise der Ausführungsumgebung sowie die von ihr durchgeführten Aufgaben werden daher im Folgenden genauer betrachtet. Hierzu zählt vor allem die zugrunde liegende Systemarchitektur, die Navigation des Kontrollflusses zur automatischen Steuerung des Prozesses und die Einbindung von Ressourcen.

2.7.1 Systemarchitektur zur Ausführung von Prozessen

Durch den Einsatz einer Softwarekomponente als Laufzeitunterstützung für die Ausführung von Prozessen werden die Entwickler und Benutzer von prozessorientierten Anwendungen von den Aufgaben der Steuerung des Kontroll- und Datenflusses zwischen einzelnen Aktivitäten sowie von der Integration und dem Aufruf von Ressourcen befreit [DGH03]. Für den Anwender wird dabei ein neuer Abstraktionsgrad geschaffen, bei dem nur die Anwendungslogik innerhalb des (fachlichen) Prozessmodells spezifiziert werden muss und die (technische) Steuerungslogik von dem verantwortlichen Ausführungssystem gekapselt wird. Aus der Perspektive der Systeminfrastruktur stellt eine Ausführungseinheit zusammen mit ihren technischen Integrationsdiensten daher ein Informationssystem auf der Ebene der *Middleware* dar, welches die potentielle Verteilung von Ressourcen und die damit verbundenen technischen

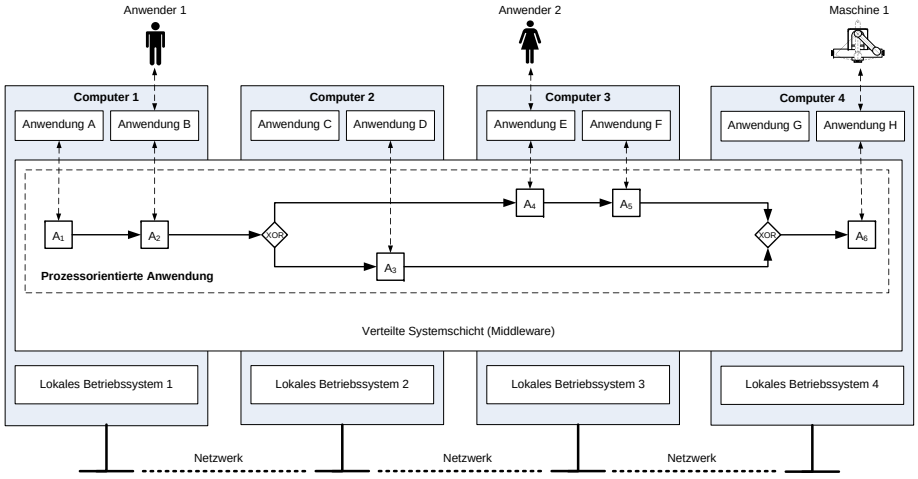


Abbildung 2.18: Middleware zur Unterstützung verteilter (prozessorientierter) Anwendungen (vgl. (LR00, TS08, Mel10))

Details der Prozessausführung vor dem Entwickler bzw. vor dem Anwender verbirgt [Jab97, DGH03, ACKM04].

Bei einer *Middleware* handelt es sich allgemein um eine zusätzliche Softwareschicht zur Unterstützung verteilter Anwendungen, wobei diese logisch zwischen einer höheren Schicht von (lokalen) Benutzern und Anwendungen und einer darunter liegenden Ebene von Betriebssystemen und grundlegenden Kommunikationsnetzwerken angeordnet ist (vgl. Abbildung 2.18) [ACKM04, TS08]. Die Laufzeitumgebung für die Ausführung von Prozessen ermöglicht durch diese Architektur eine weitestgehend plattformunabhängige Integration von verschiedenen Systemen und ihrer Ressourcen. Dabei können sowohl Ressourcen innerhalb derselben Organisationseinheit gemeinsam von verschiedenen prozessorientierten Anwendungen genutzt werden, als auch organisationsübergreifend bereitgestellt und eingebunden werden. Die Gesamtheit der an der Prozessausführung beteiligten Systeme realisieren dabei ein *verteiltes System*, welches dem Initiator der verteilten (prozessorientierten) Anwendung die benötigten Ressourcen über das Ausführungssystem zugänglich macht und gleichzeitig – sowohl aus Anwendersicht als auch aus der Sicht der beteiligten Ressourcen – deren physische oder logische Verteilung verbirgt. Dieser Aspekt wird im Kontext verteilter Systeme als *Verteilungstransparenz* bezeichnet [TS08].

Ergänzend zu allgemeinen verteilten Anwendungen haben prozessorientierte Anwendungen in Bezug auf die Anwendungsintegration jedoch erweiterte Anforderungen: Während bei einer „konventionellen“ verteilten Anwendung die Kommunikation zwischen eher homogenen Anwendungs*komponenten* mit einem relativ hohen Koppelungsgrad im Vordergrund steht, fokussiert der Ansatz der Prozessorientierung auf die Integration *eigenständiger Anwendungen*, welche oft eine hohe Heterogenität aufweisen können und nur *lose gekoppelt* sind [Ham05]. Ein Prozessausführungssystem besitzt daher in der Regel eine zentrale Komponente, welche für die Verwaltung des Kontrollflusses einer

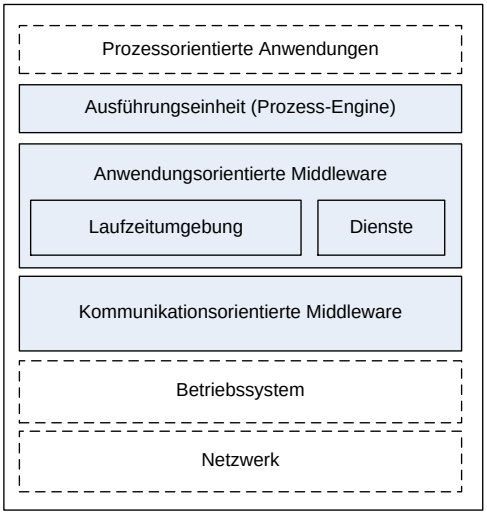


Abbildung 2.19: Einordnung und schematischer Aufbau der Prozessausführungsumgebung als Middleware zur Ausführung prozessorientierter Anwendungen (tlw. nach (Ham05))

Prozessinstanz verantwortlich ist, während der Aufruf von Anwendungen und ggf. das Entgegennehmen von Ergebnissen über Integrationsdienste geschieht, welche in der Lage sind, die Heterogenität der beteiligten Computersysteme und Netzwerke zu überwinden oder die Interaktion mit menschlichen Benutzern zu steuern. Auf dieser Ebene kann zwischen *kommunikationsorientierten* und *anwendungsorientierten Middleware-Komponenten* unterschieden werden. Kommunikationsorientierte Middleware-Komponenten stellen höheren Schichten einheitliche Schnittstellen zur Verfügung, um den Zugriff auf das zugrunde liegende Netzwerk zu regeln und somit von der Netzwerkprogrammierung zu abstrahieren. Anwendungsorientierte Middleware-Komponenten stellen darauf aufbauend zusätzlich eine Laufzeitumgebung und weitere technische Dienstkomponenten zur Verfügung, um verteilte Anwendungen weitergehend zu unterstützen. Wichtige Aufgaben dieser Schicht sind z. B. die Verwaltung von Ressourcen und Verbindungen, der Umgang mit Nebenläufigkeit, die Bereitstellung von Sicherheitsmechanismen zur Authentifizierung und Autorisierung, der Zugriff auf Verzeichnisdienste, die Transaktionsverwaltung und die persistente Speicherung von Daten [Ham05, TS08]. Abbildung 2.19 zeigt einen Überblick über die gesammelten Zusammenhänge. Dabei ist festzuhalten, dass der tatsächliche Umfang und die tatsächliche Ausgestaltung der Ausführungsumgebung implementierungsabhängig ist und zum Teil oder im Ganzen auch auf bereits bestehenden Middleware-Infrastrukturen aufsetzen kann.

2.7.2 Kontrollflussnavigation

Zur Ausführung und Steuerung des Prozessablaufs greift die Ausführungseinheit auf die ausführbare Beschreibung des technischen Prozessmodells zurück. Dazu müssen zunächst alle nach außen erforderli-

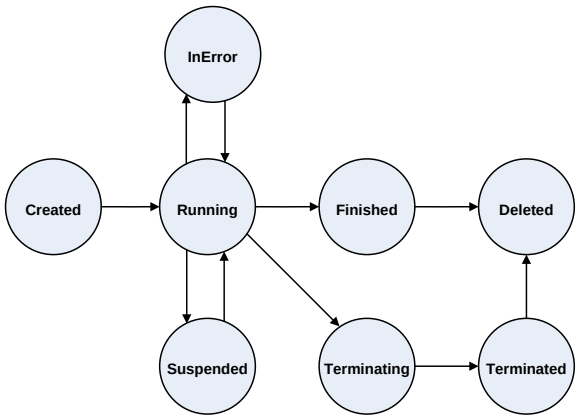


Abbildung 2.20: Zustandsmodell einer Prozessinstanz (LR00)

chen Schnittstellen für die Kommunikation mit dem Prozess bereitgestellt werden. Neben der Schnittstelle für die Instantiierung des Prozesses umfasst dies auch die notwendigen Vorkehrungen für das etwaige Abonnement von Ereignistypen, welche in der Prozessbeschreibung spezifiziert sind, sowie für den Nachrichtenaustausch einer etwaigen Interprozesskommunikation. Tritt das Ereignis zur Instantiierung des Prozesses ein, steuert die Prozess-Engine die logisch und zeitlich korrekte Abarbeitung der entstehenden Prozessinstanz. Entsprechend der zu bearbeitenden Aktivität werden hierbei entweder Softwareanwendungen aufgerufen, welche die spezifizierte fachliche Funktionalität automatisiert erfüllen, oder die auszuführende Aufgabe wird einem verantwortlichen und autorisierten Anwender zur Bearbeitung vorgelegt. Dabei erzielte Teilergebnisse werden automatisiert entlang des spezifizierten Kontrollflusses weitergeleitet, bis der Prozess vollständig abgearbeitet ist oder aufgrund eines Fehlers terminiert [Str08]. Dieses Bearbeitungsmodell wird als *Kontrollflussnavigation* [LR00] oder auch als *Routing* [Str08] bezeichnet.

Der resultierende Lebenszyklus einer Prozessinstanz kann vereinfacht durch die Angabe ihrer möglichen Zustände und der entsprechenden Zustandsübergänge beschrieben werden, welche in Abbildung 2.20 dargestellt sind [LR00] (vgl. auch [Ses10]):

- **Created:** Der *Created*-Zustand ist der Anfangszustand eines Prozesses nach dessen Erzeugung und Instantiierung.
- **Running:** Im Zustand *Running* erfolgt die eigentliche Ausführung des Prozesses und die Navigation des Kontrollflusses. Der Zustand wird erreicht, indem der Prozess explizit durch ein Ereignis oder den Aufruf eines Benutzers gestartet oder als Subprozess von einem anderen Prozess aufgerufen wird.
- **Suspended:** Im *Suspended*-Zustand wird der Prozess vorübergehend angehalten, kann aber zu einem späteren Zeitpunkt weitergeführt werden.
- **InError:** Der Zustand *InError* gibt an, dass der Prozess aufgrund eines Fehlers angehalten wurde. Entsprechend durchgeführte Fehlerbehand-

lungsmaßnahmen können den Prozess wieder in den *Running*-Zustand versetzen.

- **Finished:** Wenn die auf dem gewählten Pfad beschriebenen Aktivitäten erfolgreich bearbeitet wurden und der Prozess seinen vordefinierten Endzustand erreicht hat, befindet sich die Prozessausführung im Zustand *Finished*. Gegebenenfalls wird an dieser Stelle ein Ergebnis ausgegeben.
- **Terminating:** Wird der Abbruch des Prozesses vorbereitet, so befindet sich die Prozessausführung im Zustand *Terminating* und es werden keine neuen Aktivitäten mehr gestartet. Eventuell muss gewartet werden, bis alle laufenden Aktivitäten abgeschlossen sind.
- **Terminated:** Der *Terminated*-Zustand wird erreicht, sobald alle vorgenannten Aktivitäten beendet wurden. Der Prozess wird nicht weiter ausgeführt.
- **Deleted:** Der abgeschlossene oder abgebrochene Prozess wird schließlich aus dem System entfernt und Ressourcen, welche vom Prozess verwendet wurden, werden wieder freigegeben. Der Prozess wird danach als *Deleted* bezeichnet.

Die konkrete Ausführung einer Prozessinstanz ist vor allem von den eingegeben Daten, von den Teilergebnissen der durch die Aktivitäten ausgeführten fachlichen Funktionen sowie durch das Eintreten externer Ereignisse abhängig, welche die Wahl von Kontrollflussvarianten und damit das Ergebnis des Prozesses beeinflussen [LR00]. Das Ausführungssystem ist daher neben der Abarbeitung des Prozesses auch für die Auswertung von Bedingungen und die Verwaltung von Ereignissen zuständig. Die Ausführung einer (atomaren oder komplexen) Aktivität wird genau dann gestartet, wenn alle dafür erforderlichen Vorbedingungen erfüllt sind. Dies ist in der Regel der Fall, wenn

- es sich bei der betrachteten Aktivität um die erste Aktivität des Prozesses handelt, das den Prozess auslösende Ereignis eingetreten ist und es keine Aktivierungsbedingung für die Aktivität gibt bzw. der mit der Bedingung verbundene Ausdruck zu *true* ausgewertet wird, oder
- die Bearbeitung der im Kontrollfluss direkt vorgelagerten Aktivität(en) erfolgreich abgeschlossen wurde und sich entweder an der Transition zwischen den beiden Aktivitäten keine Exit-, Transitions- oder Aktivierungsbedingung befindet oder der mit der Bedingung verbundene Ausdruck zu *true* ausgewertet wird, oder
- ein global definiertes Ereignis eingetreten ist, welches die Ausführung der Aktivität außerhalb des regulären Kontrollflusses anstößt, oder
- der Kontrollfluss des Prozesses die Definition eines lokalen Ereignisses erreicht hat und dieses im gegebenen Zeitraum eintritt oder (im Fall von persistenten Ereignissen) bereits zuvor eingetreten war, und

- alle zur Bearbeitung der Aktivität benötigten Daten und Ressourcen vorliegen.

Im Rahmen von Verzweigungen ist entsprechend der genannten Kontrollflussmuster (vgl. Abschnitt 2.4) ebenfalls im Bearbeitungsmodell festgelegt, wie sich die Abarbeitung des Prozesses an den *Split*-Bedingungen und an den entsprechenden Synchronisationspunkten oder Zusammenführungen zu verhalten hat. Eine wichtige Aufgabe besteht darin, zu entscheiden, ob die Abarbeitung des Prozesses an einer *Join*-Bedingung anhält und wartet, bis der Kontrollfluss aller eingehenden Pfade die *Join*-Bedingung erreicht hat, bzw. welche und wie viele Bedingungen ausgewertet werden müssen, um den Prozess fortzuführen. Bei einem *OR-Split* (Mehrfachauswahl) mit entsprechendem *OR-Join* ist für den Fall, dass die Bearbeitung mehrerer paralleler Pfade gestartet wurde, eine Synchronisation nötig [vdATHKB03]. Alternativ kann z. B. eine *N-out-of-M-Join*-Bedingung fordern, dass zwei Drittel von vorgelagerten Aktivitäten erfüllt sein müssen, wobei diese bei Vorliegen zweier gleicher Ergebnisse bereits ausgewertet werden könnte, auch wenn die Bereitstellung des Ergebnisses des dritten Pfades noch aussteht [LR00]. Für den Fall, dass nur ein einzelner Pfad gewählt wurde, reicht hingegen an der *Join*-Bedingung eine einfachen Zusammenführung bzw. Fortführung des Prozesses [vdATHKB03].

Eine wichtige Maßnahme, um derartige Entscheidungen zu ermöglichen und Verklemmungssituationen zu vermeiden, ist die spezielle Markierung von nicht ausgeführten Pfaden. Bei der sogenannten *Dead Path Elimination* wird deshalb in so einem Fall der „tote Pfad“ im Kontrollfluss weiter berücksichtigt, jedoch werden die sich auf diesem Pfad befindlichen Transitionsbedingungen auf *false* gesetzt und die folgenden Aktivitäten solange übersprungen, bis eine *Join*-Bedingung oder ein Endzustand des Prozesses erreicht wird [LR00, vdADH03]. Somit wird vermieden, dass Prozesse in unkontrollierte Wartezustände verfallen und nachfolgende Aktivitäten nicht mehr ausgeführt werden, ohne dass der Prozess explizit beendet wird [vdADH03].

Unter Berücksichtigung dieser Zusammenhänge resultiert analog zur Prozessinstanz ein Zustandsmodell für Aktivitäten, welches die möglichen gültigen Zustände einer Aktivitätsinstanz angibt (vgl. Abbildung 2.21) [LR00]:

- **Inactive:** Der Zustand *Inactive* kennzeichnet den Anfangszustand einer Aktivität nach der Instantiierung des Prozesses.

- **Ready:** Eine Aktivität wird als *Ready* bezeichnet, wenn alle Vorbedingungen für ihre Ausführung erfüllt sind. Zu den Vorbedingungen gehört die erfolgreiche Bearbeitung (oder ggf. das erlaubte Überspringen) der ihr im Kontrollfluss vorgelagerten Aktivitäten sowie die Auswertung etwaiger Aktivierungs- und Anfangszeitbedingungen. Man spricht in diesem Kontext auch davon, dass die Aktivität „aktiviert“ ist.

- **Skipped:** Sind die Vorbedingungen für die Ausführung einer Aktivität nicht zu erfüllen, so wird die Aktivität in den Zustand *Skipped* versetzt.

Im Rahmen der *Dead Path Elimination* erlaubt dieser Zustand ein Fortschreiten der Kontrollflussnavigation ohne Ausführung der betroffenen Aktivität.

- **Expired:** Eine Aktivität, welche sich im Zustand *Ready* befindet, kann gegebenenfalls nicht mehr ausgeführt werden, wenn ein vorgegebener Zeitrahmen für ihre Ausführung bereits verstrichen ist, bevor sie einer Ressource zur Ausführung zugewiesen werden konnte. Die Aktivität wird dann in den Zustand *Expired* versetzt und nicht mehr ausgeführt.

- **Executing:** Der Zustand *Executing* beschreibt die eigentliche Ausführung der Aktivität. Der Zustand wird erreicht, wenn nach der Zuweisung der Ressource mit der (automatisierten) Bearbeitung der Aktivität begonnen wird.

- **InError:** Tritt bei der automatisierten Bearbeitung einer Aktivität ein Fehler auf, der nicht durch Recovery-Maßnahmen behoben werden kann, so wird die Aktivität in den Zustand *InError* versetzt.

- **Terminated:** Wird die Bearbeitung der Aktivität von außen abgebrochen, so wird sie in den Zustand *Terminated* versetzt.

- **CheckedOut:** Im Falle einer manuellen Bearbeitung wird die Aktivität einer geeigneten Person zugewiesen oder zur Bearbeitung durch eine Person ausgewählt. Die manuelle Bearbeitung selbst wird jedoch nicht durch das Ausführungssystem kontrolliert. Die für das System „unsichtbare“ manuelle Ausführung geschieht im Zustand *CheckedOut* bis dem System das Ende der manuellen Ausführung angezeigt wird. Etwaige Fehler müssen in der Zwischenzeit ebenfalls manuell behandelt werden.

- **Executed:** Der *Executed*-Zustand kennzeichnet die erfolgreiche Ausführung der durch die Aktivität vorgegebenen fachlichen Aufgabe.

- **Finished:** Sind nach der erfolgreichen Bearbeitung der Aktivität auch alle Nachbedingungen erfüllt, befindet sich die Aktivität im Zustand *Finished*. Gegebenenfalls werden an dieser Stelle etwaige Ausgabedaten der Aktivität für die Weiterverarbeitung freigegeben.

Die für die Kontrollflussnavigation benötigten Daten, der Status des Prozesses und die Zustände der einzelnen Aktivitäten werden in der Regel persistent in einer zentralen Datenhaltungskomponente wie z. B. einer Datenbank abgelegt [LR00], wo sie von der für die Prozesssteuerung zuständigen Prozess-Engine abgerufen und aktualisiert werden können. Die Datenhaltungskomponente ist in der Regel auch für die Nebenläufigkeitskontrolle zuständig, so dass die Konsistenz von innerhalb eines Prozesses oder auch prozessübergreifend gemeinsam genutzten Daten erhalten bleibt. Hierfür werden z. B. klassische Sperrmechanismen oder optimistische Verfahren zur Ne-

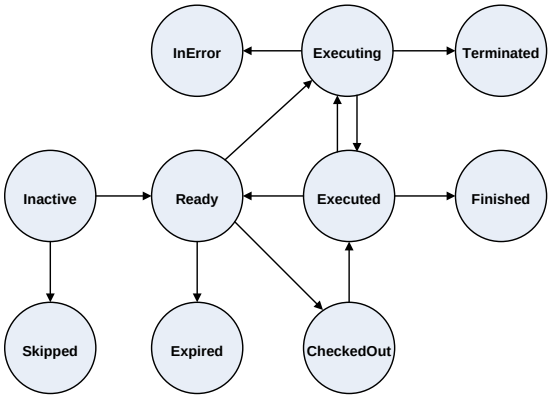


Abbildung 2.21: Zustandsmodell einer Aktivitätsinstanz (LR00, Kun08)

benläufigkeitskontrolle aus dem Bereich der Transaktionsverarbeitung eingesetzt (vgl. [GR92, LR00]).

2.7.3 Bindung und Aufruf von Ressourcen

Während der computergestützten Ausführung eines Prozesses müssen die durch die atomaren Aktivitäten spezifizierten fachlichen Aufgaben zur Durchführung an eine geeignete Ressource gebunden werden. Ist bereits eine konkrete Instanz einer Ressource in der technischen Prozessbeschreibung angegeben, so ist das Ausführungssystem lediglich für den Aufruf dieser Ressource zuständig. Gegebenenfalls müssen hierfür zur Ausführungszeit benötigte technische Details, wie zum Beispiel die Adresse bzw. der derzeitige Aufenthaltsort der Ressource in Erfahrung gebracht werden (z. B. über einen Verzeichnisdienst), um einen Aufruf bzw. eine Zuweisung zu ermöglichen. Durch die Angabe einer Rolle gekennzeichnete Ressourcen müssen zuvor unter Beachtung aller geforderter Rolleneigenschaften identifiziert werden. Diese beiden Aufgaben werden in der Regel von einem *Verzeichnisdienst* bzw. von einer *Rollenauflösungskomponente* [GG99] erbracht.

Die Bindung von Aktivitäten an Ressourcen kann in verschiedenen Zeiträumen geschehen (vgl. Abbildung 2.22). Insbesondere die Ausführung manueller Aktivitäten erfordert oft einen gewissen Planungszeitraum, um während der Ausführung einen reibungslosen Ablauf zu gewährleisten. Eine entsprechende Zuordnung von Ressourcen bereits *vor der Aktivierung der Aktivität* (d. h. vor der Überführung in den *Ready*-Zustand) wird als *frühe Bindung* (*Early Distribution*) bezeichnet [RHE05]. Ein Beispiel hierfür ist die Zuordnung von Fahrzeugführern im Rahmen der Personaleinsatzplanung eines Lieferunternehmens, damit gesetzlich vorgeschriebene Ruhezeiten eingehalten werden können. Aus Effizienzgründen ist eine frühe Bindung für prozessorientierte Anwendungen jedoch oft unvorteilhaft, da der tatsächliche Kontrollfluss einer Prozessinstanz erst zur Ausführungszeit festgelegt wird und ggf. erst kurz vor oder genau zum Zeitpunkt der Aktivierung der Aktivität feststeht, ob diese überhaupt ausgeführt werden soll und welche Ressource in dem

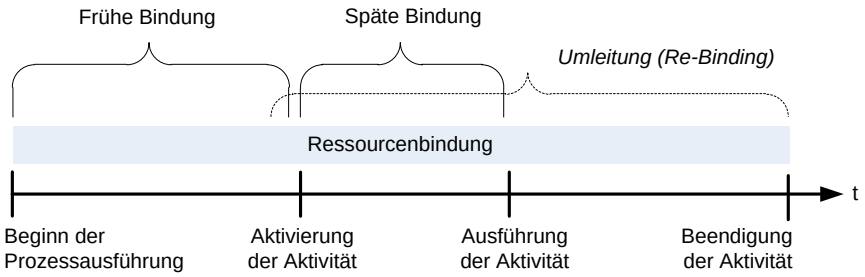


Abbildung 2.22: Zeitliche Einordnung der Ressourcenbindung zur Ausführung einer Aktivität

vorliegenden temporären Kontext hierfür geeignet ist. Im Kontext der Anwendungsintegration und bei prozessorientierten Anwendungen ohne Beteiligung menschlicher Akteure ist daher in der Regel eine *späte Bindung* vorteilhaft, wobei nach RUSSELL nochmals zwischen einer Zuweisung zum exakten Zeitpunkt der Aktivierung (*Distribution on Enablement*) und einer verzögerbaren Bindung (*Late Distribution*) unterschieden werden kann [RHE05]. Die letztere Form der Zuweisung erlaubt es zum Beispiel, auf eine geeignete Ressource zu warten, wobei die Optimalität der Ressource gegen den entstehenden Zeitverlust durch die Verzögerung abgewogen werden muss [SLI08]. Desweiteren können konkrete Ressourcen, welche bereits in einem ersten Schritt gebunden wurden, bei Bedarf durch andere (äquivalente) Ressourcen ersetzt werden (*Re-Binding*).

Kommen mehrere Ressourcen für die Ausführung einer Aktivität in Betracht, können verschiedene Strategien angewendet werden, um die für die Ausführung verantwortliche Ressource zu bestimmen. Hierbei wird oftmals neben der Auswahl nach dem *Zufallsprinzip* oder nach *Zirkulationsverfahren* eine Auswahl der Ressource mit der *geringsten Arbeitsbelastung* vorgenommen [RHE05]. Neben einer solchen automatischen Zuweisung von Aktivitäten durch das Ausführungssystem (*Push-Pattern*) können Aktivitäten jedoch auch aktiv von Ressourcen ausgehend zur Bearbeitung ausgewählt werden (*Pull-Pattern*). Hierbei werden die auszuführenden Aktivitäten in eine Warteschlange eingefügt und können von geeigneten Ressourcen entnommen werden, sobald Kapazitäten für die Ausführung vorhanden sind [LR00].

Aufruf von Softwareanwendungen

Bei einem Aufruf von Softwareanwendungen werden die von der Implementierung benötigten Daten zur Ausführung der fachlichen Funktion mittels lokalem Methodenaufruf oder durch das Senden einer Nachricht über das Netzwerk übermittelt (*Eingabedaten*). Die Menge der Eingabedaten kann dabei auch leer sein. Die angestoßene Funktion kann ein Ergebnis erzeugen, welches als Rückgabewert an das Prozessausführungssystem zurückgeliefert wird (*Ausgabedaten*). Die Anwendungen können dabei synchron oder asynchron in den Prozess eingebunden sein [DGH03]. Bei einer *synchronen Kommunikation* ist es erforderlich, dass sowohl die aufrufende als auch die aufgerufene Partei

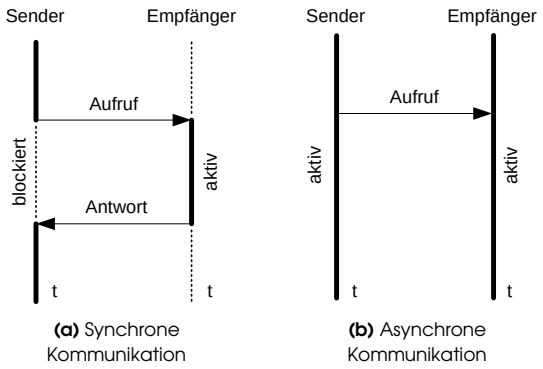
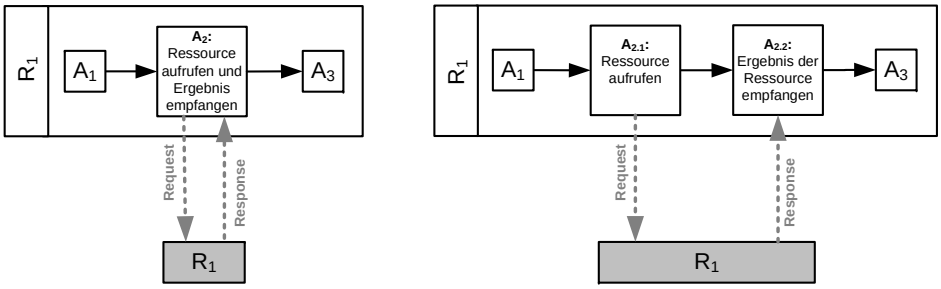


Abbildung 2.23: Synchrone und asynchrone Kommunikation auf technischer Ebene (ACKM04, Ham05)

während der Dauer der Ausführung verfügbar und durch ein Kommunikationsmedium miteinander verbunden sind. Zudem muss die aufrufende Partei warten, bis die aufgerufene Partei ihre Bearbeitung beendet hat (vgl. Abbildung 2.23a). Bei einer *asynchronen Kommunikation* ist dies nicht notwendig, d. h. beide Parteien müssen nicht dauerhaft verfügbar und miteinander verbunden sein, und die aufrufende Partei ist für die Zeit der Ausführung nicht blockiert (vgl. Abbildung 2.23b) [LR00, Ham05]. In diesem Hinblick ergeben sich aus der Sicht der prozessorientierten Anwendung vier Muster der Interaktion, welche von der Ausführungseinheit unterstützt werden müssen:

- Einseitiger bzw- asynchroner Aufruf einer Ressource von Seiten der Ausführungseinheit (*One Way*)
- Synchroner Aufruf einer Ressource von Seiten der Ausführungseinheit (*Request-Response*)
- Einseitiger Aufruf der Ausführungseinheit von Seiten einer Ressource bzw. Antwort einer Ressource auf einen asynchronen Aufruf der Ausführungseinheit (*Notification*)
- Synchroner Aufruf der Ausführungseinheit von Seiten der Ressource (*Solicit-Response*)

Eine aus Request und Response bestehende asynchrone Kommunikation mit einer Ressource ist für prozessorientierte Anwendungen in der Regel nur geeignet, wenn das Ergebnis der ausgeführten Aktivität nicht direkt nachfolgenden Aktivitäten zur Verfügung stehen muss. Andernfalls darf die Kontrollflussnavigation nicht vor Beendigung der Verarbeitung durch die Ressource fortgeführt werden, was durch eine synchrone Kommunikation mit der Ressource besser abzubilden ist (vgl. Abbildung 2.24a). Handelt es sich jedoch um eine potentiell länger dauernde und logisch entkoppelte Tätigkeit, so kann während der angestoßenen Ausführung durch die gewählte Ressource bereits mit der Ausführung nachfolgender Aktivitäten begonnen und das Ergebnis



(a) Synchroner Aufruf einer Ressource **(b)** Asynchroner Aufruf einer Ressource

Abbildung 2.24: Synchrones und asynchrones Kommunikationsmodell auf Anwendungsebene

des Ressourcenaufrufs zu einem späteren Zeitpunkt integriert werden. Eine solche asynchrone Kommunikation muss jedoch in der Regel aufwendig explizit im Prozess modelliert werden (vgl. Abbildung 2.24b). Um eine möglichst lose Koppelung zwischen der Prozesslogik und den beteiligten Anwendungssystemen zu erreichen, sollte die Verwendung von synchroner und asynchroner Kommunikation auf Anwendungsebene daher vor allem fachlich motiviert sein. Unterstützt daher eine Ressource lediglich ein synchrones bzw. asynchrones Kommunikationsmodell, so müssen die entsprechenden Nachrichten durch die Integrationsdienste der Ausführungsumgebung verwaltet und der Anwendungsschicht in geeigneter Form bereitgestellt werden.

Einbindung von menschlichen Akteuren

Bei einer Aktivität, welche die Beteiligung oder vollständige Bearbeitung durch einen menschlichen Akteur erfordert, spricht man im Rahmen des Workflow Managements bzw. des *Human Task Managements* oft von einem *Workitem* [LR00, DGH03]. Bei einer computergestützten Ausführung ist für die Interaktion des verantwortlichen Systems mit einem menschlichen Benutzer stets eine geeignete *Benutzungsschnittstelle* erforderlich, über die dem Benutzer die auszuführende Aufgabe vermittelt und nach Beendigung von Seiten des Benutzers die Fertigstellung der manuellen Ausführung angezeigt werden kann (*Worklist*) [DGH03]. Bei der Zuweisung wird dabei entweder einer ausgewählten Person das Workitem direkt über eine Benutzungsschnittstelle präsentiert oder in eine Warteschlange eingefügt, aus der die ausgewählte Person oder eine Gruppe von geeigneten Personen dieses entnehmen kann, sobald Kapazitäten für dessen Bearbeitung vorhanden sind [LR00]. Bei dieser Interaktion können zudem etwaige Daten zur Ausführung an den Benutzer übermittelt sowie am Ende der Ausführung vom Benutzer erarbeitete Ergebnisdaten an das System zurück übertragen werden.

Kommt es zu einem Engpass an Ressourcen, der durch die Auslastung oder Unverfügbarkeit von Ressourcen ausgelöst wird, können verschiedene Strate-

gien angewendet werden. Handelt es sich um einen Engpass an menschlichen Akteuren, können Workitems an andere qualifizierte Personen *umgeleitet* werden (vgl. Abbildung 2.22). Schlimmstenfalls muss hierfür ggf. die zugrunde liegende Organisationsstruktur bzw. die vorhandene Rollendefinition geändert werden. Ein Engpass oder Ausfall an Maschinen bzw. die Überlastung von Softwarekomponenten ist hingegen nicht so einfach zu kompensieren, da hier festgelegte technische Schnittstellen vorliegen und so eine einfache Austauschbarkeit auch von Ressourcen mit nur geringer Heterogenität erschwert wird. Eine mögliche, aber nicht immer akzeptable Lösung stellt die vorübergehende Einstellung anderer Aktivitäten dar, welche die belastete Ressource verwenden. Eine andere Möglichkeit besteht darin, die von dem Engpass betroffene Prozessinstanz an eine andere Instanz des Prozessmanagementsystems zu übertragen, falls für diese eigene Ressourcen bereitstehen [LR00].

2.8 Integrierter Prozesslebenszyklus

Basierend auf den dargestellten Eigenschaften prozessorientierter Anwendungen und ihrer Unterstützung durch Informations- und Kommunikationstechnologie kann analog zu allgemeinen Verfahren der Softwareentwicklung ein Phasenmodell für die Modellierung und technische Umsetzung eines Prozesses auf Anwendungsebene abgeleitet werden. Im Vergleich zu den meisten Phasenmodellen zur Entwicklung von Softwareprodukten (wie z. B. des *Wasserfall-*, *Spiral-* oder *V-Modells*) [Ver02, VH03], welche durch einen hauptsächlich gerichteten linearen oder iterativen, jedoch in der Regel zeitlich terminierenden Entwicklungsverlauf gekennzeichnet sind, unterliegt der Einsatz prozessorientierter Anwendungen einem höheren Grad an Dynamik und zeichnet sich dementsprechend durch einen *fortwährenden Anpassungsbedarf* aus [Ses10]. Das vorliegende Entwicklungsmodell für diese Art von Anwendungen ist daher in der Regel durch einen kontinuierlichen Verbesserungsprozess [All05] geprägt und gibt daher einen inhärent zyklischen Ablauf vor. Man spricht daher in der Literatur auch von einem *Prozessmanagement-Kreislauf* oder *Prozesslebenszyklus* (vgl. [vdAvH02, All05, Mül05, Gad10, Wes07, Ses10]).

Insbesondere im betrieblichen Kontext steht eine fortwährende Optimierung der fachlichen (Geschäfts-) Prozesse im Vordergrund, um die Wettbewerbsfähigkeit des Unternehmens zu erhöhen oder auf Dauer zu erhalten. Diese Optimierung beginnt in der Regel bereits durch die Identifikation und explizite Modellierung des relevanten Realweltausschnitts. In einer *Analysephase* wird dabei der Anwendungsbereich zunächst untersucht (*Ist-Zustand*) und es werden die strategischen Ziele und erwarteten Ergebnisse des Prozesses definiert, was oft eine Restrukturierung der vorgefundenen Prozesse erforderlich macht (*Soll-Zustand*) [All05, Gad10, Wes07]. Hierbei können auch nicht-funktionale fachliche Aspekte vorgegeben werden, wie zum Beispiel Grenzwerte für die Prozessdauer oder die hierdurch verursachten Kosten [Gad10]. Anhand der Definition von *Kennzahlen* kann der Fortschritt oder der Erfüllungsgrad hinsichtlich wichtiger Zielsetzungen gemessen bzw. ermittelt

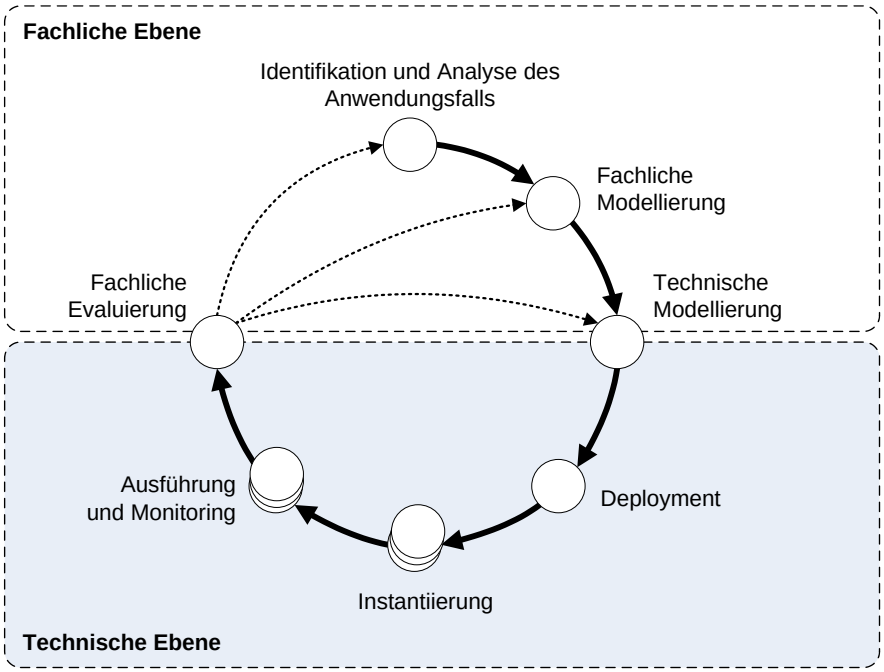


Abbildung 2.25: Lebenszyklusmodell einer prozessorientierten Anwendung (tlw. nach (Mül05, Gad10, Wes07, Ses10))

werden. Im betriebswirtschaftlichen Kontext werden diese Kennzahlen als *Key Performance Indicators (KPIs)* bezeichnet [vdAvH02, Mau09, KS10].

Im Anschluss wird identifiziert, welche fachlichen Aufgaben in welcher Reihenfolge auszuführen sind, um die strategischen Ziele unter den festgelegten Bedingungen zu erreichen. Dieser Schritt entspricht der in Abschnitt 2.3 eingeführten operativen *fachlichen Prozessmodellierung*. Das resultierende Prozessmodell wird für eine automatisierte Ausführung für die gewählte technische Plattform auf ein *technisches Prozessmodell* abgebildet. Gegebenenfalls können an dieser Stelle eine *Simulation* und *Validierung* der Prozessausführung erfolgen [Gad10, Wes07, Ses10], bevor der Prozess schließlich in den Produktivbetrieb überführt wird. Dazu wird das fertige technische Prozessmodell im Rahmen des *Deployments* auf dem Ausführungssystem installiert und kann von diesem Zeitpunkt durch *Instantiierung* ausgeführt werden [Ses10].

In der *Ausführungsphase* ermöglicht die computergestützte Steuerung des Prozessablaufs das systematische und (im Vergleich zu einer manuellen Verwaltung) beschleunigte Abarbeiten aller anstehenden Aktivitäten in logisch richtiger Reihenfolge. Bereits während der Ausführung kann dabei der Status des Prozesses nachverfolgt, der Aufruf von Ressourcen und die dabei erbrachten Teilergebnisse überwacht sowie die Einhaltung von Kennzahlen kontrolliert werden [Wes07]. Die Möglichkeiten zum lokalen *Monitoring* erlauben es dabei, Informationen über den aktuellen Stand der Prozessbearbeitung in Echtzeit zu beziehen und ein permanentes *Tracking* des Prozessfort-

schritts zu unterstützen. Ebenso wird es möglich, die temporäre Auslastung und Leistung der integrierten Ressourcen zu bestimmen [Gad10, Str08]. Es entsteht somit hierbei ein zusätzlicher qualitativer Mehrwert über die eigentliche Ausführung der fachlichen Anwendungsaufgabe hinaus [All05].

Die während der Ausführung von Prozessinstanzen gesammelten Daten und deren Ergebnisse können schließlich nach Beendigung der Prozessausführung verdichtet, evaluiert und mit den vorgegebenen Kennzahlen verglichen werden (*Fachliche Evaluierungsphase*) [Wes07]. Auf Basis der gesammelten Erkenntnisse können Änderungen an technischen Prozessmodellen, fachlichen Prozessmodellen oder sogar an der strategischen Ausrichtung und den Zielen der prozessorientierten Anwendung notwendig werden [Gad10]. Dabei resultiert der Änderungsbedarf nicht zwangsläufig aus der (noch) nicht optimalen Abbildung des Anwendungsfalls, sondern ist oft aufgrund externer Ereignisse und Veränderungen innerhalb der *dynamischen Systemumgebung* erforderlich. Durch die fortwährende Überprüfung und Anpassung der fachlichen Logik und der Steuerungslogik kann dieser Änderungsbedarf erkannt und auf der entsprechenden Ebene auf die geänderten Bedingungen reagiert werden [vdAvH02, All05].

In der Literatur ist der resultierende Prozesslebenszyklus in verschiedenen Ausprägungen mit unterschiedlicher Anzahl bzw. Granularität der genannten Phasen und mit Fokus auf bestimmte Anwendungsdomänen zu finden. In dieser Arbeit wird ein weitestgehend neutrales Lebenszyklusmodell verwendet, welches sich auf Basis des verwendeten Prozessbegriffs in eine fachliche und eine technische Ebene unterteilen lässt. Abbildung 2.25 fasst die charakteristischen Eigenschaften dieses allgemeinen integrierten Prozesslebenszyklus zusammen. Es ist hierbei auffällig, dass das Potential zur Optimierung der prozessorientierten Anwendung in der Regel nur zur Entwicklungszeit von Prozessmodellen und nur auf fachlicher Ebene aufgezeigt wird. Auf technischer Ebene und zur Laufzeit einzelner Prozessinstanzen wird in der betrachteten Literatur keine Anpassungsfähigkeit des Prozesses dargestellt.

2.9 Zusammenfassung

Prozessorientierte Anwendungen stellen bei der Entwicklung verteilter Systeme eine wichtige Grundlage dar, um die für ein Anwendungsgebiet der Realwelt relevanten Vorgänge mittels Informations- und Kommunikationstechnologien abzubilden und in ihrem Ablauf systemübergreifend unterstützen zu können. Die wesentlichen Vorteile bestehen dabei in der losen Koppelung von autonomen Anwendungssystemen, dem dadurch resultierenden geringen Grad an Abhängigkeit zwischen Systemkomponenten, ihrer prozessübergreifenden Wiederverwendbarkeit und der einfachen Durchführbarkeit von Änderungen sowohl an den lokalen Systemen als auch an der prozessorientierten Anwendung selbst.

Domänenübergreifend lassen sich mit dem Management von Ablauforganisationen, der Integration von Anwendungssystemen und der Steuerung von

Kommunikationsabläufen drei wesentliche Anwendungsgebiete für computergestützte Prozesse identifizieren, wobei insbesondere der *Integrationssystemcharakter* [Str08] der Prozessorientierung im Vordergrund steht. Die sich ergebene Unterstützung von kooperativen Vorgängen und die damit verbundenen positiven Effekte in Bezug auf Flexibilität, Effizienz und Kosteneinsparungen führen dazu, dass ein Großteil der praktischen Anwendungsdomänen in einem wirtschaftlichen Kontext steht (Abbildung von *Geschäftsprozessen*) oder eine Integration von Ressourcen in ähnlich dynamischen Umgebungen ermöglicht.

Durch die Vermeidung monolithischer Strukturen und die Trennung von fachlicher Anwendungslogik und technischer Ausführungslogik stellen Prozesse ein wesentliches Mittel zur Flexibilisierung verteilter Anwendungen dar. Die lose Kopplung und späte Bindung von Ressourcen über die Zuweisung von Individuen zu Rollen erlaubt eine weitestgehende Austauschbarkeit von Ressourcen zur Laufzeit des Prozesses. Die Möglichkeit zur Überwindung heterogener Systeme durch die Anwendung von aktuellen Middleware-Technologien ist ein vielversprechender Ansatz, um auch über Organisationsgrenzen hinaus weitestgehend von der Verteilung von Ressourcen zu abstrahieren.

Darauf aufbauend ist die Anpassungsfähigkeit prozessorientierter Anwendungen an geänderte Anforderungen im Regelfall durch das Durchlaufen des Prozesslebenszyklus zur kontinuierlichen Verbesserung des Prozesses determiniert. Sofern es sich hierbei um einen langfristigen, inhaltlichen Entwicklungsprozess handelt, kann hierdurch ein umfassendes Anpassungspotential ausgeschöpft werden. In Fällen von kurzfristigen, technischen Änderungsanforderungen, die bei der Ausführung einer Prozessinstanz im Einzelfall auftreten können, ist diese Anpassungsfähigkeit jedoch oft nicht ausreichend. Im folgenden Kapitel werden daher ergänzend zu der dargestellten allgemeinen Entwicklungsmethodik prozessorientierter Anwendungen Möglichkeiten zu deren *Flexibilisierung* diskutiert.

3 Flexibilität und Anpassbarkeit prozessorientierter Anwendungen

Sowohl auf der fachlich-strategischen als auch auf der operational-technischen Ebene stellt das Bestreben nach Flexibilität und Anpassungsfähigkeit einen fundamentalen Beweggrund für die Entwicklung prozessorientierter Anwendungen dar [Wes07, Mar10]. Moderne Informations- und Kommunikationssysteme werden dabei gleichzeitig als Ursache, aber auch als Adressat von Flexibilitätsanforderungen gesehen [GP00, KB04a]. Einerseits erlaubt ein Fortschritt im Bereich der Automatisierungstechnik neue Organisationsstrukturen und Kooperationsformen, bei denen schnell und flexibel auf ungenutzte Ressourcen zugegriffen werden kann. In Hinblick auf die Flexibilität bestehen aber andererseits auch negative Auswirkungen dieser Technologien, da durch einen höheren Automatisierungsgrad oftmals starre Strukturen resultieren [GP00], die im Einzelfall nur schwer zu umgehen sind.

Eben diese Einzelfälle gewinnen jedoch zunehmend an Relevanz. Zahlreiche theoretische und empirische Studien weisen auf eine fortwährende Bedeutungszunahme des Erfolgsfaktors Flexibilität insbesondere im betriebswirtschaftlichen Kontext hin (vgl. [KB04a, VMSS07]). Gründe hierfür sind ein zunehmender globaler Wettbewerb, die steigende Diffusion moderner Informations- und Kommunikationssysteme und eine sich immer schneller wandelnde Umwelt [KB04a, VMSS07]. Hinzu kommt ein erhöhter Drang nach Differenzierung und Individualisierung von Produkten und Leistungen, und die Notwendigkeit, temporäre Diskontinuitäten möglichst ohne Verlust dieser Leistungsfähigkeit überwinden zu können [VMSS07]. Durch den erhöhten Stellenwert der Flexibilität ergibt sich damit ein Trend zur sogenannten *flexiblen Automatisierung* [KB04a], welcher für die Entwicklung und den Einsatz von geeigneten Informations- und Kommunikationssystemen große Herausforderungen birgt.

Dieses Kapitel untersucht den Stand der Forschung in Hinblick auf Flexibilität und Anpassbarkeit prozessorientierter Anwendungen. Dazu wird zunächst eine Taxonomie der Flexibilität eingeführt, auf deren Basis bestehende Ansätze zur Flexibilisierung von Prozessen eingeordnet werden können (vgl. Abschnitte 3.1 bis 3.3). Im Anschluss daran werden grundlegende aktuelle Strategien und Paradigmen zur Realisierung flexibler prozessorientierter Anwendungen untersucht. In technischer Hinsicht hat sich hierbei vor allem im Rahmen der modularen Softwareentwicklung das Konzept der *dienstorientierten Architektur* durchgesetzt, welches die Kapselung von Softwarefunktionalitäten als elektronische Dienste und damit die flexible Einbindung und den Austausch grundlegender Ressourcen unterstützt (vgl. Abschnitt 3.4). Des Weiteren hat es sich als Reaktion auf eine zunehmend dynamische Umwelt als

vorteilhaft erwiesen, den relevanten Kontext prozessorientierter Anwendungen zu identifizieren, zu beobachten und die gesammelten Informationen für die Prozessausführung nutzbar zu machen. Strategien und Ansätze zu diesem Thema werden unter dem Begriff des *Context-Aware (Business) Process Management* zusammengefasst (vgl. Abschnitt 3.5).

Um die Reaktion auf wahrgenommene Umgebungsänderungen flexibel zu gestalten, wird die Spezifikation von *fachlichen Regeln* angestrebt, welche weitestgehend von der Kontrollflusslogik des Prozesses entkoppelt sind und so einfache Änderungen an den Regelwerken, ggf. sogar zur Laufzeit des Prozesses, erlauben. Der Umgang mit solchen Regeln ist Inhalt des Fachgebiets *Business Rule Management*, welches in Abschnitt 3.6 vorgestellt wird. Hiermit verwandt spielt zudem das Auftreten von *Ereignissen*, welches die Prozessausführung positiv oder negativ beeinflussen kann, eine besondere Rolle. Das Forschungsgebiet des *Event-based (Business) Process Management* misst daher der Erkennung, der Auswertung und der (regelbasierten) Integration solcher Ereignisse im Rahmen prozessorientierter Anwendungen eine besondere Bedeutung bei (vgl. Abschnitt 3.7).

Auf der Basis der genannten grundlegenden Ansätze kann auf einer höheren Ebene die Automatisierung von Anpassungen unterstützt werden, indem den betroffenen Systemen die Möglichkeit zur Selbstbestimmung ihres Verhaltens anhand vorgegebener Ziele ermöglicht wird. Methoden zur Auswahl und Umsetzung von geeigneten Strategien zur Erreichung dieser Ziele werden u.a. durch das Paradigma des *Autonomic Computing* (vgl. Abschnitt 3.8.1) und den Ansatz der *agentenorientierten Softwareentwicklung* (vgl. Abschnitt 3.9) vorgeschlagen. Im Kontext prozessorientierter Anwendungen wird aufgrund der Schwerpunktsetzung der beiden Forschungsgebiete in dieser Arbeit das Potential des Autonomic Computing insbesondere in Hinblick auf die Erreichung technischer Ziele und die Agentenorientierung als Ansatz zur Erreichung fachlicher Ziele genauer untersucht. Das Kapitel schließt mit einer Zusammenfassung der erarbeiteten Flexibilisierungsanforderungen und -möglichkeiten prozessorientierter Anwendungen im Allgemeinen.

3.1 Einführung und Begriffsklärung

In seiner ursprünglichen lateinischen Wortbedeutung und im heutigen allgemeinen Sprachgebrauch bezeichnet der Begriff der *Flexibilität* eine Eigenschaft physischer Objekte, welche es erlaubt, nach Formveränderungen wieder ihren ursprünglichen Zustand annehmen zu können [KB04a]. Dies entspricht im Wesentlichen dem Verständnis von *Biegsamkeit*. Aufgrund der vielschichtigen Verwendung des Flexibilitätsbegriffs in unterschiedlichen Disziplinen hat sich jedoch eine große Bandbreite an spezialisierten Definitionen herausgebildet, welche sich nicht nur auf physische Objekte beschränken (vgl. [VMSS07]). So hat sich insbesondere in der Betriebswirtschaft Flexibilität als Voraussetzung für eine langfristige Anpassung im Sinn einer *Evolution* [KB04a] durchgesetzt, wobei naturgemäß keine Forderung an eine Rückkehr

zum ursprünglichen Zustand gestellt wird. Flexibilität wird hierbei eher den Eigenschaften *Agilität* bzw. *Wandlungsfähigkeit* gleichgestellt, so dass sich mit der Bedeutung der *Anpassungsfähigkeit an wechselnde Umstände* eine allgemeinere Sichtweise des Begriffes mit dem Ziel der *Anpassung* (*Adaption*) herausgebildet hat. Vor dem Hintergrund einer etwaigen Rückkehr zu vorherigen Zuständen ist dabei demnach zu unterscheiden, ob die Anpassung nur temporär oder längerfristig bzw. dauerhaft ist.

Die *Anpassungsfähigkeit* beschreibt die Eigenschaft, einen angestrebten Gleichgewichtszustand zwischen System und Umwelt herzustellen und zu halten [Ede88, KB04a]. Die Anpassungsfähigkeit eines Systems kann dabei lediglich eine passive Eigenschaft darstellen, d. h. das System kann durch externe Eingriffe an veränderte Bedingungen angepasst werden. Diese passive Eigenschaft wird zur genaueren Unterscheidung als *Adaptierbarkeit* bezeichnet. Kann die Anpassung hingegen *selbständig* durch das System geschehen, spricht man von *Adaptivität* [Leu02, Kla04]. Unabhängig von dieser Unterscheidung kann das Verhalten gegenüber der Umwelt *reaktiv* sein, d. h. erst nach dem Eintreten eines Ereignisses initiiert werden, oder *proaktiv* sein, d. h. vorausschauend auf ein potentiell oder sicher eintretendes Ereignis in der Zukunft initiiert werden [GP00, VMSS07]. Für den Fall, dass keine besondere Unterscheidung für das Verständnis relevant ist, wird in dieser Arbeit der allgemeinere Begriff der Anpassungsfähigkeit als Oberbegriff dieser Eigenschaften verwendet.

Anpassungsfähigkeit und Flexibilität werden insbesondere im Kontext der Betriebswirtschaft oft synonym verwendet [Pib01, Dam02]. Nach den Autoren KALUZA und BLECKER ist Flexibilität demnach die Eigenschaft eines Systems, proaktive oder reaktive sowie zielgerichtete Änderungen einer Systemkonfiguration zu ermöglichen, um die Anforderungen von sich ändernden Umweltbedingungen zu erfüllen [KB04a]. Die Autoren VOIGT et al. verfolgen eine ähnliche Sichtweise, geben aber zusätzlich an, dass die Veränderungsbewältigung dazu dient, um das Erreichen der *bisherigen Ziele* zu garantieren [VS05]. Dies impliziert, dass die als wesentlich erachteten Eigenschaften des Betrachtungsgegenstands bei einer Anpassung erhalten bleiben.

Im Bereich der Natur- und Ingenieurwissenschaften wird basierend auf der Bedeutung von Biegsamkeit eine komplexere Beziehung gesehen, die darin besteht, dass Flexibilität eine Grundvoraussetzung dafür ist, eine Art von Anpassungsfähigkeit zu ermöglichen, welche das anzupassende Objekt nicht in einer Weise verändert, die zukünftige weitere Anpassungen verhindern würde (*reversible Formveränderung* bzw. „biegen, ohne zu brechen“). Dies inkludiert sowohl die mögliche Rückkehr zum ursprünglichen Zustand als auch alternative oder konsekutive Anpassungsschritte im Rahmen eines evolutionären Prozesses. Die umfassende Möglichkeit, sich auf diese Art *fortwährend anpassungsfähig* verhalten zu können, liegt demnach in der Flexibilität. Somit kann die Flexibilität als Mittel zum Zweck der Anpassungsfähigkeit aufgefasst werden [GP00, VS05, RSS06a, VMSS07]:

Flexibilität *bezeichnet die Eigenschaft eines Systems, ohne unverhältnismäßigen Aufwand fortwährend Anpassungen an veränderte Bedingungen zu erlauben, ohne dabei das System komplett zu ersetzen.*

Da kein allgemeingültiges Modell zur Messung und Bewertung von Flexibilität existiert (vgl. [KB04a]), hängt es in der Regel vom Anwendungskontext ab, ob einem Objekt die Eigenschaft der Flexibilität zugesprochen wird oder nicht [KB04a]. Bei der Bewertung der Flexibilität ist es dabei in der Regel wesentlich, dass eine mögliche Anpassung keinen oder nur geringen *Aufwand* verursacht und die als wesentlich erachteten Eigenschaften des Objektes bei der Anpassung erhalten bleiben [KB04a, All05, VMSS07]. Zum Beispiel würde ein solider Eisenträger in der Regel nicht als „flexibel“ bezeichnet werden, obwohl man ihn durch Einschmelzen durchaus (auch wiederholt) in eine andere Form bringen könnte. Ebenso unterliegt die Bewertung von Flexibilität auch bei nicht-physischen Betrachtungsgegenständen einer hohen Subjektivität.

Um eine objektive Einschätzung von Flexibilität zu erarbeiten, wird daher im Folgenden der Flexibilitätsbegriff für den vorliegenden Anwendungskontext dieser Arbeit qualitativ klassifiziert und im Kontext des Untersuchungsgegenstands prozessorientierter Anwendungen genauer betrachtet.

3.2 Dimensionen der Flexibilität soziotechnischer Systeme

Wie im letzten Abschnitt erläutert wurde, hängt die subjektive Einschätzung von Flexibilität stark mit dem Aufwand zusammen, der für eine notwendige Anpassung des betrachteten Objekts benötigt wird. Prozessorientierte Anwendungen können den technischen bzw. den soziotechnischen Systemen zugeordnet werden [Alt08], deren Flexibilität zumindest durch eine qualitative Bewertung anhand von Eigenschaften und Klassifikationen eingeschätzt werden kann [GP00]. Nach GOLDEN und POWELL können in Bezug auf solche Systeme im Allgemeinen die vier verschiedenen Dimensionen *Zeit*, *Anpassungsspielraum*, *Verhalten* und *Wirkungsrichtung* identifiziert werden [GP00]. Als weitere Dimension kann zudem nach KALUZA und EICKER et al. die *Objektdimension* betrachtet werden [Kal93, KB04a, ENS07]. Die folgende Aufstellung fasst die bestehenden wissenschaftlichen Ansätze zur Klassifikation von Flexibilitätsarten zusammen:

- **Objektdimension**: In Bezug auf das Objekt der Flexibilität können allgemein *Zielflexibilität* und *Mittelflexibilität* unterschieden werden [Kal93]. Die Zielflexibilität ermöglicht die Aufnahme neuer Ziele sowie die Veränderung des Zielsystems oder des Zielerreichungsgrades eines Objekts [KB04a, ENS07]. Es handelt sich hierbei also um inhaltliche Anpassungen eines Objekts. Die Mittelflexibilität ist die Voraussetzung für eine geeignete Anpassung der Mittel, die zur Erreichung dieser Ziele eingesetzt werden können [KB04a, ENS07]. Dabei handelt es sich also um organisatorische, personelle oder technische Anpassungen.

- **Zeitdimension**: Diese Dimension umfasst die *Zeitdauer*, welche ein System benötigt, um ausgehend von der Feststellung eines externen oder internen Ereignisses die notwendige Anpassung hierauf vorzunehmen und beizubehalten [VMSS07]. Diese Zeitdauer lehnt sich dabei in der Regel an eine Typologie der auslösenden Ereignisse an (vgl. [GP00]). Demnach kann eine allgemeine Klassifizierung durch die Einteilung in *kurzfristige* (*operative*), *mittelfristige* (*taktische*) und *langfristige* (*strategische*) Anpassungsfähigkeit vorgenommen werden. Kurzfristige Anpassungen betreffen die Behandlung von Einzelfällen, wie zum Beispiel eines Lieferengpasses oder dem temporären Ausfall einer Maschine. Mittelfristige Anpassungen sind aufgrund von gelegentlichen Schwankungen nötig, wie zum Beispiel die Änderung von Produktionsraten eines Fertigungsunternehmens aufgrund saisonaler Unterschiede in der Nachfrage. Langfristige Anpassungen sind meist nicht reversibel und betreffen eine Überarbeitung oder eine Neuausrichtung strategischer Ziele. Beispiele sind Investitionen und Expansionen in neue Märkte [GP00, VMSS07]. Die beiden Extrema als Voraussetzung für kurzfristige und langfristige Anpassungsfähigkeit werden im Kontext der betriebswirtschaftlichen Flexibilitätstheorie auch als *Bestands-* bzw. *Entwicklungsflexibilität* bezeichnet [KB04a, ENS07].

- **Anpassungsspielraum**: Der *Anpassungsspielraum* gibt die Reichweite bzw. die Anzahl der Handlungsalternativen an, welche einem System bereitstehen, um auf verschiedene Situationen zu reagieren [GP00]. Die Aktions- bzw. Reaktionsreichweite wird dabei in Hinblick auf ihre Qualität im Umgang mit verschiedenen Ereignissen bewertet [VMSS07]. Nach CARLSSON [Car89] werden dabei zwei Stufen unterschieden: *Typ-I-Flexibilität*, welche den Umgang mit *vorhersehbaren Ereignissen* erlaubt, und *Typ-II-Flexibilität*, welche den Umgang mit *ungeplanten* oder sogar *unvorhersehbaren Ereignissen* gestattet.

- **Verhaltensdimension**: Den möglichen Verhaltensweisen eines Systems wird im Rahmen der Bewertung von Flexibilität eine hohe Bedeutung beigemessen (vgl. [GP00, KB04a, VS05, VMSS07]). Es wird hierbei unterschieden, ob ein System anpassungsfähig ist, nachdem eine bestimmte Situation eingetreten ist (*Reaktivität*) oder ob es möglich ist, durch offensives Verhalten eine Anpassung zu ermöglichen, bevor eine bestimmte Situation eintritt, oder diese Situation sogar auszulösen bzw. gänzlich zu vermeiden (*Proaktivität*) [GP00, VMSS07].

- **Wirkungsdimension**: Diese Dimension beschreibt den Umfang des Bereichs, in dem die Anpassungsfähigkeit eines Systems ermöglicht wird und/oder Auswirkungen auf seine Umwelt hat. Es kann zwischen *interner* und *externer Anpassungsfähigkeit* unterschieden werden. Interne Anpassungsfähigkeit eines Systems wird durch dessen eigene Organisationsstruktur erreicht, wie z. B. durch die Flexibilisierung von unterneh-

menseigenen Arbeitskräften. Externe Anpassungsfähigkeit kann durch die Interaktion mit anderen Systemen erzielt werden, wie z. B. durch Kooperationen und Unternehmensnetzwerke, die den Austausch von Lieferanten oder die Diversifikation von Produkten und Märkten erlauben [GP00, ENS07].

Tabelle 3.1 zeigt die Eigenschaften der genannten allgemeinen Dimensionen als Übersicht. Für eine objektive Einschätzung von Flexibilität können anwendungsabhängige Mechanismen zur Kosten-Nutzen-Analyse eingesetzt werden, wobei in der Regel die Kosten für nötige Anpassungen gut zu erfassen sind, jedoch der Nutzen der Flexibilität kaum zu quantifizieren ist, sofern dieser nicht selbst in der Reduktion von Kosten liegt [KB04a, VMSS07]. Basierend darauf lassen sich jedoch für einige Dimensionen grundlegende Metriken ableiten, welche eine objektive Bewertung der Flexibilität eines Systems ermöglichen. Dabei können in zeitlicher Hinsicht sowohl die *Reaktionszeit* eines Systems als auch die *Effizienz* der Anpassung betrachtet werden [APK+95, GP00]. Bei der Effizienz wird es als wesentliches Kriterium angesehen, dass Anpassungen vorgenommen werden können, ohne dass die Leistung des Systems während der Anpassung wesentlich beeinträchtigt wird. In Hinsicht auf den Anpassungsspielraum wird die *Vielfältigkeit* an Möglichkeiten bewertet, die dem System zur Verfügung stehen, um auf erwartete Situationen reagieren zu können. Analog dazu dient das Kriterium der *Robustheit* als Grad der Anpassungsfähigkeit an unerwartete Situationen [Car89, GP00]. Die Fähigkeit zur Pro- bzw. Reaktivität stellt bereits ein eigenes Qualitätskriterium dar, während die Wirkungsrichtung der Anpassungsfähigkeit in der Regel gut durch den Aufwand an (monetären) Kosten abgeschätzt werden kann (z. B. durch den Mehraufwand für das Vorhalten zusätzlicher interner bzw. den Einkauf externer Leistungen). Objekt- und Verhaltensdimension umfassen lediglich nominal skalierende Merkmale, so dass hier keine grundlegenden Metrik zum Größenvergleich angegeben ist.

In konkreter Hinsicht auf die *Flexibilität von Informationssystemen* liefern die Autoren GEBAUER und SCHOBER [GS05, GS06] zusätzlich zwei weitere Flexibilitätsformen, welche ebenfalls den Aspekt des Änderungsaufwands hervorheben. Hierfür definieren sie den Begriff der *bedeutenden Veränderung* als Modifikation eines Informationssystems, welche zu einer Neuinstallation des Systems (inklusive einer vorhergehenden Deinstallation) und zu erneuten Tests des Systems führt. Eine Aktivierung vorinstallierter Parameter, die nur eine geringe Unterbrechung der Verfügbarkeit verursacht, stellt nach ihrem Modell hingegen keine bedeutende Änderung dar [GS05, ENS07]:

- **Nutzungsflexibilität**: Nutzungsflexibilität umfasst den Umfang der Anforderungen, die durch ein Informationssystem unterstützt werden, ohne dass bedeutende Änderungen am Informationssystem vorgenommen werden müssen. Die Nutzungsflexibilität ist von den folgenden vier Faktoren abhängig, wobei jeweils nicht nur die Zahl der angebotenen

Dimension	Anpassungsfähigkeit	Metriken
Objekt	fachlich (Ziele)	
	organisatorisch, personell, technisch (Mittel)	
Zeit	kurzfristig (operativ)	Reaktionszeit, Effizienz
	mittelfristig (taktisch)	Reaktionszeit, Effizienz
	langfristig (strategisch)	Reaktionszeit, Effizienz
Anpassungsspielraum	vorhersehbare Ereignisse	Vielfältigkeit
	unvorhersehbare Ereignisse	Robustheit
Verhalten	proaktiv	
	reaktiv	
Wirkung	intern	(monetäre) Kosten
	extern	(monetäre) Kosten

Tabelle 3.1: Dimensionen der Flexibilität (tlw. nach (GP00, KB04a))

Möglichkeiten, sondern auch deren Verwaltung und Wartung für eine Beurteilung der Flexibilität des Gesamtsystems in Betracht gezogen werden müssen [GS05, ENS07]:

- ▷ **Funktionalität**: Die Summe der Funktionen, die von einem System bereitgestellt werden.
- ▷ **Datenverwaltung**: Die Bandbreite an unterstützten Kategorien und Entitäten sowie die Anzahl an möglichen Analysen und Berichten (ähnlich in [SWG+07]).
- ▷ **Benutzungsschnittstellen**: Die Summe der Zugriffsmöglichkeiten eines menschlichen Benutzers zur Interaktion mit dem System.
- ▷ **Verarbeitungskapazität**: Die Anzahl der Transaktionen und Nutzeranfragen, die ein System gleichzeitig ohne bedeutenden Verlust der Leistungsfähigkeit verarbeiten kann.

- ► **Änderungsflexibilität**: Änderungsflexibilität erlaubt Änderungen, Aktualisierungen und Erweiterungen, die erst *nach* der (ersten) Implementierung eines Informationssystems durchgeführt werden. Hierbei kommt es in der Regel zu den oben genannten *bedeutenden Änderungen*, ohne jedoch das Informationssystem dabei völlig zu ersetzen, z. B. durch eine Überarbeitung, Ergänzung oder Reduktion von Quellcode. Diese Änderungen können im Allgemeinen nicht zur Laufzeit des Systems vorgenommen werden. Eine Bewertung der Änderungsflexibilität ist zum Beispiel durch den Zeitaufwand oder durch monetäre Kosten der Anpassung möglich.

Die dargestellte Unterteilung nach *Nutzungs-* und *Änderungsflexibilität* von Informationssystemen weist deutliche Parallelen zur Bestands- und Entwicklungsflexibilität der allgemeineren Dimension *Zeit* und der Dimension des *Anpassungsspielraums* auf, welche durch den Umgang mit vorhersehbaren und

unvorhersehbaren Ereignissen gekennzeichnet ist. Durch eine Erhöhung der (technisch realisierten) Handlungsalternativen kann dabei die Nutzungsflexibilität eines Informationssystems im Umgang mit vorhersehbaren Ereignissen gesteigert werden, während der Umgang mit unvorhergesehenen Ereignissen in vielen Fällen durch die Schaffung von zusätzlichen Abstraktionsebenen und durch größere Freiheitsgrade für den menschlichen Benutzer verbessert werden kann [SMR+08]. Ist der zur Entwicklungszeit geschaffene Anpassungsspielraum jedoch erschöpft, sind Änderungen erforderlich, die eine Reimplementierung des Informationssystems mit entsprechender Neuinstallation bedürfen. Die Flexibilität kann dabei in beiden Fällen sowohl in fachlicher Hinsicht (d. h. in Bezug auf die Ziele) als auch in organisatorischer bzw. technischer Hinsicht (d. h. in Bezug auf die Mittel) betrachtet werden. Im Folgenden wird daher, basierend auf dieser grundlegenden Klassifikation, der Stand der Forschung in Hinblick auf die Arten von Flexibilität prozessorientierter Anwendungen als spezielle Ausprägung von Informations- und Kommunikationssystemen dargestellt.

3.3 Arten der Flexibilität prozessorientierter Anwendungen

Wie im vorhergehenden Kapitel gezeigt wurde, ist durch die explizite Formulierung und Darstellung von Prozessmodellen die fachliche Logik der prozessorientierten komplexen Anwendung von den eingebundenen Ressourcen wie den integrierten Softwareanwendungen, Personen und Maschinen hinreichend entkoppelt. Der Prozessablauf kann daher in der Regel angepasst werden, ohne den Programmcode integrierter Anwendungen oder die Organisationsstruktur, in welcher Personen oder Maschinen angeordnet sind, verändern zu müssen [Wes07, Ses10]. Dies umfasst neben der durch den Prozesslebenszyklus definierten kontinuierlichen Verbesserung des Prozesses vor allem dessen Fähigkeit zur Ausnahmebehandlung [Sof05]. So können im Rahmen des Prozesslebenszyklus auf der fachlichen Ebene die Ausführungsreihenfolge der Aktivitäten modifiziert, Bedingungskonstrukte und Aktivitäten geändert, entfernt oder hinzugefügt und die Zuordnung von Ressourcen erneuert werden. Entsprechende Änderungen dieser fachlichen Ebene ziehen im Idealfall automatisch ein entsprechend verändertes Verhalten der genutzten Informations- und Kommunikationssysteme nach sich, ohne dass diese selbst expliziter Änderungen bedürfen [Ses10], so dass eine schnelle Reaktion auf notwendige Änderungen der fachlichen Prozesslogik ermöglicht wird [Wes07]. Umgekehrt können im Idealfall auf technischer Ebene Ressourcen umorganisiert, modifiziert, hinzugefügt oder entfernt werden, ohne dass dies eine Änderung der bestehenden, übergeordneten prozessorientierten Anwendungen erforderlich macht.

Auf Basis dieser grundlegenden Eigenschaften kann nach WESKE [Wes07] und in Hinblick auf die Objektdimension (vgl. Abschnitt 3.2) bereits eine erste wesentliche Unterscheidung der Flexibilität prozessorientierter Anwendungen getroffen werden:

- **Flexibilität in fachlicher Hinsicht** erlaubt Anpassungen der prozessorientierten Anwendung auf inhaltlicher Ebene (*Was?-Perspektive*). Bei diesen Anpassungen handelt es sich in der Regel um die Durchführung von geeigneten Aktionen, die durch externe Ereignisse ausgelöst werden, welche den entsprechenden Vorgang in der Realwelt beeinflussen. Beispiele hierfür sind Gesetzesänderungen, eine geänderte Nachfragesituation bei Produkten oder kurzfristig geänderte Kundenwünsche zur Individualisierung einer Leistung. Die notwendigen Anpassungen betreffen daher die Prozesslogik der vorliegenden Prozessmodelle und/oder -instanzen in einer inhaltlichen Art und Weise und führen in der Regel zu einem *fachlich qualitativ oder quantitativ anderen Ergebnis* der Prozessausführung. Wird zum Beispiel aufgrund einer erhöhten Zahl von Betrugsvorfällen ein zusätzlicher Kontrollschritt in einen Bestellvorgang eingearbeitet, so wird durch den resultierenden Ausschluss unseriöser Bestellungen das fachliche Gesamtergebnis der Prozessausführung verändert. Das wesentliche fachliche Ziel des Prozesses („Bestellung“) bleibt jedoch durch die Anpassung erhalten. Es handelt sich somit also nicht um einen unabhängigen neu erstellten Prozess für einen anderen Anwendungsfall, sondern um eine inhaltliche Anpassung eines bestehenden Prozesses.

- **Flexibilität in technischer Hinsicht** erlaubt eine Anpassung der prozessorientierten Anwendung auf der Ebene seiner Ausführung (*Wie?-Perspektive*). Bei diesen Anpassungen handelt es sich in der Regel um die Durchführung von geeigneten Aktionen, die durch interne Ereignisse der organisatorischen oder technischen Infrastruktur ausgelöst werden, wie zum Beispiel durch die Nicht-Verfügbarkeit von aktuell benötigten Ressourcen. Die Bewältigung derartiger Situationen erfordert im Idealfall keine Änderungen an der Prozesslogik, sondern eine geeignete organisatorische oder technische Anpassung der Ausführung, um das durch die Prozesslogik festgelegte Ergebnis trotz des Ereignisses dennoch in der fachlich vorgegebenen Art und Weise erzielen zu können. Dabei können jedoch durchaus technische nicht-funktionale Aspekte der Prozessausführung, wie zum Beispiel Zeit oder Kosten, beeinflusst werden. Ein typisches Beispiel ist die Zuweisung einer Aktivität an einen externen Mitarbeiter, falls organisationsintern temporär kein geeignetes Personal zur Verfügung steht. Diese Art der Anpassung kann ggf. ein *technisch qualitativ oder quantitativ anderes Ergebnis* der Prozessausführung verursachen. Zum Beispiel können die Ausführungszeit des Prozesses oder die Kosten der Prozessausführung beeinflusst werden. Das fachliche Ergebnis der Prozessausführung bleibt jedoch erhalten.

Beide Perspektiven sind durch die Entwicklungsmethodik prozessorientierter Anwendungen in der Regel relativ strikt voneinander getrennt. Das bedeutet einerseits, dass in der Prozesslogik keine Anweisungen vorkommen sollen, welche Anpassungen auf technisch-organisatorischer Ebene ermöglichen (z. B.

zusätzliche Verzweigungen, welche überprüfen, ob Mitarbeiter oder Softwaresysteme zur Ausführung der anstehenden Aktivitäten verfügbar sind). Andererseits sollen Anpassungen auf fachlicher Ebene der übergeordneten Prozessstruktur nicht an der technischen Infrastruktur geschehen, da dies deren Wiederverwendbarkeit für andere Anwendungsfälle gefährdet (vgl. [Mül05, Wes07, WML09b]). Demnach können beide Perspektiven der Flexibilität im Kontext prozessorientierter Anwendungen weitestgehend getrennt voneinander betrachtet und in Hinblick auf die oben genannten Eigenschaften zur Bewertung der Flexibilität untersucht werden. Im Folgenden wird zunächst die fachliche Perspektive zusammengefasst und daran anschließend der Stand der Forschung entlang der technischen Perspektive analysiert.

3.3.1 Fachliche Perspektive

Der Prozesslebenszyklus als Phasenmodell für die Entwicklung von (Geschäfts-)Prozessen zeigt bereits, dass prozessorientierte Anwendungen einer besonderen inhaltlichen Dynamik unterliegen. Dabei stellt insbesondere die Fähigkeit, den Kontrollfluss eines Prozesses ohne Programmierung ändern und einzelne Funktionalitäten austauschen zu können, ohne dass andere Inhalte betroffen werden, einen großen Vorteil dar. Oft ist diese einzelne Art der Flexibilität jedoch nicht ausreichend, um alle Forderungen nach Anpassungsfähigkeit zu erfüllen. Insbesondere der Wunsch nach einer schnelleren Reaktionszeit und die Behandlung von Einzelfällen im Rahmen der genannten *flexiblen Automatisierung* machen eine detailliertere Auseinandersetzung mit den Ursachen und den Möglichkeiten inhaltlicher Anpassungen prozessorientierter Anwendungen erforderlich.

Inhaltliche Anpassungen einer prozessorientierten Anwendung können Objekte verschiedener Perspektiven betreffen [RSS06a]:

- **Funktionale Perspektive:** Das Ziel der Prozessausführung wird verändert.
- **Operationale Perspektive:** Einzelne Aktivitäten werden ausgetauscht, gelöscht oder hinzugefügt.
- **Verhaltensperspektive:** Die Reihenfolge und Bedingungen, unter denen Aktivitäten ausgeführt werden, wird verändert.
- **Informationelle Perspektive:** Informationen, welche zwischen Aktivitäten ausgetauscht werden, werden verändert, entfernt oder ergänzt.
- **Organisatorische Perspektive:** Die inhaltliche Zuordnung von Ressourcen zu Aktivitäten wird modifiziert.

Für die Bewertung der Flexibilität prozessorientierter Anwendungen werden insbesondere die Dimensionen *Zeit*, *Anpassungsspielraum* und *Verhalten* als zentrale Aspekte angesehen (vgl. Abschnitt 3.2). Fachliche Situationen,

die zur Entwicklungszeit des Prozesses voraussehbar sind und deren Eintreten als wahrscheinlich angesehen wird, können dabei bereits im Prozessmodell berücksichtigt und entsprechend modelliert werden, was dem System ein proaktives Verhalten ermöglicht. Unvorhersehbare oder nicht beschreibbare Situationen können hingegen erst bei ihrem Auftreten reaktiv behandelt werden [SMR+08]. Auf Basis dieser Erkenntnisse werden in der Literatur in Bezug auf prozessorientierte Anwendungen zwei Arten von Flexibilität unterschieden: *A-priori-Flexibilität* und *A-posteriori-Flexibilität* [HHJ+99, Joe00, Nur08].

A-priori-Flexibilität *A-priori-Flexibilität* (auch: *Flexibility by Design* [SMR+08]) ist vom Grad der Determiniertheit des Prozessmodells abhängig und umfasst alle Maßnahmen zur Entwicklungszeit des Prozesses, um nachträgliche Änderungen zu vermeiden [Joe00]. Die Autoren HEINL et al. [HHJ+99] unterscheiden hinsichtlich der A-priori-Flexibilität danach, ob die Flexibilität durch den *Abstraktionsgrad* des Prozessmodells gegeben ist (d. h. dass Anpassungsfähigkeit durch die Vermeidung einer zu detaillierten Prozessbeschreibung geschaffen wird [All05]) oder ob die Flexibilität aus der expliziten Modellierung möglichst vieler *Varianten* der Ausführung resultiert, aus der eine zur aktuellen Situation passende Variante gewählt werden kann (z. B. durch die Modellierung möglichst vieler alternativer Pfade oder umfangreicher Ausnahmebehandlungen) [HHJ+99, Nur08, SMR+08]. Da die Modellierung aller denkbarer Prozessverläufe jedoch in der Regel zu sehr komplexen und unverständlichen Prozessmodellen führt und es zudem kaum möglich ist, alle Varianten a-priori zu determinieren, ist diese Strategie in ihrer Anwendbarkeit eingeschränkt [HHJ+99, SMR+08]. Die Offenhaltung von Freiheitsgraden für die Prozessdurchführung besitzt hingegen den entscheidenden Nachteil, dass aufgrund der fehlenden Strukturiertheit eine automatische Ausführung des Prozesses erschwert wird (vgl. Abschnitt 2.2). Der hybride Ansatz der *späten Modellierung* (*Late Modelling*) [Nur08] vereint daher beide Strategien, indem zur Entwicklungszeit kritische Teile des Prozesse nur als Black Box modelliert, und zur Ausführungszeit des Prozesses entsprechende Details der Modellierung nachgeliefert werden [HHJ+99]. Verzögerungen auch längerer Art sind hierbei jedoch nicht auszuschließen, insbesondere bei unerwarteten Situationen, für die erst eine geeignete Ausnahmebehandlung entwickelt werden muss. Abbildung 3.1 zeigt eine graphische Veranschaulichung der genannten Varianten der A-priori-Flexibilität.

A-posteriori-Flexibilität Die Problematik, eine für alle Situationen ausreichende A-priori-Flexibilität zu erlangen, kann entschärft werden, indem zugunsten der Automatisierung das Prozessmodell hinreichend stark strukturiert wird und bei Bedarf eine nachträgliche Änderung daran vorgenommen wird. Die Fähigkeit, solche nachträglichen Änderungen zu erlauben, wird als *A-posteriori-Flexibilität* (oder auch als *Flexibility by Adaption* [HHJ+99,

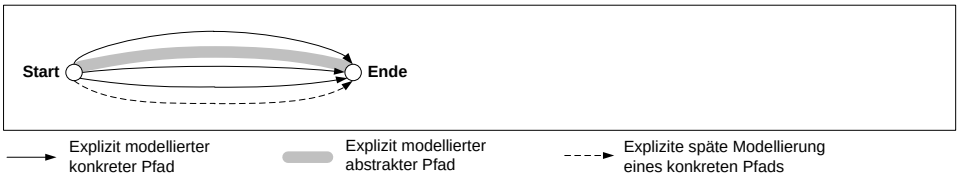


Abbildung 3.1: A-priori-Flexibilität (nach (HHJ+99))

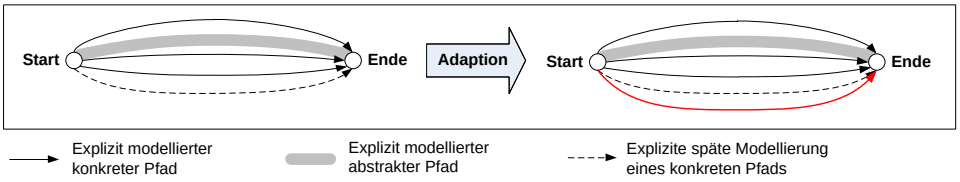


Abbildung 3.2: A-posteriori-Flexibilität (nach (HHJ+99))

Nur08]) bezeichnet [Joe00]. Insbesondere NURCAN [Nur08] gibt jedoch zu bedenken, dass dieser Ansatz nicht wirklich als „flexibel“ bezeichnet werden könne, sondern lediglich als „adaptiv“, denn aus den Anpassungen resultiere wieder nur eine (relativ) statische Beschreibung, welche einer erneut veränderten Situation ggf. wieder nicht standhalten könne. Abbildung 3.2 verdeutlicht den Unterschied und das Zusammenspiel von A-priori- und A-posteriori-Flexibilität.

Hinsichtlich der Dimension *Zeit* kann insbesondere bei der A-posteriori-Flexibilität unterschieden werden, ob es sich bei der erforderlichen Änderung um eine *kurzfristige* Anpassung handelt, bei der nur temporär von der üblichen Art der Ausführung des Prozesses abgewichen wird, oder ob es sich um Änderungen der *langfristigen* Rahmenbedingungen handelt, welche eine dauerhafte Weiterentwicklung der bisherigen Prozessausführung erforderlich machen [Sof05]. Abhängig von den Ursachen und deren zeitlicher Ausdehnung kann der Abstraktionsgrad des Prozesses identifiziert werden, welcher von den Änderungen betroffen wird:

- Langfristige Anpassungen werden am *Prozessmodell* vorgenommen und entsprechen der oben genannten kontinuierlichen Evolution des Prozesses [RSS06a, SMR+08].

- Kurzfristige Anpassungen werden notwendig, wenn die beabsichtigten Ziele durch eine Ausnahmesituation nicht auf die übliche Weise erreicht werden können. Sie betreffen den Prozess in der Regel nur temporär. Es muss daher entschieden werden, ob ein Prozessmodell hierfür zeitweilig geändert werden soll oder die Anpassung auf der Ebene einzelner *Prozessinstanzen* vorgenommen wird [RSS06a, PRR08, SMR+08].

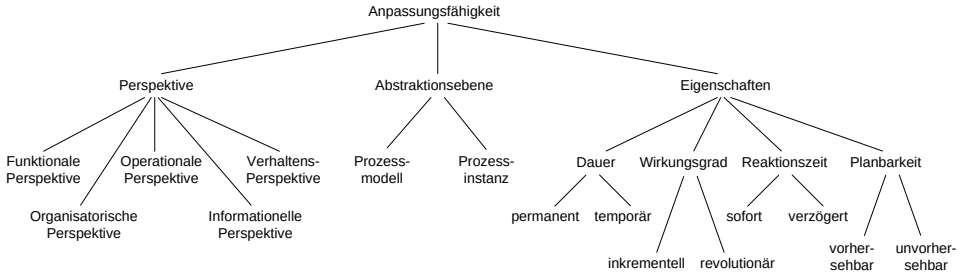


Abbildung 3.3: Taxonomie der Anpassungsfähigkeit von Prozessen (nach (RSS06a))

Bei der Anpassung eines Prozessmodells wird die Reaktionszeit in Bezug auf die Auswirkungen der Änderung auf die Realwelt davon beeinflusst, wie sich die vorgenommenen Änderungen auf die Prozessinstanzen auswirken. Bei einer *verzögerten Ausbreitung* werden die Anpassungen nur auf in Zukunft instantiierte Prozesse angewendet [RSS06a]. In der Regel müssen hierfür zunächst alle laufenden Prozessinstanzen beendet werden, um die Co-Existenz verschiedener Prozessversionen in einem System zu vermeiden [RSS06a], wodurch sich abhängig von der Dauer der Prozesse eine relativ starke Verzögerung ergeben kann. Es existieren jedoch auch Ansätze zur Prozessversionierung, welche die Verwaltung verschiedener Prozessinstanzen erlauben (vgl. z. B. [KG99, SO99]). Bei einer *sofortigen Ausbreitung* der Änderungen müssen auch laufende Prozessinstanzen des geänderten Prozessmodells angepasst werden [RSS06a]. Dies ist oft problematisch, da laufende Prozessinstanzen in der Regel keine Referenz auf ihr Prozessmodell besitzen. Um eine konsistente Prozessausführung zu gewährleisten, können entweder nur Prozessinstanzen angepasst werden, deren Kontrollfluss noch nicht bis zur Änderung fortgeschritten ist, oder die betreffenden Prozessinstanzen müssen zumindest teilweise zurückgesetzt und neu gestartet werden (vgl. [CRW98, RRD04, Wes07, SMR+08]). In diesem Fall geht bereits geleistete Arbeit verloren und muss ggf. wiederholt werden, was erneut zu Verzögerungen der Prozessausführung führt.

Abbildung 3.3 zeigt eine Zusammenfassung der inhaltlichen Anpassungsfähigkeit von prozessorientierten Anwendungen. Abhängig von der Dynamik des Anwendungskontextes können dabei mehr oder weniger starke Änderungen von unterschiedlicher Zeitdauer und Spontanität Auswirkungen auf verschiedene Perspektiven von Prozessmodellen und/oder -instanzen haben. Für die kontinuierliche Weiterentwicklung und die Ausnahmebehandlung stark strukturierter Prozesse bestehen mehr oder weniger flexible Möglichkeiten zu deren (gelegentlicher) Anpassung. Mit einer zunehmenden Dynamik des Anwendungskontextes wird jedoch trotz der bestehenden Möglichkeiten zur Anpassung eine Automatisierung des Prozesses erschwert. Schwach strukturierte oder fallbezogene Prozesse mit einem hohen individuellen oder kreativen Anteil erfordern, dass der Ablauf spontan nach den jeweiligen Erfordernissen gestaltet wird (vgl. Abschnitt 2.2). Herkömmliche Ent-

wicklungsmethodiken bieten für diese Klassen von Prozessen zu wenig Flexibilität, da insbesondere der Modellierungsprozess hierfür zu aufwendig ist [HHJ+99, All05]. Für eine Unterstützung dieser sogenannten *flexiblen Prozesse* werden daher andere Möglichkeiten der Unterstützung durch Informations- und Kommunikationstechnologie angewendet. Eine Möglichkeit zur Flexibilisierung der Anwendungslogik besteht in der Abbildung des Anwendungsfalls mit anderen Entwicklungsansätzen, welche teilweise oder gänzlich auf starre Prozeduren und Schemata verzichten und stattdessen die *Zielorientierung* als zentrales Konzept betrachten (vgl. [NC04, BPJ+10]). Ad-hoc Prozesse und Projekte können schließlich durch Ansätze aus dem Bereich der *Computer Supported Cooperative Work* (*CSCW*) unterstützt werden, bei der Informations- und Kommunikationswerkzeuge zur rechnergestützten Gruppenarbeit bereitgestellt werden (*Groupware*) [All05]. In dieser Arbeit wird vor dem Hintergrund der angestrebten Automatisierung im Folgenden jedoch insbesondere die Klasse der strukturierten Prozesse und ihre (technische) Ausführung bei gegebenem fachlichen Inhalt betrachtet.

3.3.2 Technische Perspektive

Als Basis für schnelle Prozessänderungen ist vor allem die Flexibilität der genutzten Informations- und Kommunikationssysteme ein wichtiger Aspekt. Die explizite Darstellung des Prozessmodells und dessen Abstraktion von Informations- und Kommunikationssystemen erlauben es prinzipiell, den Kontrollfluss ohne Programmierung zu ändern oder einzelne Funktionalitäten auszutauschen, ohne dass der Rest des Systems betroffen wird [All05]. Doch nicht nur für inhaltliche Änderungen der Prozesslogik ist eine flexible technische Infrastruktur von Vorteil. Vor allem tragen diese Systeme dazu bei, dass die auf fachlicher Ebene definierte Anwendungslogik unter Einhaltung oder Optimierung nicht-funktionaler Rahmenbedingungen transparent und robust ausgeführt werden kann. Kritische Situationen müssen hierfür erkannt und in angemessener Weise behandelt werden, um das fachlich geforderte Ergebnis zu erbringen.

Neben der *expliziten Modellierung* des Prozesses und somit seiner Abstraktion von unterliegenden Informations- und Kommunikationssystemen ist nach WESKE [Wes07] die hierfür notwendige (technische) Flexibilität insbesondere durch eine explizite Darstellung der Organisationsstruktur der betreffenden Organisation und der Abstraktion der darin enthaltenen Individuen zu Rollen zu erreichen. In Bezug auf menschliche Akteure hat dies den Vorteil, dass sich insbesondere personelle Veränderungen nicht auf die Ausführbarkeit der Prozesse auswirken. In Hinblick auf Maschinen und Softwaresysteme ergibt sich eine erleichterte Austauschbarkeit bei Ausfällen oder hoher Auslastung. In Abhängigkeit von der Dynamik der Systemumgebung können hinsichtlich der Dimensionen *Zeit* und *Anpassungsspielraum* verschiedene Varianten identifiziert werden [SLI08]:

- Bindung einer Aktivität an eine Ressource zur Entwicklungszeit des Prozesses
- Dynamische Bindung einer Ressource zur Laufzeit, wobei dies variabel zwischen der Initiierung des Prozesses und dem Ausführungszeitpunkt der Aktivität geschehen kann (vgl. Abschnitt 2.7.3).

Eine möglichst *frühe Bindung* erleichtert dabei den Umgang mit vorhersehbaren Ereignissen (wie zum Beispiel dem Urlaub eines Mitarbeiters oder die geplante Wartung einer Maschine) und ist somit für die Einsatzplanung von Ressourcen von hoher Bedeutung. Eine *späte Bindung* erleichtert hingegen den Umgang mit nicht vorhersehbaren Ereignissen, wie zum Beispiel Krankheit eines Mitarbeiters oder Ausfall einer Maschine [RHE05]. Ein hybrider Ansatz erlaubt die Bereitstellung von sekundären Ressourcen als *Ersatz*, falls ursprünglich zugeordnete Ressourcen nicht verfügbar sind (*Acquisition of Substitutes* [SLI08]). Dabei können die Ersatzressourcen sowohl bereits zur Entwicklungszeit vorgegeben oder zur Laufzeit des Prozesses ausgewählt werden [SLI08].

In Hinblick auf die *Wirkungsdimension* wird nicht nur die flexible Nutzung organisationsinterner Ressourcen, sondern aufgrund zunehmender Vernetzung auch eine organisationsübergreifende, gemeinsame Nutzung von Ressourcen als großes Flexibilitätspotential gesehen [GP00, SD03, Wes07, Nur08]. Im betriebswirtschaftlichen Kontext können beteiligte Unternehmen in einer zwischenbetrieblichen Kooperation schnell und flexibel auf die ungenutzten Ressourcen ihrer Kooperationspartner zurückgreifen, so dass es nicht notwendig ist, die für die Flexibilität benötigte Quantität an Ressourcen selbst vorzuhalten [KB04a]. Die Nutzung externer Ressourcen verspricht daher eine hohe Ressourceneffizienz, während die Bereitstellung interner Ressourcen die Sicherheit einer exklusiven Nutzung beinhaltet und damit die relative Verfügbarkeit der Ressource erhöht. Die Flexibilität, spontan über die Verwendung von Ressourcen zu entscheiden, ist daher zu einem gewissen Grad abhängig von den infrastrukturellen Verbindungen zwischen Organisationen, d. h. von den Informations- und Kommunikationssystemen, die für die Umsetzung solcher Kooperationen zur Verfügung stehen [GP00, SD03, Nur08].

Auf Basis dieser Überlegungen kann analog zu der fachlichen Perspektive daher auch für die technische Perspektive hinsichtlich der Flexibilität prozessorientierter Anwendungen eine Einteilung in *A-priori-Flexibilität* und *A-posteriori-Flexibilität* vorgenommen werden:

A-priori-Flexibilität A-priori-Flexibilität in technischer Hinsicht umfasst alle Maßnahmen zur Entwicklungszeit des Prozesses oder der Ausführungsumgebung, um technische bzw. organisatorische Anpassungen zu ermöglichen, ohne dabei die prozessorientierte Anwendung (inhaltlich) zu ändern oder jeweils bedeutende Änderungen an den verwendeten Informationssystemen oder Organisationsstrukturen vorzunehmen. Die Flexibilität kann dabei durch die Abstraktion von konkreten Ressourcen (d. h. durch die

Entkopplung der Ressourcenauswahl von der Beschreibung des fachlichen Prozessmodells) gegeben sein oder durch die explizite Vorbereitung möglichst aller Varianten der Prozessausführung resultieren. Voraussetzungen zur Anwendung beider Strategien sind vor allem die Erschließung und Bereitstellung geeigneter (ggf. alternativer) Ressourcen, geeigneter Methoden zu deren zielgerichteten Auswahl sowie, zwecks Automatisierung, deren prinzipielle (syntaktische und semantische) Austauschbarkeit (vgl. [Wes07]).

A-posteriori-Flexibilität A-posteriori-Flexibilität in technischer Hinsicht erlaubt die Anpassung des Prozessmodells durch die Neuzuordnung von Ressourcen innerhalb der Prozessbeschreibung oder durch die nachträgliche Anpassung von Schnittstellen, das Ändern von Zugriffsrechten und Organisationsstrukturen sowie das Implementieren neuer oder das Ändern bestehender Softwarekomponenten zum Zweck der Prozessausführung. Analog zu der Betrachtung der A-posteriori-Flexibilität in fachlicher Hinsicht können diese Änderungen in Abhängigkeit der Zeitdimension und der Ursache der Änderung langfristig alle oder einige Prozessmodelle oder nur einzelne Prozessinstanzen betreffen.

In beiden Fällen stellt bei prozessorientierten Anwendungen die Möglichkeit zur Automatisierung der notwendigen Anpassungsvorgänge einen wichtigen Aspekt dar. Grundlegende organisatorische und technische Konzepte zum Auffinden, Einbinden und Austauschen von Ressourcen, zur Wahrnehmung von Umgebungsinformationen und Ereignissen sowie zu deren Verarbeitung und zur Ableitung von angemessenen Reaktionen werden daher in der Literatur als wichtige Einflussfaktoren der Flexibilität angesehen. Im den nachfolgenden Abschnitten dieses Kapitels werden daher grundlegende aktuelle Strategien und Paradigmen zur Realisierung flexibler prozessorientierter Anwendungen untersucht.

3.4 Flexibilität durch dienstorientierte Architekturen

Die Abbildung und Umsetzung verteilter Abläufe und die Ausnutzung von systemübergreifenden Wiederverwendungsmöglichkeiten werden durch die klassischen Entwicklungsansätze relativ wenig unterstützt. Insbesondere die Heterogenität der verschiedenen Systeme in Bezug auf Schnittstellen und Technologien erfordert hiermit oftmals eine sehr aufwendige *manuelle* Integration der benötigten Anwendungen mit beträchtlichem Entwicklungs- und Konfigurationsaufwand [All05]. Die Unzulänglichkeiten der bestehenden Ansätze haben daher zu einem Paradigmenwechsel im Bereich der Softwareentwicklung geführt, welcher nun die automatisierbare Integration von Softwarekomponenten stärker in den Vordergrund stellt [Szy02, MR09].

Eine Basis für ein automatisiertes Auffinden, Einbinden und Austauschen von Komponenten bietet dazu das Konzept der *Dienstorientierung*, welches

	2010	2009	2008	2007
Steigerung der Flexibilität	29%	27%	23%	28%
Optimierung der Prozesse	21%	21%	-	-
Senkung Time-to-Market	16%	14%	15%	-
Steigerung des Innovationsgrades	10%	8%	9%	9%
Steigerung der Kundenzufriedenheit	5%	3%	13%	13%
Senkung der Kosten	5%	4%	11%	15%
Steigerung der Produktivität	2%	7%	14%	13%

Tabelle 3.2: Ziele der Einführung einer dienstorientierten Architektur (Mar10)

die Kapselung von Anwendungsfunktionalität als Dienst hinter einer selbstbeschreibenden Schnittstelle und das Verbergen technologieabhängiger Implementierungsdetails propagiert [Pap03, MR09]. Zu den wichtigsten strategischen Zielen der Einführung einer auf Diensten basierenden Infrastruktur im Unternehmensumfeld zählen nach einer aktuellen Studie der Technischen Universität Darmstadt [Mar10] insbesondere die *Steigerung der Flexibilität* und die *Optimierung von Prozessen* (vgl. Tabelle 3.2). Ein Zusammenspiel von Prozessen und Diensten stellt demnach eine aktuelle Grundlage für die Umsetzung prozessorientierter Anwendungen dar, um vorgegebene lokale oder verteilte Abläufe der realen Welt mittels Informations- und Kommunikationstechnologie unterstützen zu können. Die Bedeutung von Diensten als technische Infrastruktur für die Flexibilisierung prozessorientierter Anwendungen, die Anforderungen und Möglichkeiten zur Nutzung verteilter Ressourcen sowie die Abbildung und Umsetzung verteilt ausgeführter Abläufe auf Basis von dienstorientierten Architekturen werden daher im Folgenden verdeutlicht.

3.4.1 Dienstbegriff

Eine *dienstorientierte Architektur* (engl. *Service-Oriented Architecture* bzw. kurz *SOA*) ist ein softwaretechnisches Architekturmuster für die Strukturierung und Nutzung verteilter Funktionalitäten [MLM+06]. Hierbei kapselt ein elektronischer *Dienst* eine in sich abgeschlossene fachliche Funktionalität, die von einer autonomen Softwareeinheit über ein Netzwerk bereitgestellt wird [Pap03, Erl05, MLM+06]. Verschiedene Softwareeinheiten können sich dabei auf denselben oder auf unterschiedlichen Hardwaresystemen befinden und ggf. der Verantwortung und Kontrolle unterschiedlicher Organisationen unterliegen [MLM+06]. Durch eine Reihe wesentlicher Eigenschaften und Anforderungen wird eine weitestgehend einfache und flexible Nutzung dieser Dienste ermöglicht und auch deren Integration bzw. Komposition zu komplexen Anwendungen mit einem höheren Abstraktionsgrad unterstützt [Pap03, PvdH07, Mel10]:

- **Wohldefinierte Schnittstelle:** Die Nutzung eines Dienstes erfolgt über eine zumeist öffentliche Schnittstelle, welche selbstbeschreibend alle

Möglichkeiten, auf den Dienst zuzugreifen, in maschinenlesbarer Form spezifiziert (*Vertrag*).

- **Lose Kopplung:** Die konkrete Implementierung eines Dienstes bleibt nach außen vollständig verborgen und von der öffentlichen Schnittstelle getrennt (*Information Hiding*), so dass nur eine minimale Abhängigkeit zwischen Anbieter und (potentiellem) Nutzer des Dienstes entsteht.
- **Dynamisches Binden:** Auf Basis dieser losen Kopplung muss bei der Erstellung einer Anwendung nicht bekannt sein, welcher konkrete Dienst genutzt werden soll, noch muss dieser Dienst bereits vorhanden und/oder erreichbar sein. Somit ist es möglich, Dienste erst bei der Ausführung der Anwendung zu lokalisieren und einzubinden.
- **Plattformunabhängigkeit:** Aufbauend auf den drei zuvor beschriebenen Eigenschaften ist weder der Ort, an dem eine Dienstleistung erbracht wird, noch die Technologie zur Realisierung der angebotenen Funktionalität für die Nutzung des Dienstes relevant. Anbieter und Nutzer eines Dienstes können somit an unterschiedlichen Orten, durch unterschiedliche Programmiersprachen oder auf verschiedenen Plattformen realisiert sein. Man spricht in diesem Kontext auch von *Orts-* bzw. *Technologietransparenz* eines Dienstes.
- **Wiederverwendbarkeit:** Die häufige Verwendung einer einmalig implementierten Funktionalität in verschiedenen Kontexten erhöht die Auslastung eines Dienstes und ermöglicht somit ein vorteilhaftes Kosten-Nutzen-Verhältnis. Ein Dienst soll daher stets eine in sich abgeschlossene Funktionalität realisieren, welche übergreifend über mehrere Ausprägungen einer Problemstellung einsetzbar ist.
- **Interoperabilität:** Um eine systemübergreifende Wiederverwendbarkeit von Diensten zu fördern, sollten insbesondere die zur Dienstnutzung verwendeten Protokolle auf nicht-proprietären Technologien beruhen, welche eine möglichst breite Akzeptanz besitzen. Für eine nahtlose Zusammenarbeit ist in der Regel die Einhaltung gemeinsamer Standards notwendig.

Bei einer konkreten dienstorientierten Architektur handelt es sich somit um eine Infrastruktur, welche die Integration von Anwendungen ermöglicht, indem die Komplexität der einzelnen Anwendungen hinter standardisierten Schnittstellen verborgen wird. Die Modularität und die wohldefinierte Schnittstelle zum Aufruf der bereitgestellten Funktionalität erlauben dabei die Zusammensetzung einfacher Dienste zu erweiterten kombinierten Diensten. Man unterscheidet hierbei zwischen *Dienst-Aggregation* und *Dienst-Komposition*: *Dienst-Aggregation* ist die Kombination von verschiedenen Diensten, die zu einem bestimmten gemeinsamen Zweck zur Verfügung gestellt werden. Bei der *Dienst-Komposition* handelt es sich um die Integration von elementaren Diensten zu einem neuen Dienst, der gegenüber den einzelnen Diensten eine

Leistung höheren Wertes erbringt. Die Unterfunktionen des neuen Dienstes bleiben jedoch unabhängig und stellen weiterhin eigenständige Dienste dar [OEH02].

In Hinblick auf die Anwendungsintegration können vorhandene elementare oder komplexe Komponenten der technischen Infrastruktur (wie Legacy-Anwendungen, der Zugriff auf Datenbanken, Server und Websites sowie die Funktionalitäten etwaiger mobiler Komponenten) als Dienste gekapselt werden, so dass ihre Funktionalitäten anderen Organisationen oder Organisationseinheiten zur Verfügung gestellt und zu höheren Diensten zusammengesetzt werden können. Im Rahmen des Prozessmanagements stellen Dienste somit *Ressourcen* dar, welche zur Erfüllung von festgelegten Aktivitäten innerhalb einer prozessorientierten Anwendung aufgerufen und genutzt werden können. Dabei tritt das Prozessmanagementsystem zunächst als Dienstnutzer auf, welcher den Aufruf von den in der Prozessbeschreibung spezifizierten Anwendungssystemen vornimmt (vgl. Abbildung 2.17). Gleichzeitig entsteht durch die Komposition dieser Dienste als Prozess eine neue, höherwertige Funktionalität, welche wiederum selbst als Dienst bereitgestellt werden kann. Durch das Zusammensetzen von Diensten niedriger Granularität können so flexibel und unter Ermöglichung größtmöglicher Wiederverwendbarkeit Dienste höherer Granularität geschaffen werden. Die aus den oben genannten Eigenschaften resultierende Flexibilität wird somit für die prozessorientierte Anwendung in technischer Hinsicht nutzbar gemacht. Die Beziehungen und Interaktionen zwischen den beteiligten Akteuren einer dienstorientierten Architektur werden im Folgenden genauer betrachtet.

3.4.2 Rollen und Interaktionen

Um die oben genannten Eigenschaften für eine flexible Systemarchitektur zu nutzen und einen möglichst dynamischen und zugleich automatisierten Zugriff auf Dienste zu erlauben, steht neben der Ermöglichung einer nahtlosen *Maschine-zu-Maschine-Kommunikation* insbesondere die *Entkopplung* der beteiligten Rollen der klassischen Client-Server-Beziehung im Vordergrund [Pap03, Mel10]. In einer dienstorientierten Architektur können daher drei grundlegende, einander nicht ausschließende Rollen identifiziert werden:

- **Dienstanbieter:** Ein *Dienstanbieter* (*Service Provider*) stellt eine softwaretechnisch realisierte fachliche Funktionalität sowie eine entsprechende Beschreibung der Dienstschnittstelle zur Nutzung dieser Funktionalität bereit. Neben den erforderlichen technisch relevanten Details wie dem Zugriffspunkt, den verwendeten Kommunikationsprotokollen und der Art und Weise der auszutauschenden Nachrichten kann der Dienstanbieter optional eine semantische Beschreibung seiner Dienste sowie eine Beschreibung relevanter Qualitäts- und Sicherheitsaspekte bereithalten [EF03, Pap03]. Der Dienstanbieter ist zudem für den Betrieb und die Eigenschaften der Plattform verantwortlich, von welcher aus der Dienst angeboten wird [Mel10].

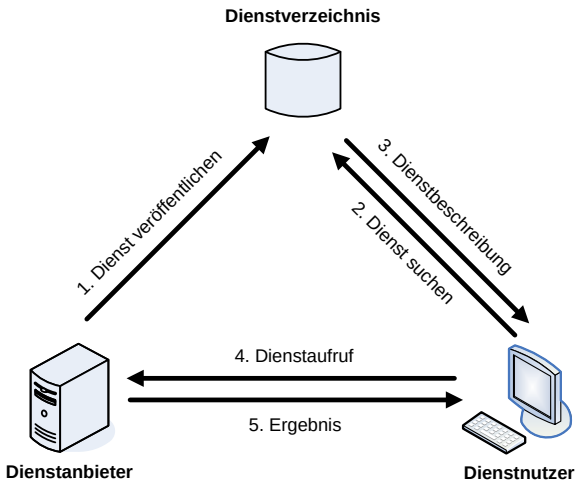


Abbildung 3.4: Rollen und Interaktionen in einer dienstorientierten Architektur (nach (Pap03))

- **Dienstnutzer** Ein *Dienstnutzer* (*Service Consumer*) stellt in einer dienstorientierten Architektur die nachfragende Partei dar. Durch die Abfrage einer Dienstbeschreibung der gesuchten Funktionalität ist dem Dienstnutzer der Aufruf des gewünschten Dienstes möglich. Eine Suchanfrage kann dabei neben funktionalen Aspekten auch nicht-funktionale Anforderungen enthalten, welche ggf. mit den entsprechenden Qualitätsangaben des Dienstanbieters abgeglichen werden müssen [EF03, Pap03, Mel10].

- **Verzeichnisdienst** Ein *Verzeichnisdienst* (*Service Registry*) stellt einen spezialisierten Dienst dar, welcher die Dienstbeschreibungen von registrierten Dienstanbietern verwaltet und auf Nachfrage von Dienstnutzern eine zur Suchanfrage passende Beschreibung herausgibt. Der Verzeichnisdienst verfügt idealerweise immer über die aktuellen Informationen bezüglich des Zugriffspunkts, der unterstützten Protokolle sowie über die aktuellen Qualitätseigenschaften der verwalteten Dienste [EF03, Mel10].

Abbildung 3.4 zeigt die genannten Rollen und ihre Beziehungen untereinander in dem sogenannten *SOA-Dreieck*. Hier nutzt ein Dienstanbieter einen Verzeichnisdienst zur Veröffentlichung der Beschreibung des von ihm angebotenen Dienstes (Schritt 1). Potentielle Dienstnutzer stellen eine Suchabfrage an den Verzeichnisdienst (Schritt 2). Bei Vorliegen einer geeigneten Beschreibung wird diese zurückgegeben (Schritt 3). Auf Basis der in der Dienstbeschreibung enthaltenen Informationen ist der Dienstnutzer in der Lage, den Dienst beim Dienstanbieter aufzurufen (Schritt 4) und erhält ggf. einen Rückgabewert, sofern der Dienst ein Ergebnis zurückliefert (Schritt 5) [Pap03, PvdH07].

Zur Realisierung der angebotenen Funktionalität kann ein Dienstanbieter gegenüber anderen Dienstanbietern auch als Dienstnutzer auftreten. Zudem kann er die Rolle eines (lokalen) Verzeichnisdienstes übernehmen und die

zur Nutzung seiner Dienste relevanten Beschreibungen zum Abruf durch potentielle Dienstnutzer selbst vorhalten. In diesem Fall ist keine Interaktion mit einem zentralen Dienstverzeichnis notwendig. Es wird aber vorausgesetzt, dass die erforderlichen Informationen über die Existenz des Dienstanbieters und den Zugriffspunkt der Dienstbeschreibung dem potentiellen Dienstnutzer schon im Voraus bekannt sind [Erl05, Mel10] oder auf andere Weise verfügbar gemacht werden. Die lose Kopplung und damit die Flexibilität ist bei diesem Vorgehen jedoch reduziert [Mel10], da der Dienstnutzer damit auf die bekannten Dienstanbieter beschränkt ist. Der einfache Austausch des Dienstanbieters ist somit nicht ohne weiteres möglich.

Um Dienste überhaupt automatisch auffinden und in eine Anwendung oder Dienstkomposition einbinden zu können, ist eine Beschränkung auf die Syntax zur Benutzung eines Dienstes in vielen Fällen nicht ausreichend. Vielmehr wird eine semantische Beschreibung benötigt, um zu definieren, welchen funktionalen Inhalt und welche Qualitätsmerkmale ein Dienst besitzt. Zur Realisierung semantischer Beschreibungen können unter anderem *Ontologien* eingesetzt werden (vgl. [DS05, DP06, FKZ08]). Zur Beschreibung von Faktoren, die über die reine Dienstbeschreibung hinausgehen und nicht-funktionale Aspekte oder Sicherheitsrichtlinien der Dienstleistung betreffen, können zudem sogenannte *Policies* oder auch *Service Level Agreements* (*SLAs*) formuliert werden. Eine *Policy* stellt im Rahmen einer dienstorientierten Architektur eine (nichtfunktionale) Anforderung dar, die von Seiten des Dienstnutzers durchgesetzt werden soll. Im Rahmen eines *Service-Level-Agreement* verpflichtet sich ein Dienstanbieter, ein verabredetes Maß an Qualität zu liefern und dafür eine Garantie auf die Dienstleistung zu gewähren [OEH02, ACKM04]. Der Auswahl eines geeigneten Dienstes durch Abgleich von Anforderungen und angebotenen Leistungen (*Matchmaking*) kann unter Umständen zudem eine manuelle oder automatisierte *Verhandlungsphase* vorausgehen (vgl. [NPS07, Wes07]).

3.4.3 Realisierung am Beispiel von Web Services

Um die in einer dienstorientierten Architektur grundlegenden abstrakten Konzepte der Dienstbeschreibung, der Kommunikation und des Verzeichnisses angemessen zu realisieren, stellt insbesondere der Einsatz gemeinsamer Standards einen zentralen Aspekt für die Interoperabilität der beteiligten Parteien dar. Hierzu wurden in der Vergangenheit zahlreiche Technologien und Middleware-Ansätze entwickelt, welche die genannten Eigenschaften und Anforderungen für die konkrete Umsetzung einer dienstorientierten Architektur voll oder zumindest teilweise erfüllen. Als Beispiele zu nennen sind hierbei vor allem die *Common Object Request Broker Architecture* (*CORBA*) der *Object Management Group* (*OMG*) [OMG08], das *Distributed Component Object Model* (*D-COM*) von Microsoft [Mic95, BK96] und die auf verschiedenen Internet-Standards aufbauende *Web-Service*-Architektur [BHM+04] (vgl. [ACKM04, Mel10]).

Die beiden erstgenannten Ansätzen haben in Hinblick auf die direkte Realisierung komplexer prozessorientierter Anwendungen in der Praxis nur untergeordnete Bedeutung. Während bei D-COM Abstriche bei der Plattformunabhängigkeit hingenommen werden müssen [Mel10], wird bei CORBA in der Regel eine hohe Komplexität, eine starke Verzahnung von Objekten und die Entwicklung vieler herstellereigener CORBA-Varianten kritisiert [Ham05, Hen08]. Im Gegensatz dazu wird den Web Services vor allem von Seiten der Industrie eine große Akzeptanz zugesprochen [ACKM04, Mar10]. Im Folgenden soll daher kurz auf die Implementierung einer dienstorientierten Architektur am Beispiel von grundlegenden Web-Service-Technologien eingegangen werden.

Dienstbeschreibung Im Vordergrund der beschriebenen Rollen und Interaktionen des SOA-Dreiecks (vgl. Abbildung 3.4) steht der Austausch von Dokumenten zur Dienstbeschreibung. Der beim *W3C* (*World Wide Web Consortium*) verwaltete Standard *WSDL* (*Web Service Description Language*) [W3C07b] spezifiziert hierzu eine formale Sprache zur Beschreibung und automatisierten Verarbeitung von Dienstschnittstellen. WSDL ist eine XML-basierte Sprache, welche die Definition von Datentypen als XML-Schema (vgl. [W3C04]) als Basis für die erlaubten ein- und ausgehenden Nachrichten, die Zuordnung von Nachrichten zu Operationen sowie die Definition von technischen Zugriffsinformationen definiert. Um ein dynamisches Binden zu erlauben, ist die abstrakte Beschreibung der Dienstfunktionen von der konkreten Beschreibung der austauschbaren technischen Zugriffsinformationen trennbar. Letztere legt fest, welche Protokolle für den Nachrichtenaustausch verwendet werden können (*Binding*) und unter welcher Adresse eine Dienstinstanz aufgerufen werden kann (*Service Endpoint*). Eine semantische Beschreibung der Dienstfunktionalität sowie eine Beschreibung von Qualitäts- und Sicherheitseigenschaften ist nicht Teil der WSDL-Spezifikation, kann aber durch entsprechende Zusatzspezifikationen ergänzt werden [W3C07b, ACKM04, Mel10].

Nachrichtenformat Für den Aufruf eines Dienstes sowie für die Rückgabe etwaiger Ergebnisse werden die in der Dienstbeschreibung spezifizierten Nachrichten ausgetauscht. Hierfür müssen sich die beteiligten Parteien darauf einigen, welche Sprache zum Austausch der Nachrichten verwendet wird, wie die Nachricht verpackt ist und welches Kommunikationsprotokoll zur Übertragung der Nachricht genutzt wird. Der W3C-Standard *SOAP* [W3C07a] legt dazu allgemein fest, wie eine Nachricht aufgebaut sein muss. Eine SOAP-Nachricht ist XML-basiert und besteht grob aus einem optionalen Kopfteil mit Verarbeitungsvorschriften (*Header*) und den zu übertragenden Nutzdaten (*Body*). Mögliche Inhalte des Headers werden üblicherweise in begleitenden Spezifikationen definiert und umfassen z. B. Informationen zur Authentifizierung und Autorisierung oder zum Routing und zur Identifizierung von Nachrichten. Der Body kann zudem zusätzlich zur Übertragung von Nutzdaten den Umgang mit Fehlermeldungen in einem vorgegebenen Format re-

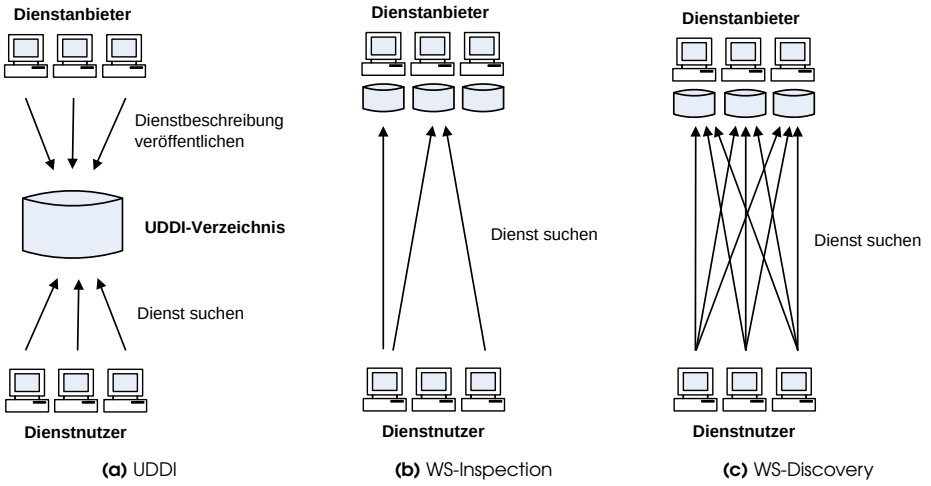


Abbildung 3.5: Vereinfachte Gegenüberstellung von UDDI, WS-Inspection und WS-Discovery (teilw. nach (Mel10))

geln (*SOAP-Fault*) [W3C07a]. Da die Beschreibung von Diensten nicht auf ein konkretes Format zur Nachrichtenübertragung festgelegt ist, ist es generell möglich, alternative Protokolle zum Nachrichtenaustausch einzubinden. Web Services sind somit nicht zwingend auf die Nutzung von SOAP festgelegt [ACKM04, Mel10].

Transportprotokoll Die zuvor in einem bestimmten Format spezifizierte Nachricht wird zur Übertragung in ein gewünschtes Transportprotokoll eingebettet. Die Wahl des Transportprotokolls ist von den Anforderungen an die Eigenschaften der zugrunde liegenden Infrastruktur abhängig. Im Rahmen der Kommunikation im Internet ist das zustandslose *Hypertext Transfer Protocol* (*HTTP*) [FGM$^+$99] das zur Zeit am häufigsten genutzte Protokoll. Entsprechend den Anforderungen können jedoch auch andere Protokolle gewählt werden, wie zum Beispiel das *Simple Mail Transfer Protocol* (*SMTP*) [Kle01] oder der *Java Messaging Service* (*JMS*) [HBS$^+$02] zur Unterstützung asynchroner Kommunikation bzw. zur Nutzung übertragungssicherer Warteschlangensysteme [ACKM04, Mel10].

Verzeichnisdienst Ein für die Web-Service-Architektur geeigneter Verzeichnisdienst muss mindestens in der Lage sein, die verwendeten Dokumente zur Schnittstellenbeschreibung zu verarbeiten, zu verwalten und eine Reihe von relevanten Operationen zur Veröffentlichung von Beschreibungen und zur strukturierten Suche nach Diensten anzubieten. Dazu ist es von Vorteil, wenn die Funktionalität des Verzeichnisdienstes selbst als Web Service angeboten und über entsprechend standardisierte Schnittstellen zugegriffen werden kann. Eine entsprechende durch die *Organization for the Advancement of Structured Information Standards (OASIS)* standardisierte Verzeichnisstruktur für die Verwaltung von Web-Service-Metadaten ist durch das Protokoll *UD-*

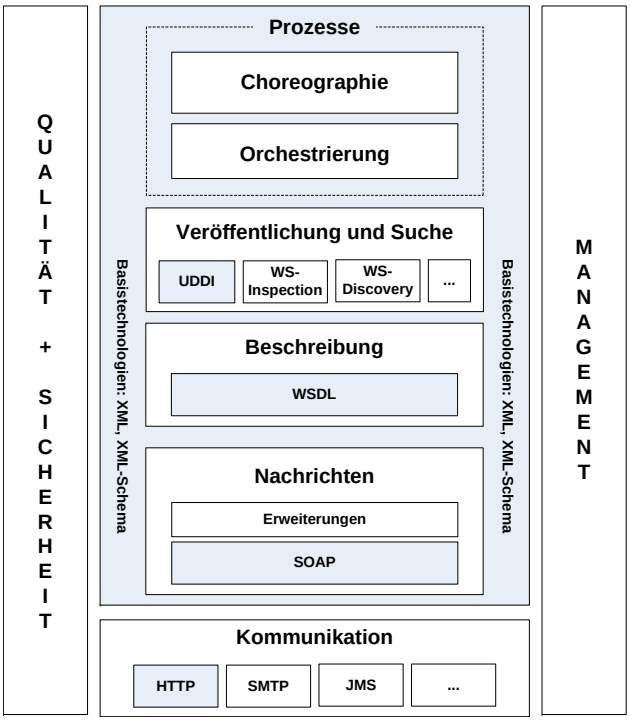


Abbildung 3.6: Web-Service-Protokollstapel mit Kerntechnologien (nach (BHM+04, Pap08))

DI (*Universal Description, Discovery and Integration*) [CHvRR04] gegeben. Die Akzeptanz für derart öffentliche zentrale Verzeichnisdienste ist jedoch zur Zeit noch gering [Mel10]. Ein alternativer Standard zum Einsatz in lokalen Netzwerken (etwa innerhalb einer Organisation) wird durch das ebenfalls von OASIS standardisierte Protokoll *WS-Discovery* (*Web Services Dynamic Discovery*) [MK09] etabliert. Hierbei können die Verzeichnisse lokaler Dienste nach einer vorgegebenen Struktur dezentral verwaltet und innerhalb des lokalen Netzwerks durch Multicast-Nachrichten abgefragt werden [Mel10]. In der Praxis ist zudem das ebenfalls dezentrale Konzept von *WS-Inspection* [BBM+01] verbreitet, bei dem jeder Dienstanbieter ein eigenes Verzeichnis über die bei ihm verfügbaren Dienste hält. Ein Zugriff hierauf ist jedoch nur möglich, sofern dieser Anbieter dem Dienstnutzer im Voraus bekannt ist [Mel10]. Abbildung 3.5 zeigt eine vereinfachte Übersicht der genannten drei Varianten.

Die klassische Web-Service-Architektur wird in der Regel durch das Zusammenspiel der standardisierten Technologien *WSDL*, *SOAP* über *HTTP* und *UDDI* als Verzeichnisdienst determiniert [ACKM04, Ham05, Pap08, Mel10], weshalb diese auch als *Kerntechnologien* bezeichnet werden [Pap08]. Eine Übersicht des grundlegenden Protokollstapels einer dienstorientierter Architektur mit Web Services und der Beziehung als zugrunde liegende Infrastruktur zur Ausführung prozessorientierter Anwendungen ist in Abbildung 3.6 dargestellt. Insbesondere durch die Möglichkeit zur abstrakten Spezifikation eines Dienstes innerhalb der Dienstbeschreibung können die konkreten Details der

Kommunikation fast vollständig vor der konsumierenden Anwendung verborgen werden bzw. müssen nicht zu deren Entwicklungszeit vorliegen [Pap08]. Hinsichtlich der Flexibilität liegt daher ein großer Vorteil in der prinzipiellen Austauschbarkeit und Erweiterbarkeit dieser der Prozessschicht untergeordneten Komponenten. Im Rahmen der dienstorientierten Architektur wird auf dieser Basis die Komposition prozessorientierter Anwendungen als sogenannte *Orchestrierung* und, darauf aufbauend, als *Choreographie* unterstützt [Pel03]. Die folgenden beiden Abschnitte gehen auf diese beiden Arten der Komposition genauer ein.

3.4.4 Orchestrierung

Das Modellierungsziel von Dienstkompositionen ist in der Regel die Definition kollaborativer Prozesse auf Basis einer dienstorientierten Architektur. Eine *Orchestrierung* beschreibt dazu die ausführbaren Aspekte eines dienstbasierten Prozesses aus der Sicht einer einzelnen Partei innerhalb einer Kollaboration [Mel10]. Die Prozesslogik der Orchestrierung beinhaltet dabei die Reihenfolge und die Ausführungsbedingungen der Aufrufe einzelner Dienste und steuert damit in kontrollierter Art und Weise den Nachrichtenaustausch zwischen den beteiligten Parteien. Während dabei durchaus über Organisationsgrenzen hinweg verteilte Ressourcen durch den Prozess aufgerufen und somit integriert werden können, wird die Kontrolle über die Ausführung des Prozesses durch eine einzelne Instanz kontrolliert. Als eine Technik zur Komposition von elektronischen Diensten stellt Orchestrierung somit eine Methode zur zentral gesteuerten Integration von verteilten Ressourcen dar [Wes07].

Im Gegensatz zur sogenannten *Choreographie* von Diensten (vgl. Abschnitt 3.4.5) bezieht sich die Orchestrierung auf einen tatsächlich ausführbaren Prozess, welcher aus der alleinigen Sicht der ausführenden Partei beschrieben ist. Dabei kennt nur diese ausführende Partei die Interna der Implementierung. Zum einen erleichtert dies das Zusammenwirken der Dienste, da die beteiligten Parteien sich nicht mit internen Details der Geschäftspartner auseinandersetzen müssen, zum anderen kann es sich sogar um interne Informationen einer Organisation handeln, welche als Geschäftsgeheimnisse angesehen werden und somit verborgen bleiben sollen [Ley03]. Der beschriebene interne Prozess muss daher auch nicht notwendigerweise konform zu einem bestimmten Koordinationsprotokoll sein [Pel03, Mel10]. Abbildung 3.7 veranschaulicht, wie verschiedene Dienste zu einem Prozess orchestriert werden können, welcher wiederum über eine eigene Dienst-Schnittstelle als komplexer Dienst aufgerufen werden kann [Ses10]. Der Inhalt der Komposition bleibt dabei für den Aufrufer dieses zusammengesetzten Dienstes verborgen (*Black-Box*-Prinzip). Diese Technik wird auch als *rekursive Komposition* [GHJJ10] bezeichnet.

Die Interaktionen einer Komposition können sich dabei über mehrere (verteilte) Anwendungen und verschiedene Organisationen erstrecken. Die Orchestrierung von Diensten kann daher einerseits zur innerbetrieblichen Anwendungsintegration genutzt werden und darüber hinaus zur Beschreibung

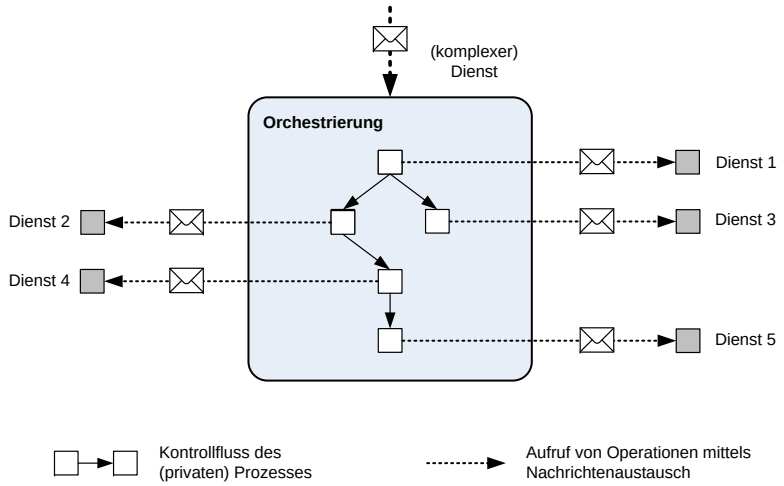


Abbildung 3.7: Orchestrierung (tlw. nach (Pel03))

von Prozessen verwendet werden, welche unternehmensübergreifende Anwendungsfunktionalität integrieren. Die Vision des Konzeptes ist die automatische Auswahl und prozessorientierte Komposition von Diensten, so dass Anwendungen in der Lage sind, nur aufgrund einer genauen Vorgabe der benötigten Funktionalität selbständig (d. h. ohne Eingriffe von menschlichen Benutzern) zur Laufzeit geeignete Dienste auffinden, selektieren und in die Komposition bzw. in den Prozess einsetzen zu können [ACKM04, Wes07].

Entsprechend dieser Vision kann die Orchestrierung von Diensten den Lebenszyklus einer prozessorientierten Anwendung (vgl. Abbildung 2.25) teilweise begleiten und in Hinblick auf die technische Flexibilität des Prozesses erweitern [KB04b, KHC+05]. In der Phase der technischen Modellierung wird der für die Geschäftslogik relevante Kontroll- und Datenfluss in einer geeigneten Prozessbeschreibungssprache festgehalten und die in Form von Diensten dargestellten Anwendungsfunktionalitäten als abstrakte Aktivitäten des Prozesses eingesetzt [PY02]. Potentiell geeigneten Dienstanbietern können hierfür funktionale Verantwortlichkeiten bzw. *Rollen* zugewiesen werden [PY02]. Für die Zuweisung zu konkreten Ressourcen eröffnet die unterliegende dienstorientierte Architektur ein breites Spektrum an Bindungsmöglichkeiten. Zum einen können Dienste manuell aufgefunden und statisch in eine Orchestrierung eingebunden werden, zum anderen kann eine automatisierte Suche und Einbindung von geeigneten Diensten zur Laufzeit des Prozesses durchgeführt werden [Wes07]. Man unterscheidet daher im Hinblick auf die Bindung zwischen einer *proaktiven* und einer *reaktiven Kopplungsstrategie* [RS04]:

- **Proaktive Kopplung:** Bei einer *proaktiven Kopplung* wird der Prozess mit den aufzurufenden Dienstinstanzen bereits zur Entwicklungszeit komplett zusammengestellt. Diese frühe Bindung eignet sich insbesondere, wenn die verwendeten Dienste a priori bekannt sind und sich nur selten ändern. Der resultierende Geschäftsprozess ist in der Regel robust und kann gegenüber einer dynamischen Bindung Geschwindigkeitsvor-

teile aufweisen, da es während der Ausführung nicht zu unberechenbaren Verzögerungen durch die Suche nach geeigneten Dienstinstanzen kommt [RS04].

- **Reaktive Kopplung:** Bei einer *reaktiven Kopplung* erfolgt die Bindung von konkreten Diensten erst zu dem Zeitpunkt, zu dem der komponierte Dienst nachgefragt wird bzw. der komplexe Prozess tatsächlich ausgeführt werden soll (späte Bindung). Das dynamische Binden kann entweder über eine Referenz erfolgen oder erfordert eine vorherige Suche nach geeigneten Diensten. Zum Beispiel kann ein Verzeichnisdienst verwendet werden, um das hier zumeist automatisierte Auffinden der benötigten Softwarekomponenten zu unterstützen. Falls mehrere geeignete Dienste für eine Aktivität existieren, muss in einem weiteren Schritt eine Auswahl getroffen werden [KB04b]. Dabei sind unter anderem Qualitätsaspekte und andere nicht-funktionale Kriterien für die Entscheidung relevant. Reaktive Kopplung eignet sich insbesondere dann, wenn einzelne Komponenten der Komposition nicht a priori bekannt sind oder erst zur Laufzeit festgelegt werden können, zum Beispiel aufgrund bestimmter Benutzeranforderungen oder Eingabeparameter. Des Weiteren können bei dieser Form der Komposition auch noch zur Laufzeit Optimierungen vorgenommen werden. Insgesamt kann damit eine höhere Flexibilität während der Ausführung erzielt werden [RS04].

Da die entstehende Komposition in beiden Fällen direkt umsetzbar ist, kann die um alle erforderlichen Laufzeitaspekte ergänzte Prozessbeschreibung in der Regel sofort ausgeführt werden, indem der Prozess instantiiert und somit die in der Komposition enthaltenen konkreten Dienste in der vorgegeben Reihenfolge aufgerufen werden. Die eigentliche Ausführung wird dabei in der Regel von einer Prozess-Engine bzw. einem Prozessmanagementsystem unterstützt und für eine spätere Analyse des Prozesses entsprechend überwacht [KB04b].

3.4.5 Choreographie

Eine *Choreographie* beschreibt die Aufgaben und das Zusammenspiel mehrerer Prozesse unter dem Aspekt der Zusammenarbeit [Pel03, Mel10]. Durch die Choreographie wird dabei das *nach außen sichtbare Verhalten* während der Ausführung dieser Prozesse beschrieben, um Abhängigkeiten von Diensten verschiedener Orchestrierungen darzustellen [Wes07]. Die Choreographie setzt daher als Beschreibung eines abstrakten Kommunikationsprotokolls zwischen den Prozessen verschiedener Parteien auf (vgl. Abbildung 3.8). Die an der Choreographie teilnehmenden Organisationen bzw. Organisationseinheiten werden dabei in der Regel als *Partner* bezeichnet [Mel10].

Grundlegend für eine Beschreibung von organisationsübergreifenden Prozessen durch eine Choreographie ist, dass alle beteiligten Partner die von ihnen angebotene (komplexe) Anwendungsfunktionalität in Form von elektroni-

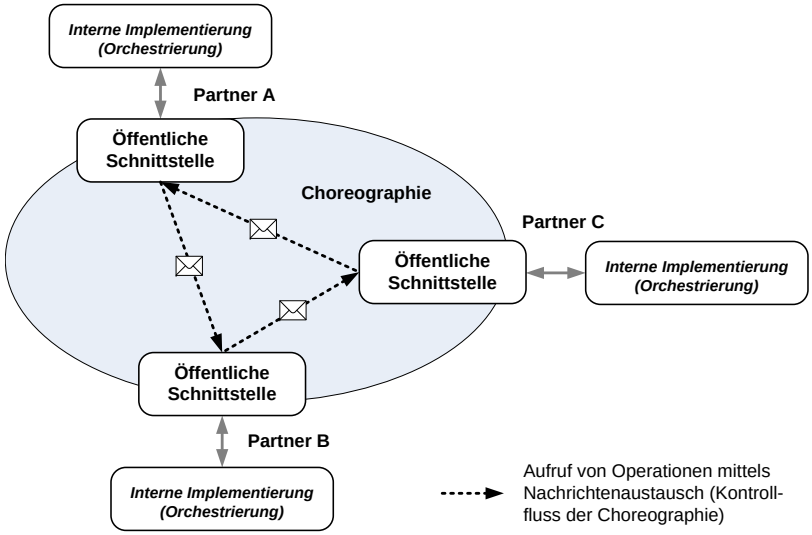


Abbildung 3.8: Choreographie (tlw. nach (Pel03))

schen Diensten bereitstellen. Die einzelnen Dienste der teilnehmenden Partner stellen dabei die Aktivitäten des Prozesses dar, während der Kontrollfluss des Prozesses durch den Ablauf des Nachrichtenaustausches zwischen den einzelnen Diensten festgelegt ist. Das Besondere bei einer Choreographie ist, dass (im Gegensatz zur Orchestrierung) keine zentrale Instanz existiert, welche die alleinige Kontrolle über die Ausführung des verteilten Prozesses besitzt. Stattdessen wird die automatisierte Zusammenarbeit mehrerer autonomer und gleichberechtigter Parteien allein auf Basis der vorliegenden Choreographie-Beschreibung ermöglicht. Die Referenz zwischen den in der öffentlichen Schnittstelle definierten Aktionen und den Dienst-Operationen ist dabei in der Regel dynamisch, so dass konkrete Dienste zur Laufzeit auf Basis bestimmter Kriterien ausgewählt werden können. Diese Flexibilität ermöglicht es, geeignete konkrete Partner einer Choreographie entweder statisch festzulegen oder in Abhängigkeit spezieller Eingabeparameter (z. B. von Seiten des Kunden) erst während der Laufzeit des Prozesses auszuwählen und einzubinden. Diese Art der dezentralen Ausführung setzt allerdings eine vorherige Einigung der (potentiell) beteiligten Partner auf eine solche gemeinsame Beschreibung, die Zuordnung der jeweilig zu erbringenden Leistung sowie die Implementierung der hierfür verabredeten Schnittstellen voraus [DKB08].

Bei der Erstellung einer Choreographie stehen für die Definition eines organisationsübergreifenden Prozesses also die reihenfolgeabhängige Kommunikation sowie die Definition der Verhaltensweisen beteiligter Parteien im Mittelpunkt. Hierzu spezifiziert die Kernbeschreibung der Choreographie alle relevanten Interaktionen aus einer abstrakten, globalen Sicht. Bei der Modellierung des Nachrichtenaustausches werden nur Aktionen einbezogen, welche von außen beobachtbar sind. Dazu konform beschreibt jeder am Prozess beteiligte Partner seine eigene Rolle an der Kollaboration in Form einer *öffentlichen Schnittstelle*. Diese öffentliche Schnittstelle wird aufgrund des Verhaltensa-

spekts daher auch als *Behavioral Interface* bezeichnet [Wes07]. Im Gegensatz zur eigentlichen Choreographie entfällt bei der Definition von öffentlichen Schnittstellen die globale Sicht auf den Gesamtprozess, da die Interaktionen nur aus der Sicht eines Partners betrachtet werden. Die interne Prozesssteuerung und die Umsetzung der tatsächlich ausführbaren (Teil-) Prozesse kann mit beliebigen alternativen bzw. sogar proprietären Technologien realisiert werden. Die Choreographie selbst ist ohne die Implementierung der öffentlichen Schnittstellen durch die Partner jedoch nicht direkt ausführbar [Wes07, DKB08].

Orchestrierung und Choreographie stellen somit zwei unterschiedliche, sich ergänzende Techniken zur Komposition von Diensten dar. Abbildung 3.8 zeigt das Zusammenspiel zwischen den internen bzw. privaten Prozessen verschiedener Partner, ihren öffentlichen Schnittstellen und der Definition einer darauf aufbauenden Choreographie, welche den gemeinsamen, verteilten Anwendungsfall beschreibt. Während die Implementierung des privaten Prozesses in technischer Hinsicht den kollaborierenden Unternehmen überlassen ist, beinhaltet die Teilnahme an einer Choreographie jedoch die Einwilligung der Geschäftspartner, sich an die inhaltlichen Vorgaben des gemeinsam ausgeführten Prozesses zu binden. Der Choreographie kann daher nicht nur eine Steuerungsfunktion zugeschrieben werden, sondern auch die Funktion eines Vertrages, in welchem der funktionale Inhalt der intern zu erbringenden Leistung aller Teilnehmer festgehalten wird [Pel03]. Dieser Mehrwert kann insbesondere durch die Kombination mit *Service Level Agreements* für Dienste der öffentlichen Schnittstelle ausgebaut werden [DP06].

Dienstbasierte Choreographien unterstützen die Kollaboration mehrerer unabhängiger Teilnehmer, ohne deren Autonomie durch die einseitige Definition von gemeinsamen Prozessen in Frage zu stellen. Die Entkopplung von öffentlicher Schnittstelle und intern zu realisierenden (Teil-)Prozessen führt für die teilnehmenden Parteien zu einem großen Flexibilitätsgewinn, da ihre internen Systeme beliebig angepasst werden können, solange weiterhin das vereinbarte Verhalten über die öffentliche Schnittstelle bereitgestellt wird. Das Modell der Choreographie stellt also eine relativ realitätsnahe Abbildung von Geschäftsbeziehungen dar, welche auf der Definition von Verträgen beruhen. Zudem kann die Erstellung von Choreographien rekursiv geschehen, das heißt, dass es möglich ist, aus vorhandenen Diensten erneut einen Dienst zu erstellen, der seinerseits in anderen Choreographien wieder verwendet werden kann. Kombinierbarkeit und Modularität von angebotenen fachlichen Funktionalitäten werden hierdurch gefördert [Pel03].

3.5 Flexibilität durch Context Awareness

Um im Bedarfsfall die geeignete Anpassung eines Systems zu ermöglichen, ist es notwendig, relevante Zustände des Systems und seiner Umgebung wahrnehmen und auswerten zu können. Die wesentlichen Aufgaben bestehen dabei darin, den für das jeweilige System *relevanten* Ausschnitt der Umgebung zu

identifizieren, dessen Dynamik in Hinblick auf die Reichweite der zu erwartenden Änderungen abzuschätzen und eine Beobachtung der relevanten Parameter vorzunehmen, so dass die für das System wesentlichen Umgebungsbedingungen nach Möglichkeit automatisiert erkannt und zeitnah nach deren Auftreten verarbeitet werden können (vgl. Abschnitt 3.2). In Bezug auf prozessorientierte Anwendungen wird dabei das ursprünglich aus dem Bereich des *Ubiquitous Computing* (vgl. [Wei93, SKP09]) stammende Konzept des *Kontextbewusstseins* (engl. *Context Awareness*) auf das Prozessmanagement übertragen und in geeigneter Weise in die Modellierung, Ausführung und Überwachung von Prozessen integriert. Die Beobachtung und Nutzung von Kontextinformationen zur Anpassung von prozessorientierten Anwendungen wird allgemein unter den Oberbegriffen des *Context-aware (Business) Process Management* [RRFA06, RR06, Krö10] bzw. des *Context-aware Workflow Management* [MBCP05, AFG+07, CCC07] zusammengefasst. Im Folgenden werden nach einer kurzen Einführung die wesentlichen Ansätze zur Nutzung von Umgebungsinformationen im Rahmen des Prozessmanagements vorgestellt.

3.5.1 Context Awareness

In der Informatik bezeichnet *Context Awareness* bzw. *Kontextbewusstsein* die Fähigkeit von Anwendungssystemen, Umgebungsinformationen erlangen und verarbeiten zu können. Der *Kontext* stellt dabei die Abbildung der Menge aller verfügbarer Umgebungsdaten auf die für das Anwendungssystem relevanten Informationen und Bedingungen dar. Er umfasst insbesondere alle Personen, Ortsdaten und Objekte, die Auswirkungen auf das Verhalten der Anwendung haben können, wobei auch die Anwendung selbst und ihre eigenen Aktivitäten Teile des Kontextes darstellen können [Dey99, Dey01].

Die für die Interaktion zwischen Benutzer und Anwendung relevanten Objekte werden nach den Autoren DEY und ABOWD [Dey99] als abstrakte *Entitäten* mit *Attributen* aufgefasst. Entitäten bzw. ihre Attribute können *Zustände* besitzen, welche allein oder in Kombination die *Situation* einer Entität bestimmen [Dey01, Krö10]. Hierbei wird den Attributen *Ort*, *Zeit*, *Identität* und *Aktivität* eine besondere Bedeutung als sogenannter *primärer Kontext* zugesprochen, während alle davon abhängigen Attribute als *sekundärer Kontext* einer Entität bezeichnet werden [Dey99]. Der primäre Kontext einer Entität kann demnach als Schlüssel dienen, um auf sekundären Kontext dieser Entität oder auf primären Kontext anderer Entitäten zugreifen zu können. Zum Beispiel kann eine Wettervorhersage als sekundärer Kontext nur bestimmt werden, wenn geeignete primäre Kontextinformationen in Form von Ort und Zeit vorliegen (vgl. [Dey99]). Die Relevanz der einzelnen Kontextdaten für die individuelle Anwendung wird dabei durch die Einteilung in primären und sekundären Kontext nicht berührt.

Ebenfalls unabhängig von dieser Unterscheidung können Kontextinformationen in direkter Weise durch physikalische oder logische Sensoren und durch Kommunikation gewonnen werden [Che04]. Derart direkt erfassbare Einzel-

informationen werden als *Low-Level-Kontext* bezeichnet, z. B. Temperatur, Systemzeit oder der Name des aktuellen Benutzers. Durch Berechnungen basierend auf Daten des Low-Level-Kontextes können wiederum Informationen einer höheren Abstraktionsebene abgeleitet werden (*High-Level Kontext*). Hierbei kann z. B. auf Basis von einfachen Umgebungsdaten wie Zeit, Position und Geräuschpegel auf komplexe Zustände und Aktivitäten wie die Teilnahme an einer Besprechung geschlossen werden [Che01].

Da die Erfassung von Kontextdaten insbesondere in einem dynamischen Umfeld von Interesse ist, ist ein Einsatz kontextbasierter Technologien insbesondere im Bereich ubiquitärer Anwendungen verbreitet [SLI08]. Hierbei stehen in der Regel Daten der physikalischen Umgebung, der (technischen) Infrastruktur und des menschlichen Benutzers einer Anwendung im Vordergrund [Che01, Bad07]:

- **Computing Context:** Diese Art des Kontextes umfasst zum Beispiel das Leistungsvermögen von Geräten (z. B. Speicher- und Rechenkapazität, verfügbare Energiereserven), die Eigenschaften von Netzwerkverbindungen (z. B. Datenrate, Latenzzeit, Kosten) oder die Anzahl und Art anderer erreichbarer Geräte (z. B. Drucker, Bildschirme oder Computer).

- **User Context:** Diese Kategorie stellt den Benutzer in den Vordergrund und kann zum Beispiel ein Profil des Benutzers, seinen Aufenthaltsort, andere Menschen in seiner Umgebung sowie Informationen über seine aktuelle Aktivität enthalten.

- **Physical Context:** Diese Art des Kontextes kann sich entweder auf die physikalischen Eigenschaften (z. B. Lichtverhältnisse, Umgebungsgeräusche, Temperatur) beziehen oder aber allgemein auf die natürliche Umwelt mit ihren Charakteristika und Objekten.

Die Eigenschaft der *Context Awareness* ist vor dem Hintergrund des allgemeinen Sprachgebrauchs zunächst einmal auf ein passives Beobachten und Bewusstsein begrenzt. Anwendungen mit *passivem Kontextbewusstsein* beschränken sich daher in erster Linie auf die Anzeige oder das Speichern von wahrgenommenen Kontextdaten [Che01, Kun08]. In seiner weitergehenden Bedeutung beinhaltet der Begriff *Awareness* dabei jedoch sowohl eine wahrnehmende Komponente als auch ein Verstehen des Wahrgenommenen. Dabei wird speziell durch das Erkennen und Verstehen der Aktivitäten anderer Objekte der Kontext für eigene Aktivitäten gebildet [DB92, Che01]. Ein Kontextbewusstsein kann es also einem Individuum ermöglichen, aktuelle Informationen oder Ereignisse, die durch die Präsenz von anderen Geräten, Personen oder Objekten ausgelöst werden, wahrzunehmen und das eigene Handeln darauf abzustimmen. Als Ergebnis eines solchen Mechanismus kann dann beispielsweise die (eigenständige) Anpassung eines Anwendungssystems resultieren. Nach CHEN et al. können kontextbezogene Anwendungen daher auch über ein *aktives Kontextbewusstsein* verfügen, wenn sie ihr Verhalten bewusst

an einen geänderten Kontext anpassen [Che01, Kun08]. Diese Fähigkeit wird auch als *Kontextadaptivität* bezeichnet [LP06].

Für die Entwicklung und den Betrieb kontextbewusster Systeme ist es erforderlich, dass die für die jeweilige Anwendung relevanten Kontextinformationen in geeigneter Weise modelliert und verwaltet werden können. Ein *Kontextdatenmodell* erlaubt es der Anwendung, benötigte Daten in vorgegebener Form zu beziehen und auf diese Weise einheitlich zu interpretieren. Das Modell legt dabei fest, wie die einzelnen Informationen in Form von Datenstrukturen ausgedrückt werden und welche Erweiterungsmöglichkeiten bestehen, um sie anwendungsspezifisch anzupassen [Tur06]. Für die Verwaltung von Modelldaten kommen *Kontextmanagementsysteme* zum Einsatz, welche die Kontextdaten von (physikalischen) Sensoren beziehen, diese bei Bedarf aktualisieren, speichern, transformieren oder verdichten und der Anwendung in geeigneter Form zur Verfügung stellen [PCP07]. Idealerweise schafft eine geeignete Middleware in Form eines Rahmenwerks eine Abstraktionsebene, welche die Ermittlung, Bereitstellung und den Austausch von Kontextinformationen aus der Perspektive der Anwendung transparent gestaltet [Che04, Tur06]. So können Kontextdaten z. B. mit Hilfe von speziellen Abfragesprachen (*Context Query Languages*) zugänglich gemacht werden [RWK+08].

3.5.2 Kontext prozessorientierter Anwendungen

In Bezug auf prozessorientierte Anwendungen bildet Context Awareness die Grundlage für kontextbewusste bzw. kontextadaptive Prozesse, welche die Flexibilität besitzen, sich speziell an ändernde Umgebungsbedingungen anpassen zu lassen [Krö10]. Ein wichtiger Schritt hierzu ist die Identifikation und geeignete Abbildung des für die prozessorientierte Anwendung *relevanten* Kontextes. Dieser Kontext kann dann entweder direkt in den Prozess eingebunden werden und einer inhaltlichen Anpassung des Prozesses an sich ändernde Begebenheiten dienen (*Context-aware Process Modelling*), oder der Ausführungsumgebung zur Verfügung gestellt werden, um eine technische Anpassung der Prozessausführung in Bezug auf Qualität, Fehlerbehandlung oder Repräsentation zu ermöglichen [NBNK05]. Zusammengefasst werden Kontextdaten im Rahmen von prozessorientierten Anwendungen insbesondere für folgende Zwecke eingesetzt [EMT06, SLI08]:

- **Benutzerbezogene Anpassung:** Kontextdaten werden zur Anpassung der prozessorientierten Anwendung an die Bedürfnisse eines speziellen Nutzers verwendet [SLI08], z. B. zur inhaltlichen Anpassung des Kontrollflusses an den Aufenthaltsort des Prozessinitiators [MBCP05, AFG+07] oder zur Anpassung der Benutzungsschnittstelle an das zur Interaktion verwendete Gerät eines Prozessteilnehmers [MBCP05, AFG+07, CELS07].

- **Fehlerkorrektur:** Kontextdaten werden zur Fehlererkennung und -behebung eingesetzt, z. B. zum Erkennen einer unzureichenden Netz-

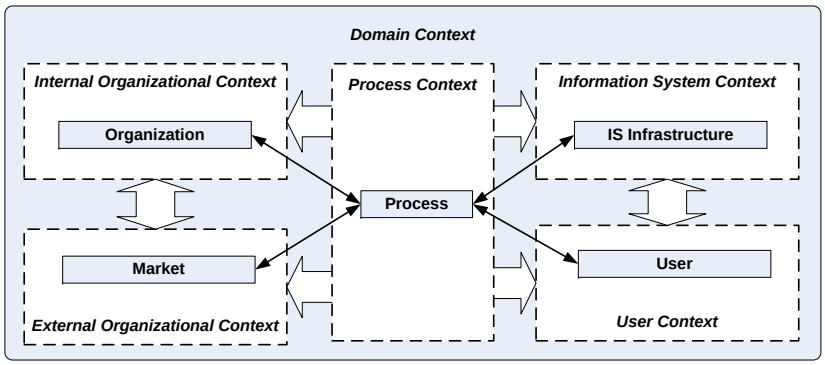


Abbildung 3.9: Arten und Beziehungen von Kontext innerhalb der abstrakten Domäne prozessorientierter Anwendungen (Krö10)

werkverbindung und zum entsprechenden Austausch von Diensten oder Kommunikationsmedien [NG06, EMT06].

- **Optimierung:** Kontextdaten werden zur Optimierung von nichtfunktionalen Aspekten verwendet, z. B. um eine Verkürzung der Ausführungszeit zu erreichen [LSPF07].

Aufgrund der Individualität einzelner Prozesse und ihrer individuellen Anwendungsdomänen [Kun08], der damit verbundenen Subjektivität [Krö10] sowie der unterschiedlichen Ziele der angestrebten Anpassungen [EMT06, SLI08] ist eine vollständige konkrete Beschreibung des relevanten Kontextes für prozessorientierte Anwendungen im Allgemeinen nicht möglich. Die relevanten Rahmenbedingungen können jedoch in abstrakter Form klassifiziert und als grundlegendes Verständnis für die wesentlichen Einflussfaktoren konzeptualisiert werden.

Für eine erste einfache Klassifizierung des relevanten Kontextes steht die Interaktion der prozessorientierten Anwendung mit ihrer Umgebung im Vordergrund. Die Autoren ROSEMANN und RECKER [RRFA06] unterscheiden demnach zwischen *intrinsischen* und *extrinsischen* Einflussfaktoren für Flexibilität und Anpassbarkeit, wobei den Kategorien eine unterschiedliche Gewichtung zugesprochen wird. Für SAIDINI und NURCAN [SN07] stehen neben zeit- und ortsbezogenen Kontextdaten insbesondere ressourcen- und organisationsbezogene Kontextdaten im Vordergrund, die mit der Zuweisung von menschlichen Akteuren zu Rollen und der Zuordnung von Rollen zu den Aufgaben und Zielen des Prozesses in Verbindung stehen. Auf Basis dieser Überlegungen und in Abhängigkeit der genannten Eigenschaften und Ziele wurde von KRÖSCHEL [Krö10] die folgende Konzeptualisierung von Kontext für prozessorientierte Anwendungen abgeleitet, welche mit ihren Interdependenzen und Kernelementen auch in Abbildung 3.9 visualisiert ist und dieser Arbeit im Folgenden zugrunde liegen soll:

- **Kontext der übergeordneten Anwendungsdomäne (*Domain Context*):** Der Kontext der *übergeordneten Anwendungsdomäne* gibt einen

äußeren Rahmen vor, in welchem alle für die prozessorientierte Anwendung relevanten Entitäten und ihre Attribute eingebettet sind. Beispiel für eine Anwendungsdomäne auf dieser Abstraktionsebene ist die Ausführung von Geschäftsprozessen im konkreten Unternehmensumfeld [Krö10].

- **Prozesskontext (*Process Context*):** Der *Prozesskontext* inkludiert alle Elemente, die direkt mit dem Prozess und seiner Ausführung in Verbindung stehen. Hierzu gehören in erster Linie der Prozess selbst, seine jeweiligen Ziele, sein Kontrollfluss, die zu seiner Ausführung benötigten Ressourcen und die dabei involvierten Organisationen bzw. Organisationseinheiten. Der Prozesskontext steht dabei eng mit dem ihn umgebenen Kontext in Verbindung [Krö10].

- **Interner Kontext einer Organisation (*Internal Organizational Context*):** Der *interne Kontext einer Organisation* umfasst Informationen über die Organisation bzw. Organisationseinheit, für welche der Prozess entwickelt wurde. Informationen eines solchen internen Kontextes einer Anwendungsdomäne im Unternehmensumfeld sind z. B. die übergeordnete Strategie des Unternehmens, das Geschäftsfeld, die Organisationsstruktur oder die Anzahl der Mitarbeiter. Im Falle von organisationsübergreifenden Prozessen wird hierbei auch das jeweilige Organisationsnetzwerk als interner Kontext betrachtet [RR06, RRFA06, Krö10].

- **Externer Kontext einer Organisation (*External Organizational Context*):** Der *externe Kontext einer Organisation* bezieht sich auf Informationen über Umgebungsbedingungen, die außerhalb des Einflussbereichs der betrachteten Organisation liegen, aber dennoch Auswirkungen auf die prozessorientierte Anwendung haben können. KRÖSCHEL [Krö10] fasst diese Bedingungen als *Wirtschaftsraum* (*Market*) zusammen, welcher die externen Teilnehmer und Aspekte im Umfeld der Organisation zusammenfasst. Hierzu gehören daher z. B. Wettbewerber, Kunden und Lieferanten [RR06, RRFA06], aber z. B. auch politische, kulturelle oder ortsbasierte Aspekte [Krö10].

- **Kontext der Informationssysteme (*Information System Context*):** Der *Kontext der Informationssysteme* enthält die technische Infrastruktur aller Informations- und Kommunikationssysteme, welche für die Ausführung des betrachteten Prozesses zur Verfügung steht. Dies umfasst sowohl die Hard- und Software, auf welcher die Implementierung des Prozesses beruht (z. B. Dienste im Rahmen einer dienstorientierten Architektur) als auch die Schnittstellen zu Benutzern [Krö10].

- **Benutzerkontext (*User Context*):** Der *Benutzerkontext* hebt die Bedeutung der menschlichen Akteure [SN07] hervor, welche an der Modellierung, Ausführung oder Überwachung des betrachteten Prozesses beteiligt sind. Hierbei sind zum Beispiel geschäftliche Rollen, Aufgaben und

Verantwortlichkeiten, aber auch persönliche Nutzerprofile, Qualifikationen oder sozialer Hintergrund von Bedeutung [Krö10].

Im Folgenden werden basierend auf diesem Kontextverständnis die wichtigsten Voraussetzungen und Ansätze zur Nutzung von Kontextdaten für die Flexibilisierung von prozessorientierten Anwendungen thematisiert.

3.5.3 Funktionale Anpassung von Prozessen

Durch die Berücksichtigung des domänenabhängigen Kontextes während der Modellierung von Prozessen kann zur Laufzeit eine Einbettung von aktuellen Kontextdaten in den Prozess und damit eine kontextabhängige inhaltliche Anpassung der Prozessausführung ermöglicht werden. Als einfacher Ansatz zur Umsetzung (adaptiver) kontextbewusster prozessorientierter Anwendungen können Kontextdaten als spezielle Variablen des Prozesses mitgeführt und zur Laufzeit der Prozessinstanzen ausgewertet werden [MBCP05]. Dazu werden zunächst alle potentiellen Anpassungen an mögliche Kontexte als alternative Kontrollflusspfade im Prozessmodell modelliert. Auf dieser Modellierung aufbauend werden zur Laufzeit Kontextdaten erhoben und als Grundlage der Entscheidung für die Ausführung eines bestimmten Pfades verwendet. Die Steuerung des Kontrollflusses geschieht dementsprechend durch Auswahl von Kontrollflusspfaden, welche für den jeweiligen Kontext geeignet sind [WKN08, SLI08]. Abbildung 3.10 zeigt einen (fiktiven) kontextbasierten Prozess zur Unterstützung von Besuchern der Hamburger Messe, wobei abhängig von erhobenen Orts- bzw. Benutzerdaten unterschiedliche Reisevorbereitungen durchgeführt werden müssen.

Die gezeigte Vorgehensweise entspricht der in Abschnitt 3.3.1 vorgestellten *A-priori-Flexibilität* durch explizit modellierte konkrete Pfade, wobei der dort genannte Nachteil in Form einer hohen Komplexität bei der Beschreibung aller potentiell auszuführenden Alternativen zum Tragen kommt. Der Ansatz erlaubt ein gewisses Maß an Flexibilität in Bezug auf den Umgang mit vorhersehbaren Ereignissen (*Typ-I-Flexibilität*, vgl. Abschnitt 3.2), ist in der Flexibilität für den Umgang mit unvorhergesehenen Ereignissen (*Typ-II-Flexibilität*) jedoch relativ eingeschränkt, da hierfür jeweils eine nachträgliche Anpassung des Prozessmodells notwendig wird [SLI08] (*A-posteriori-Flexibilität*).

Weiterführende Ansätze bauen daher in der Regel auf dem Prinzip der *Abstraktion* auf. Auf dieser Basis können zur Anpassung des Prozesses an die Situation des Nutzers *kontextabhängige Bereiche* des Prozesses identifiziert und auf Basis von aktuellen Kontextdaten zur Laufzeit konkretisiert werden [MBCP05, NBNK05, AFG+07, ATEvdA06]. In der Regel handelt es sich bei diesen kontextabhängigen Bereichen um abstrakte Aktivitäten mit Zuordnung einer (ggf. erweiterbaren) Menge von kontextabhängigen Implementierungen [AFG+07]. Diese Vorgehensweise entspricht den in Abschnitt 3.3.1 eingeführten explizit modellierten abstrakten Pfaden. Bei Erreichen einer abstrakten Aktivität wird ein Kontextmanagementsystem aufgerufen, welches die auszuführende konkrete Implementierung des Prozesses einbindet und als

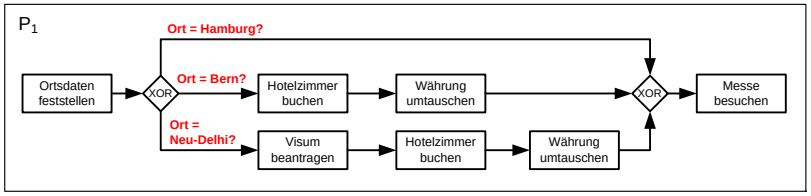


Abbildung 3.10: Steuerung des Kontrollflusses durch kontextbasierte Auswahl explizit modellierter konkreter Pfade (Beispiel)

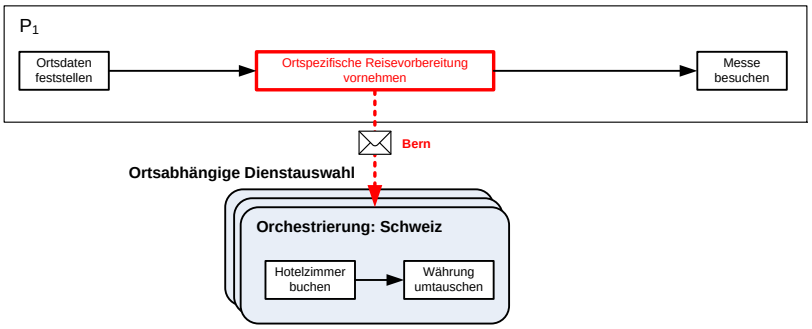


Abbildung 3.11: Steuerung des Kontrollflusses durch kontextbasierte Implementierung explizit modellierter abstrakter Pfade (Beispiel)

Subprozess des übergeordneten Prozesses ausführt [AFG+07]. Das Verfahren kann optimiert werden, indem konkrete Implementierungen für Subprozesse (sogenannte *Worklets*) auch zur Laufzeit hinzugefügt werden [ATEvdA06].

Die Umsetzung solcher Ansätze basiert in vielen Fällen auf einer dienstorientierten Architektur, deren Eigenschaften es erlauben, verschiedene Implementierungen hinter einer abstrakten Dienstschnittstelle zu kapseln (vgl. Abschnitt 3.4) oder alternative Dienste bereitzuhalten, die bei Eintreten eines bestimmten Kontextes im Prozessmodell oder in der laufenden Prozessinstanz ersetzt werden [CELS07, CCSC07, AFG+07, ATEvdA06]. Die Auswahl der zum aktuellen Kontext passenden Implementierung kann dabei auf Basis von Ontologien und nach festgelegten Regeln und Policies erfolgen [DS05, EMT06, CELS07, CCSC07, ETM07]. Ein entsprechendes (verallgemeinerndes) Beispiel ist in Abbildung 3.11 dargestellt. Hierbei dient die abstrakte Aktivität „Ortsspezifische Reisevorbereitung vornehmen“ als Platzhalter für den Aufruf eines ortsspezifischen Dienstes, welcher anhand von für den Anwendungsfall sinnvollen Klassifikationen und Zusammenhängen ausgewählt werden kann.

Für die Integration von Context Awareness zur Anpassung des Kontrollflusses wird in der Regel zusätzlich zu den bestehenden Komponenten des Prozessmanagementsystems (vgl. Abschnitt 2.7) ein geeignetes *Kontextmanagementsystem* benötigt. Zudem besteht je nach Ansatz Bedarf für spezielle Erweiterungen an der Prozess-Engine, der Prozessbeschreibungssprache (z. B. [HCKC06, CSCC07]) oder die Installation von geeigneten

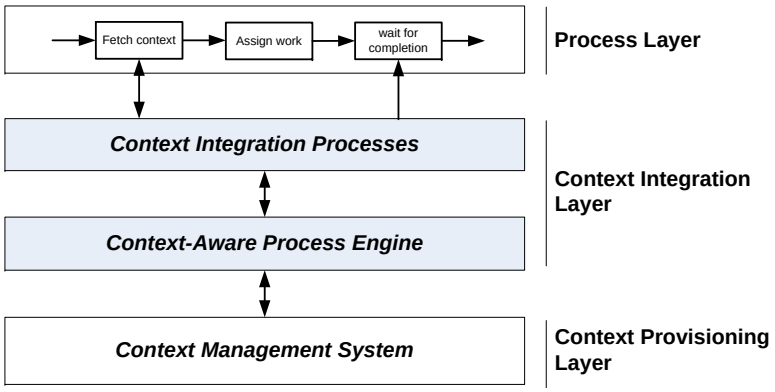


Abbildung 3.12: Schichtenmodell für die Integration von Kontextdaten in Prozesse (vereinfacht nach (WKN08, WNL08))

Adapter-Komponenten. Um Anpassungen an bestehenden Systemen und Sprachen nach Möglichkeit zu vermeiden, schlagen die Autoren WIELAND et al. [WKN08, WNL08] die in Abbildung 3.12 vereinfacht dargestellte Schichtenarchitektur vor. Hierbei können auf der untersten Ebene bestehende (generische) Rahmenwerke zum Kontextmanagement verwendet werden, um Kontextinformationen zu erheben, zu verwalten und höheren Schichten bereitzustellen (*Context Provisioning Layer*). Diese Informationen können von einer geeigneten Prozess-Engine (*Context-Aware Process Engine*) bezogen und über eine geeignete Adapterschicht in laufende Prozesse integriert werden (*Context Integration Layer*). Dabei wird vorgeschlagen, die Bereitstellung domänenabhängiger Kontextdaten als Dienste zu kapseln, so dass diese ohne zusätzlichen erheblichen Entwicklungsaufwand in Standard-Prozessbeschreibungssprachen zur Orchestrierung von Diensten (vgl. Abschnitt 3.4.4) integriert werden können (*Process Layer*).

Die Autoren SMANCHAT et al. [SLI08] kommen bei einem bewertenden Vergleich der oben genannten Ansätze zu dem Schluss, dass die im Voraus festgelegte Auswahl von veränderlichen Abschnitten des Prozesses bzw. die Vorgabe von konkreten Bindungspunkten die Flexibilität relativ stark einschränkt [SLI08]. Das Verfahren kann daher ausgeweitet werden, indem jede Aktivität des Prozesses als ein potentieller Subprozess modelliert wird, so dass in Hinblick auf die Flexibilität keine Nachteile durch eine feste Determinierung kontextrelevanter Abschnitte entstehen (vgl. [CCSC07]). Im Fall veränderlicher Abschnitte ist ggf. auch der Austausch und somit ein Rollback bereits ausgeführter Prozessabschnitte zu berücksichtigen [MBCP05]. Zudem kann neben dem Austausch von Aktivitäten der Inhalt eines Prozesses auch durch Entfernen oder Hinzufügen von Aktivitäten angepasst werden [MGR04, GMR+05]. Die Strukturierung des Prozesses wird durch diese weitergehende Flexibilisierung jedoch sehr stark relativiert.

3.5.4 Nicht-funktionale Anpassung der Prozessausführung

Neben ihrer Bedeutung für die genannte inhaltliche Anpassung von Prozessabläufen spielen Kontextdaten insbesondere bei der Gewährleistung und Optimierung der inhaltlich vorgesehenen Prozessausführung eine große Rolle [MBCP05, NBNK05, NG06, LSPF07]. Hierbei steht zum einen das Erkennen von *Fehlersituationen* (d. h. die Feststellung eines für die Ausführung unzureichenden Kontextes) [NG06, EMT06] sowie das Erkennen von Kontextänderungen als Potential für eine *verbesserte Ausführung* des Prozesses in Hinblick auf Qualitätseigenschaften (*Quality of Service*, kurz *QoS*) im Vordergrund [LSPF07, Kun08]. Die Wahrnehmung und Nutzung von qualitativen Kontextinformationen zur Anpassung nicht-funktionaler Eigenschaften der Prozessausführung wird daher auch als *QoS-Awareness* [ZBHN+04, CDPEV05, BSR+06] bezeichnet.

Nicht-funktionale Aspekte umfassen dabei einen großen Raum an Eigenschaften. Neben allgemeinen nicht-funktionalen Eigenschaften von Softwaresystemen, wie z. B. Robustheit, Zuverlässigkeit, Testbarkeit oder Benutzerfreundlichkeit, stehen als Kontext prozessorientierter Anwendungen insbesondere konkret messbare Parameter der unterliegenden (dienstorientierten) Softwarearchitektur im Vordergrund. O'SULLIVAN et al. [OEH02, OE03] stellen hierzu eine Klassifizierung nicht-funktionaler Aspekte auf Basis der Eigenschaften klassischer Dienstleistungen vor (vgl. auch [Tom05]):

- **Verfügbarkeit:** Die Verfügbarkeit bezeichnet die zeitlichen oder räumlichen Rahmenbedingungen, unter denen eine Interaktion mit einem Dienst möglich ist [OEH02]. Die *zeitliche Verfügbarkeit* kann dabei mehrere unterschiedliche Aspekte umfassen [OE03]:
  - Zeitraum der Verfügbarkeit von Dienstinformationen für Verzeichnisse und potentielle Nutzer (*Advertising Window*),
  - Zeitraum für Verhandlung und Vertragsabschluss (*Invocation Window*),
  - Zeitpunkt und Dauer der Ausführung bzw. Auslieferung der Leistung an den Nutzer (*Provision Window*),
  - Zeitraum zur Bezahlung (*Payment Window*),
  - Zeiträume, in denen ein Dienst unterbrochen (*Suspension Window*), wiederaufgenommen (*Resumption Window*) oder der Bezug der Leistung endgültig abgebrochen werden kann (*Cancellation Window*).

  Analog dazu lassen sich die Eigenschaften der räumlichen Verfügbarkeit weiter gliedern [OE03]:
  - Ort, an den die Anfrage zur Nutzung eines Dienstes erbracht werden soll (*Invocation Location*),
  - Ort, an dem sich ein Dienstnutzer befinden muss, um einen Dienst anfragen zu können (*Invoker's Location*),

- ▷ Ort, an dem sich der Dienstanbieter befindet (*Provider's Location*),
- ▷ Orte oder Regionen, an welche die Lieferung der Leistung erfolgen kann (*Provision Location*),
- ▷ Zahlungsort (*Payment Location*),
- ▷ Orte, an denen ein Dienst ausgesetzt (*Suspension Location*), wiederaufgenommen (*Resumption Location*) oder abgebrochen werden kann (*Cancellation Location*).

- **Zugangskanäle:** Zugangskanäle stellen Interaktionsmedien dar, über die sich Dienstanbieter und -nutzer verständigen bzw. die Dienstleistung ausgeführt wird. Für jeden Zugangskanal müssen entsprechende Endpunkte festgelegt, die transferierten Informationen (bzw. Güter) spezifiziert und ggf. ein Kommunikationsprotokoll implementiert sein. Zugangskanäle können selbst wieder über eigene nicht-funktionale Eigenschaften verfügen, wie z. B. unterschiedliche Verfügbarkeit, Bandbreiten oder Preiskategorien [OEH02].

- **Qualität:** Die Qualität bezeichnet das Verhältnis zwischen der vom Nutzer erwarteten und der tatsächlich vom Anbieter gelieferten Leistung. Sie umfasst eine (zumeist subjektive) Wahrnehmung der funktionalen und nicht-funktionalen Eigenschaften [OEH02].

- **Vertrauen und Sicherheit:** Diese Aspekte umfassen Eigenschaften zum Schutz von Identität, Privatsphäre und sensiblen Daten des Dienstnutzers, die Sicherheit von Übertragungskanälen, den Schutz vor Fälschungen, Änderungen und unerlaubtem Zugriff auf den Dienst sowie Maßnahmen und Kennzahlen zur Bildung von subjektivem Vertrauen.

- **Rechte:** Die Rechte von Dienstnutzern und -anbietern umfassen Möglichkeiten zum Rücktritt, zum vorzeitiger Abbruch, zur Aussetzung der Leistung (*Suspension*) oder zur Wiederaufnahme der Leistung (*Resumption*) [OEH02, Tom05].

- **Art der Vertragsbindung:** Mögliche Vetragsmodelle für die Nutzung von Diensten spezifizieren zum Beispiel, ob es sich um eine einmalige Leistung oder eine längerfristige Bindung (z. B. im Rahmen eines Abonnements) handelt [OEH02].

- **Abrechnung und Bezahlung:** Neben den monetären Kosten für die Nutzung eines Dienstes stehen hierbei die möglichen Abrechnungsraten (z. B. Pay-per-Use, Flatrate), Zahlungsarten (z. B. Überweisung, Kreditkarte) sowie Währungsinformationen im Vordergrund [OEH02, Tom05].

In Bezug auf diese Kriterien treten nicht-fachliche Ausnahmesituationen insbesondere dann ein, wenn für die Ausführung von Aktivitäten ausgewählte Ressourcen zeitlich oder räumlich nicht verfügbar sind oder die Ausführung

der Aktivität nicht den qualitativen Erwartungen an die Erfüllung der Aufgabe entspricht. Insbesondere in dynamischen Umgebungen kann es vorkommen, dass bei einer frühen Bindung die gewählte Implementierung für eine Aktivität zum Ausführungszeitpunkt nicht (mehr) zugreifbar ist. Der Austausch von unvorteilhaften, fehlerhaften oder nicht verfügbaren Diensten gegen funktional und nicht-funktional gleichwertigen oder höherwertigen Ersatz stellt daher ein großes Anwendungsgebiet für die Anpassung von dienstbasierten Prozessen dar (vgl. [DS05]).

Die Grundlagen für eine solche Auswahl von Dienstimplementierungen stellen die Beschreibung der jeweiligen Funktionen und Kontexte von Seiten des Dienstanbieters, die Beschreibung von Anforderungen von Seiten des Dienstnutzers sowie einen geeigneten Algorithmus zum Abgleich der Eigenschaften dar (vgl. Abschnitt 3.4.2). Hierfür können in Verbindung mit *Policies* und *Service Level Agreements (SLAs)* eindeutige Identifikatoren oder kontextbasierte Ontologien eingesetzt werden, um ein gemeinsames Verständnis der betreffenden Eigenschaften zu erlangen [NG06, NBNK05, Kun08]. Im Rahmen von Prozessen muss bei der Auswahl von geeigneten Diensten zwischen den angestrebten und ggf. vereinbarten Qualitätseigenschaften der komplexen prozessorientierten Anwendung (*globale Ebene*) und den Eigenschaften der einzelnen involvierten Dienste (*lokale Ebene*) unterschieden werden [ZBHN+04, AP06]. Soll der Prozess z. B. innerhalb einer vorgegeben Höchstdauer ausgeführt werden, so ist es wichtig, dass die einzelnen ausgewählten Dienste auf dem kritischen Pfad des Prozesses in der Summe ihrer Ausführungszeiten diese Vorgabe nicht überschreiten. Das *Aggregationsverhalten* [Aag01] einer Eigenschaft beschreibt dazu, wie sich die Eigenschaft in einer Komposition verhält (vgl. [AP06] zu Details).

Für eine Optimierung der Prozessausführung sind als wesentliche technische Parameter der Prozessausführung insbesondere die *Ausführungsdauer*, *monetäre Kosten* und *Verfügbarkeit* der prozessorientierten Anwendung relevant [ZBHN+04]. In Hinblick auf die Zuweisung von Aufgaben an menschliche Akteure schlagen WIELAND et al. die Erweiterung des *Human Task Managements* um eine kontextbasierte Auswahl von Mitarbeitern vor, welche sich zum Beispiel auf den Abgleich des derzeitigen *Aufenthaltsortes von Mitarbeitern* und den *vorgesehenen Ort der Aufgabenerledigung* stützt. Weiterhin wird die Integration von Kontextdaten zur Unterstützung von Mitarbeitern vorgesehen, indem Tätigkeitsbeschreibungen mit aktuellen *Daten zu beweglichen Objekten* angereichert werden. So kann zum Beispiel eine nahe Bezugsquelle von Werkzeugen oder Maschinen angeben werden, um den mit der Ausführung der Aufgabe zusammenhängenden Overhead zu minimieren und damit die Prozessausführung zu beschleunigen [WKNL07].

Für die Integration von Context Awareness zur nicht-funktionalen Anpassung der Prozessausführung werden geeignete Sensoren benötigt, um die relevanten Daten zu erheben. Hierbei können insbesondere die verfügbaren Ressourcen mit ihren Eigenschaften als Teil des relevanten Kontextes angesehen werden. Etwaige Verzeichnisse über Dienste und Mitarbeiter stellen daher in

dieser Sichtweise eine potentielle Komponente von unterstützenden Kontextmanagementsystemen dar [Kun08].

3.5.5 Anpassung von Benutzungsschnittstellen

Einen besonderen Stellenwert im Rahmen der nicht-funktionalen Anpassung prozessorientierter Anwendungen nehmen menschliche Akteure und ihre Interaktion mit dem System ein. Dabei kann generell zwischen der Rolle des *Konsumenten* der prozessorientierten Anwendung und der Rolle des daran mitarbeitenden *Prozessteilnehmers* unterschieden werden [AFG+07]. Eine Anpassung des Prozesses an den Kontext des Konsumenten macht oft eine inhaltliche oder qualitative Änderung notwendig [AFG+07] (vgl. Abschnitte 3.5.3 und 3.5.4). Hingegen steht bei der Ausführung von Aktivitäten, welche die Beteiligung oder vollständige Bearbeitung durch einen menschlichen Mitarbeiter erfordern, eine angemessene Anpassung der *Repräsentation der Benutzungschnittstelle* an den aktuellen Kontext der konkret zugewiesenen Person im Vordergrund der Betrachtung [MBCP05, AFG+07, CELS07].

Eine Benutzungsschnittstelle stellt allgemein eine Komponente dar, welche der Interaktion zwischen Mensch und Maschine dient und daher möglichst gut auf die Bedürfnisse des Benutzers und auf die auszuführende Funktion angepasst sein muss [DFAB04]. Bei der Interaktion mit einer prozessorientierten Anwendung zur Ausführung einer Aktivität umfasst dies einerseits die *Präsentation von Ausgabeparametern* (z. B. die Beschreibung der durchzuführenden Aufgabe) und ggf. die *Entgegennahme von Eingabeparametern* des menschlichen Akteurs (z. B. eine Bestätigung oder eine Liste von gesammelten Daten) [AGP08]. Die Interaktion findet dabei in der Regel im natürlichen Arbeits- und Lebensumfeld von Menschen statt, d. h. zum Beispiel im Büro, in einer Werkhalle, unterwegs auf einer Dienstreise oder zu Hause. Bei der Anpassung von Benutzungsschnittstellen im Kontext prozessorientierter Anwendungen sind daher folgende Aspekte besonders zu beachten:

- **Geräteeigenschaften:** Die Benutzungsschnittstelle muss an die Eigenschaften des Endgeräts, welches zur Ausführung der interaktiven Aufgabe verwendet wird, angepasst werden. Für die Ausführung von Prozessen sind insbesondere stationäre Arbeitsplatzrechner und Terminals, aber auch mobile Geräte wie Notebooks, PDAs oder leistungsfähigere Mobiltelefone (z. B. Smartphones) relevant. Die Präsentation von Aufgaben sowie die Ein- und Ausgabemöglichkeiten müssen daher an die Größe, die Auflösung und die Darstellung (z. B. Farbtiefe) der verfügbaren Displays sowie an die angeschlossene Peripherie (z. B. Tastaturen, Drucker, Scanner, Kameras) angepasst werden. Des Weiteren spielt die Leistungsfähigkeit der Geräte in Bezug auf Rechengeschwindigkeit und Speicherkapazität sowie die verfügbare Bandbreite der unterstützten Netzwerkverbindungen (z. B. Ethernet, Wireless LAN, Bluetooth) eine Rolle [MBCP05, CELS07].

- **Rolle des interagierenden Nutzers:** Berechtigungen, Kompetenzen und Erfahrungen von Benutzern können erheblichen Einfluss auf die zur Ausführung einer Aufgabe notwendigen Informationen und ihre Darstellung haben. Der Detaillierungsgrad der Aufgabenbeschreibung, zusätzliche Hintergrundinformationen oder Hilfestellungen müssen daher auf die Rolle des Benutzers angepasst werden [AFG+07, AGP08].

- **Ort der Nutzung:** Zusätzlich zur Rolle des Benutzers kann auch der Ort der Nutzung die Aufbereitung von Informationen zur Ausführung einer Aktivität beeinflussen. Informationen, die nur am Arbeitsplatz verfügbar sind, müssen dem Benutzer ggf. zugänglich gemacht werden, wenn dieser z. B. von zu Hause oder von unterwegs an der Ausführung eines Prozesses beteiligt ist. Falls am Arbeitsort in Verbindung mit dem genutzten Gerät keine Netzwerkverbindungen verfügbar sind, müssen die zu bearbeitenden Aktivitäten und der relevante Kontrollfluss des Prozesses *offline* verfügbar gemacht und entsprechend später synchronisiert werden [MBCP05, CELS07].

- **Aktivität und persönliche Eigenschaften des Benutzers:** In weiterführenden Szenarien kann auch die aktuelle Aktivität des Benutzers Einfluss auf die Darstellung bzw. die Modalitäten von Aufgabenbeschreibungen haben. Führt der Benutzer z. B. aktuell eine Aktivität aus, welche seine visuelle Aufmerksamkeit verlangt (z. B. das Führen eines Kraftfahrzeugs oder das Bedienen einer komplexen Maschine), so ist eventuell eine knappe Darstellung der Aufgabe in einer hohen Schriftgröße oder sogar eine Sprachausgabe bzw. -eingabe von Vorteil. Ebenso ist eine solche Anpassung für Benutzer mit körperlichen Beeinträchtigungen (z. B. Blindheit) notwendig (*Barrierefreiheit*) [AGP08, CELS07, Vil08].

- **Physikalische Umgebung:** Schließlich können extreme physikalische Bedingungen wie Dunkelheit oder Kälte eine spezielle Anpassung der Ein- und Ausgabemodalitäten erforderlich machen [AGP08].

Konkrete kontextbasierte Ansätze zur Anpassung von Benutzungsschnittstellen werden aufgrund ihrer Relevanz im Rahmen der dynamisch verteilten Prozessausführung unter Berücksichtigung mobiler Geräte in Abschnitt 5.8.4 genauer betrachtet.

3.5.6 Vorhersage von Kontextdaten

Für eine proaktive Nutzung von Kontextdaten ist eine möglichst genaue Vorherbestimmung zukünftiger Kontexte erforderlich [May05, Fer07]. Bei einer *Kontextdatenprognose* werden dabei auf Basis gegebener Informationen (über Gegenwart und Vergangenheit) Informationen über Kontexte in der Zukunft abgeleitet [SHD07, Mei09]. Hierfür können Kontextdaten eines Kontextmodells basierend auf Entitäten und Attributen (vgl. Abschnitt 3.5.1) als historische Daten dargestellt und als Grundlage für ein Erkennen von Zusam-

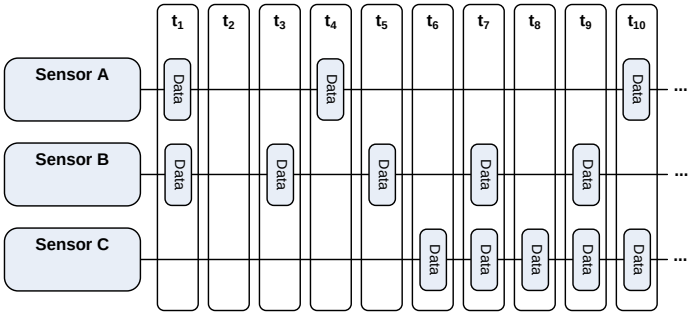


Abbildung 3.13: Beispiel einer konkreten Zeitreihe von Kontextdaten unterschiedlicher Quellen (SHD07)

menhängen für die Prognose von zukünftigen Kontexten einzelner Prozessinstanzen verwendet werden. Des Weiteren werden Techniken des *Data Minings* zur Analyse von Monitoring-Daten für eine längerfristige, evolutionäre Anpassung von Prozessmodellen verwendet [CCSD05, Han05].

Die gesammelten (historischen) Daten beschreiben dabei Daten im zeitlichen Verlauf im Sinne einer *Zeitreihe* (vgl. [ABH85]), wobei im Rahmen der Kontextdatenprognose die betreffenden Daten durch physikalische oder logische Sensoren, durch Kommunikation oder durch Verarbeitung gewonnen werden [Che04, SHD07]. Abbildung 3.13 zeigt ein Beispiel einer solchen Zeitreihe mit Sensordaten, welche abhängig von der Art der Daten, ihrer Verfügbarkeit und der Art der Erhebung in regelmäßigen oder unregelmäßigen Abständen berücksichtigt werden müssen. Die Autoren SIGG et al. sprechen daher auch von *konkreten Zeitreihen* [SHD07].

Die bei der Zeitreihenanalyse üblichen mehrdimensionalen Zustände können explizit in ihre einzelnen Dimensionen zerlegt und als Werte entsprechender *Variablen* angesehen werden [ABH85, Kri99, Mei09]. Variablen besitzen unterschiedliche Wertebereiche bzw. Skalen (z. B. Nominalskala, Ordinalskala, Intervallskala, Ratioskala) [Ste07], welche Auswirkungen auf die Anwendbarkeit und Qualität von Prognosemethoden haben [Sig08]. Deterministische Zeitreihen können durch eine Funktion beschrieben werden und sind theoretisch sicher vorhersehbar, während stochastische Zeitreihen nur durch Wahrscheinlichkeitsverteilungen beschrieben werden können [Kri99]. Hierbei ist insbesondere der Umgang mit Unsicherheit geeignet zu unterstützen (vgl. [Hüb03]).

In vielen Fällen basiert das Verhalten des betrachteten Systems auf *Zusammenhängen*, mit denen eine Vorhersage des zukünftigen Verhaltens dieses Systems ermöglicht werden kann. Zwischen Teilen von historischen Daten besteht ein Zusammenhang, wenn die Wahrscheinlichkeit, dass ein Teil der Daten bestimmte Werte annimmt, davon abhängig ist, welche Werte die anderen Teile besitzen [Mei09]. Mögliche Arten von Zusammenhängen sind Trends, sequentielle oder periodische Muster sowie deterministische, lineare oder kausale Zusammenhänge [And71, ABH85, KS97, SGS00, May04b, Sig08].

Die Erkennung, Abstraktion und Nutzung solcher Zusammenhänge von Kontextdaten wird in aktuellen Verfahren zur Kontextdatenprognose konkretisiert. Bei dem von MAYRHOFER [May04b] vorgeschlagenen Ansatz können komplexe Situationen erkannt und vorausgesagt werden. Hierzu werden zunächst Sensordaten gemessen (*Low-Level-Kontext*), hieraus Merkmale extrahiert und durch Clustering zu einer komplexen Situation (*High-Level-Kontext*) zusammengefasst. Resultierende abstrakte Zustände können dann durch den Benutzer identifiziert und mit einem aussagekräftigen Namen versehen werden (z. B. „zu Hause“ oder „in einer Besprechung“) [May04b]. Bei einer Prognose wird in diesem Fall ein zukünftiger Cluster prognostiziert. Hierfür müssen vorher ausreichend Daten mit erkennbaren Zusammenhängen gesammelt worden sein. Der weitergehende Ansatz von SIGG [Sig08] beschränkt sich nicht nur auf die Prognose von High-Level-Kontext, sondern erlaubt die Ein- und Ausgabe von beiden Kontextarten. Das Verfahren zur Prognose verwendet dazu Vorwissen von gelernten Zusammenhängen auf Basis von Trainingsdaten aus einer Historie, die alle Kontextdaten bis zu einem gewissen Alter speichert. Zur Erhöhung von Effizienz und Genauigkeit der Prognose schlägt PETZOLD [Pet05] die parallele, hybride Verwendung von mehreren Prognosemethoden vor, wobei der Ansatz jedoch auf die Prognose von *Primärkontext* (d. h. Ort, Zeit, Identität und Aktivität) beschränkt ist (vgl. Abschnitt 3.5.1).

Die genannten und ähnliche Verfahren können zur Kontextdatenprognose im Rahmen prozessorientierter Anwendungen verwendet werden, wenn die für die Ausführung des betreffenden Prozesses benötigten domänenabhängigen Kontextvariablen im Voraus bekannt sind oder es sich bei den zu prognostizierenden Daten um generelle Eigenschaften handelt, welche die Ausführung von Prozessen im Allgemeinen betreffen (z. B. die Berücksichtigung der verfügbaren Bandbreite von regelmäßig genutzten Kommunikationsverbindungen oder die Berechnung des am Energiebedarf orientierten Einsatzplans von Maschinen [DSB10]). Diese Voraussetzungen sind notwendig, da in der Regel eine ausreichende Menge von Daten benötigt wird, um komplexere Zusammenhänge erkennen und ein qualitativ ausreichendes Prognoseergebnis bereitstellen zu können [Mei09]. Besondere Anwendung findet die Vorhersage von prozessrelevanten Daten daher bislang bei der Abschätzung von Ausführungszeiten wiederholt ausgeführter Aktivitäten und, darauf aufbauend, der zu erwartenden verbleibenden Ausführungszeit von laufenden Prozessinstanzen [MM07, Rei07, vdASS11].

3.6 Flexibilität durch Entkopplung fachlicher Regeln

Die Erhebung von Kontextdaten und das Erkennen von kritischen Situationen ist eine notwendige, aber nicht hinreichende Voraussetzung für die flexible Anpassung von prozessorientierten Anwendungen. Vielmehr müssen darüber hinaus die beobachteten Geschehnisse geeignet ausgewertet und mit zielgerichteten Anpassungsmaßnahmen in Verbindung gebracht werden. Hierfür können für prozessorientierte Anwendungen im Allgemeinen *regelbasierte Sy-*

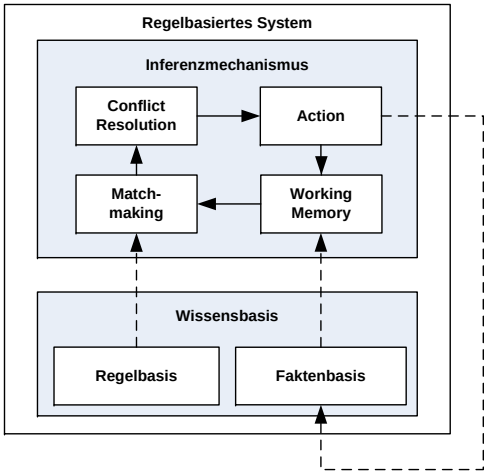


Abbildung 3.14: Architektur eines regelbasierten Produktionssystems (nach (KT01a, BKI08)

steme zum Einsatz kommen, welche einerseits als Entscheidungsgrundlage für die Auswahl von Kontrollflussverzweigungen und andererseits für die benutzerdefinierte Überwachung von Prozessausführungen (z. B. durch einfache Zeitbeschränkungen oder komplexe Dienstgütevereinbarungen) Anwendung finden.

Hinsichtlich einer Flexibilisierung von prozessorientierten Anwendungen ist hierbei insbesondere die Entkopplung der Spezifikation solcher Regeln von den einzelnen Prozessbeschreibungen interessant. Dabei werden die zu einer konkreten Entscheidung führenden *(Geschäfts-)Regeln* in einem externen Regelwerk festgehalten und durch ein spezialisiertes Verwaltungssystem (*Business Rule Management System*, kurz *BRMS*) ausgewertet. Hintergrund dieser Maßnahme ist, dass sich die Regeln zur Findung einer Entscheidung oft häufiger ändern als der Kontrollfluss der prozessorientierten Anwendung oder dass diese von der eigentlichen Prozesslogik relativ unabhängig sind [CM04, RD05].

Dieser Abschnitt geht nach einer kurzen Einführung der Grundlagen regelbasierter Systeme auf die Flexibilisierung von prozessorientierten Anwendungen durch die Integration fachlicher Regeln ein und zeigt relevante Anwendungsgebiete hierfür auf.

3.6.1 Regelbasierte Systeme

Ein regelbasiertes System ist ein Wissensverarbeitungssystem, welches neues Wissens aus bereits bekanntem Wissen auf der Basis von Regeln ableitet. Regeln sind formalisierte Konditionalsätze der Form *WENN p DANN q* [KT01a]. Der WENN-Teil der Regel p wird als *Prämisse* oder *Antendenz*, der DANN-Teil q als Konklusion oder *Konsequenz* bezeichnet [KT01a, BKI08]. Wenn die Konsequenz einer Regel mit einer Aktion verbunden ist, spricht man von einer *Produktions-* oder *Aktionsregel* [BKI08]. Ist die Prämisse einer Regel erfüllt, so sagt man, dass „die Regel feuert“ [DEF+08].

Ein regelbasiertes (Produktions-)System besteht aus *Fakten*, *Regeln* und einem *Inferenzmechanismus* (*Inference Engine*) (vgl. Abbildung 3.14). Fakten sind alle Informationen, welche dem regelbasierten System bereitgestellt werden oder während der Regelverarbeitung von den Regeln erzeugt werden. Die Fakten und Regeln bilden zusammen die *Wissensbasis*. Der Teil der Wissensbasis, der die Fakten enthält, wird als *Faktenbasis* bezeichnet, während die *Regelbasis* die Regeln des Systems enthält. Die Definition der Regeln erfolgt mit Hilfe einer *Regelsprache* [Hal01, KT01a].

Der Inferenzmechanismus ist ein Kontrollsystem mit Regelinterpreter und kontrolliert die Anwendung der Regeln auf die ihm bekannten Fakten (*Working Memory*). Dazu bestimmt er mit Hilfe eines *Musterabgleichs* (*Pattern Matching*), welche Regeln anwendbar sind (*Konfliktmenge*), wobei alle möglichen Kombinationen von Fakten getestet werden. Sind mehrere Regeln gleichzeitig erfüllt, wird die Konfliktmenge anhand einer vorgegebenen Strategie (*Conflict Resolution Strategy*) geordnet und die erste Regel der resultierenden Liste (*Agenda*) ausgeführt. Da sich durch die Ausführung einer damit verbundenen Aktion (*Action*) die Faktenbasis verändern kann, wiederholt sich der beschriebene Vorgang gegebenenfalls [KT01a].

3.6.2 Beschreibung von Geschäftsregeln

In Hinblick auf den dargestellten Prozesslebenszyklus (vgl. Abbildung 2.8) und dessen mögliche Einteilung in fachliche und technische Betrachtungsebenen kommt der Anpassung von prozessorientierten Anwendungen durch unterschiedliche Zielgruppen eine große Bedeutung zu. Bedingung für ein hohes Maß an fachlicher Flexibilität und Erweiterbarkeit ist es daher, dass Änderungen von der Fachseite weitestgehend ohne technisches Hintergrundwissen durchgeführt werden können [Hal01, RD05, Ses10]. Gerade bei einer hohen Volatilität von fachlichen Rahmenbedingungen ist dies jedoch nur möglich, wenn die fachliche *Geschäftslogik* von der *Programmlogik* bzw. *Prozesslogik* entkoppelt und getrennt verwaltet werden kann. Ziel einer solchen Entkopplung ist es, grundlegende Änderungen der fachlichen Rahmenbedingungen zu ermöglichen, ohne Änderungen am Programm-Code bzw. am technischen Prozessmodell vornehmen zu müssen [SG06].

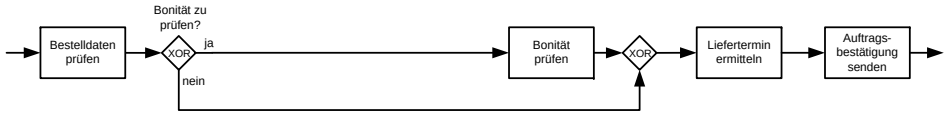
Die Verwendung von *Geschäftsregeln* stellt eine Möglichkeit dar, um fachliche Bedingungen *deklarativ* auszudrücken und an zentraler Stelle adressatengerecht zu beschreiben [RD05, Gra07]. Die *Business Rules Group* [HH00] definiert eine Geschäftsregel (*Business Rule*) als eine Direktive oder Richtlinie, die das fachliche Verhalten eines Systems beeinflussen oder leiten soll. Dabei werden konkrete Vorgaben beschrieben, nach denen sich ein Computerprogramm (bzw. eine prozessorientierte Anwendung, welche durch ein Computerprogramm gesteuert wird) verhalten soll. Da es sich bei einer solchen Vorgabe nicht notwendigerweise um Bedingungen im Unternehmenskontext handeln muss, kann hierbei auch verallgemeinernd von einer *fachlichen Regel* gesprochen werden [BD00, CM04, Gra07].

Fachliche Rahmenbedingungen sind oft nur als implizites Wissen in einer Organisation vorhanden und müssen zunächst einmal identifiziert und explizit als Geschäftsregeln niedergeschrieben werden. Als Ausgangsbasis kann hierfür eine natürlichsprachliche Darstellung auf Basis von Textschablonen verwendet werden, welche später in eine automatisch verarbeitbare Form von eindeutigen Konditionalsätzen oder Formeln transformiert wird [SG06, Bus08]. In der Literatur werden im Allgemeinen folgende Arten von Regeln zu den Geschäftsregeln gezählt [HH00, Hal01, Ros03, RD05, SG06]:

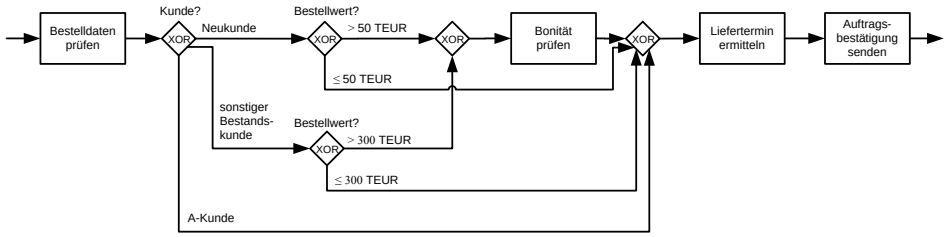
- **Integritätsregeln:** Integritäts- oder auch Plausibilitätsregeln bestimmen, ob die in eine Anwendungssoftware eingegebenen Daten in fachlicher Hinsicht vollständig und gültig sind [RD05]. Die Bestellung einer Ware erfordert zum Beispiel, dass der Kunde eine gültige Lieferadresse eingegeben hat.
- **Einschränkungen:** Eine Einschränkung stellt eine Aussage dar, welche den gültigen Wertebereich einer Eingabe begrenzt [Ros03, SG06]. Die Regel, dass sich eine Lieferadresse in Deutschland befinden muss, stellt zum Beispiel eine solche Einschränkung dar.
- **Berechnungsregeln:** Eine Berechnungsregel oder auch Ableitungsregel [HH00, RD05, SG06] stellt basierend auf der Auswertung von Bedingungen einen geeigneten Algorithmus zur Berechnung eines Terms zur Verfügung. Durch die Berechnung wird somit neues Wissen abgeleitet. Beispiel ist die Berechnung eines Preises aufgrund von Alter und Status des Kunden, des Bestellzeitraums und der Art und Menge der bestellten Ware [Hal01, Ros03, RD05].
- **Reaktive Regeln:** Reaktive Regeln initiieren eine Aktion, wenn die Bedingungen der Regel erfüllt werden [Hal01, Ros03]. Im Rahmen der oben genannten Produktionsregeln würde die mit der Konsequenz einer Regel verbundene Aktion ausgeführt werden. Ein Beispiel ist das Versenden der Ware, wenn der Besteller die Bedingung erfüllt, 18 Jahre alt zu sein. Ebenso werden bei Regeln nach dem Prinzip *Event-Condition-Action (ECA)* Aktionen auf der Basis von *Ereignissen* ausgelöst [KEP00, CM04, RD05, SG06]. Ein Beispiel ist das Zusenden einer Geburtstagskarte, wenn das Ereignis eintritt, dass der Kunde 18 Jahre alt wird (vgl. [BBB+07]).

Abbildung 3.15 zeigt ein einfaches Beispiel für die Anwendung von Geschäftsregeln bei der Prozessmodellierung und -ausführung (vgl. [FRH10]). Es handelt sich hierbei um einen fiktiven Prozess, der über die Umstände der Bestellannahme entscheidet. Insbesondere soll unter bestimmten fachlichen Bedingungen die Bonität des Kunden geprüft werden. Es sei angenommen, dass hierfür u.a. folgende beispielhafte Geschäftsregeln identifiziert werden können [FRH10]:

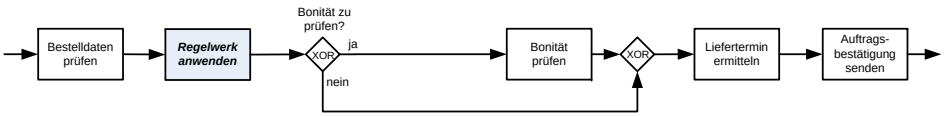
- Eine Prüfung erfolgt, wenn der Bestellwert 300.000 EUR übersteigt.



(a) Prozess zur Bestellannahme mit Prüfung der Bonität unter bestimmten (nicht modellierten) Umständen



(b) Prozess zur Bestellannahme mit Modellierung der Umstände zur Bonitätsprüfung



(c) Prozess zur Bestellannahme mit Referenzierung von Geschäftsregeln

Bedingungen		Entscheidung
Kundentyp	**Bestellhöhe**	**Bonität zu prüfen?**
A-Kunde	egal	nein
Sonstiger Bestandskunde	> 300.000 EUR	ja
	≤ 300.000 EUR	nein
Neukunde	> 50.000 EUR	ja
	≤ 50.000 EUR	nein

(d) Geschäftsregeln in Form einer Entscheidungstabelle

Abbildung 3.15: Beispiel zur Anwendung von Geschäftsregeln für prozessorientierte Anwendungen (FRH10)

- Eine Prüfung erfolgt, wenn der Kunde ein Neukunde ist und der Bestellwert 50.000 EUR übersteigt.
- Eine Prüfung erfolgt nicht, wenn der Kunde ein „A-Kunde“ ist.

Der Prozess in Abbildung 3.15a visualisiert den allgemeinen Ablauf der Bestellabwicklung mit dem vorgegebenen Kontroll- und Datenfluss des Anwendungsfalls. Das Wissen über die oben angegebenen Geschäftsregeln ist hier nicht explizit formalisiert. Eine automatisierte Bearbeitung dieses Prozesses ist in Hinblick auf das dem Prozess nur implizit zugrunde liegende fachliche Wissen zur Bearbeitung der Kontrollflussentscheidung daher nicht ohne weiteres möglich. Im Prozess in Abbildung 3.15b wurden die genannten fachlichen Bedingungen als jeweils alternative Kontrollflusspfade modelliert. Der Prozess verfügt nun zwar über das relevante Anwendungswissen, um eine automatische Ausführung zu ermöglichen. Jedoch hat durch die Integration dieser Regeln seine Komplexität stark zugenommen, was sich negativ auf Lesbarkeit, Wartbarkeit und Anpassbarkeit des Prozesses auswirkt. Insbesondere die Verschachtelung von Bedingungen wird durch jeden weiteren zu beachtenden

fachlichen Aspekt weiter verschärft. Jede Änderung an den Bedingungen erfordert zudem eine komplexe Anpassung der umgebenden Transitionen, Split- und Join-Konstrukte sowie ggf. der Aktivitäten. Des Weiteren ergeben sich erhebliche Redundanzen, falls die betreffenden Bedingungen auch in weiteren Prozesspfaden oder anderen Prozessen geprüft werden [FRH10]. Die beiden in den Abbildungen 3.15a und 3.15b dargestellten Varianten zum Umgang mit Geschäftsregeln entsprechen den in Abschnitt 3.3.1 genannten Möglichkeiten zur Verwirklichung der *A-priori-Flexibilität* durch Abstraktion bzw. durch Modellierung aller denkbaren Prozessverläufe.

Eine einfache und flexible Verwaltung der Geschäftsregeln wird für viele Organisationen als kritischer Erfolgsfaktor angesehen und stellt daher die Motivation für eine strikte Trennung von *Geschäftsregeln* und *Routing-Regeln* dar [FRH10]. Dabei stellen die Geschäftsregeln im Kontext einer prozessorientierten Anwendung die Vorgaben dar, nach denen Entscheidungen getroffen werden, während die Routing-Regeln die Durchführung des Kontrollflusses auf Basis des Ergebnisses dieser Entscheidung umsetzen [FRH10]. Im einfachsten Fall kann die Aufgabe, eine solche fachliche Entscheidung zu treffen, als Aktivität innerhalb des Prozesses modelliert werden und dabei einen entsprechenden Mechanismus zur Interpretation und Auswertung der aktuell gültigen Geschäftsregeln referenzieren. Abbildung 3.15c zeigt den Beispielprozess mit einer solchen sogenannten *Geschäftsregelaufgabe* (*Business Rule Task*) [FRH10]. Das Ergebnis der Regelauswertung kann wiederum als Prozessdaten (z. B. als einfache boolesche Variable) integriert und an einer nachfolgenden Verzweigung des Prozesses automatisiert ausgewertet werden. Weitergehende Ansätze erfordern die Erweiterung von Prozessbeschreibungssprachen zur Integration von komplexen Regelwerken und entsprechender Vor- und Nachbedingungen für Aktivitäten [CM04, RD05].

Als Nachteil solcher fachlicher Regeln ist ein zusätzlicher Kostenfaktor in Form der Notwendigkeit zusätzlicher Software zur Regelverwaltung zu nennen. Für die Modellierung und Verwaltung von Geschäftsregeln können dabei verschiedene Medien zum Einsatz kommen. Einfache Regeln können mit Hilfe von Entscheidungsbäumen oder Entscheidungstabellen spezifiziert und entweder manuell oder automatisch ausgewertet werden. Abbildung 3.15d zeigt eine mögliche Umsetzung der Geschäftsregeln zu dem illustrierten Beispiel als Entscheidungstabelle (vgl. [FRH10]). Für komplexere Regelwerke bietet sich die Unterstützung der Regelverwaltung durch ein *Geschäftsregel-Managementsystem* (*Business Rule Management System*, kurz *BRMS*) an [Gra07]. Eine mögliche Anbindung von regelbasierten Systemen kann dabei zum Beispiel über entsprechende Dienste einer dienstorientierten Architektur erfolgen [RD05, Gra07].

Das angemessene Entkoppeln der Informationen, die konkret der Entscheidung über die Fortführung des Kontrollflusses dienen, birgt eine hohe Flexibilität. Geschäftsregeln können logisch losgelöst von einem bestimmten Prozessmodell und ggf. sogar physisch auf einem anderen System verwaltet werden. Dies erlaubt die vom Prozessmodell unabhängige Änderung, Erweiterung

oder Reduzierung der Regeln. Unter Umständen können Geschäftsregeln auf diese Weise sogar zur Laufzeit einzelner Prozessinstanzen angepasst werden [Luc02].

Neben der durch die logische bzw. physische Entkopplung erreichbaren Flexibilität ist außerdem die *Wiederverwendbarkeit* der definierten Regeln als besonderer Vorteil zu nennen [Ros03, Gra07]. So müssen relevante Entscheidungsgrundlagen nicht an mehreren Stellen redundant im Prozess vorgehalten werden, sondern können zentral verwaltet und somit ggf. sogar von mehreren verschiedenen Prozessen referenziert werden. Die Entkopplung von Geschäftsregeln wirkt sich also insgesamt sehr positiv auf die Wartbarkeit von prozessorientierten Anwendungen aus [CM04].

3.7 Flexibilität durch ereignisgesteuerte Systeme

Wie in den beiden vorhergehenden Abschnitten beschrieben wurde, ist die Wahrnehmung, Verarbeitung und Vorhersage von relevanten Umgebungsbedingungen eine wesentliche Voraussetzung, um adäquate Entscheidungen über die Notwendigkeit und ggf. die Art und Weise von Anpassungen prozessorientierter Anwendungen zu treffen. Eine Erhebung von Daten an fest vorgegebenen Zeitpunkten birgt jedoch die Gefahr, dass veränderte Umgebungsbedingungen nicht zeitnah wahrgenommen und behandelt werden können, was unter Umständen eine proaktive Anpassung verhindert oder sogar eine zeitlich adäquate reaktive Anpassung unmöglich macht. Der zeitnahen Übertragung und Verarbeitung von relevanten Zustandsänderungen sowie die Möglichkeit zur sofortigen Ausführung der dazugehörigen Anpassung kommt daher eine hohe Bedeutung zu.

Eine Möglichkeit zur Flexibilisierung prozessorientierter Anwendungen bietet in diesem Zusammenhang die Verknüpfung der beiden Disziplinen des Prozessmanagements und der ereignisgesteuerten Systeme sowie das darauf basierende Forschungsgebiet des *ereignisgesteuerten (Geschäfts-) Prozessmanagements* (engl. *Event-driven (Business) Process Management*, kurz *ED-BPM*) [vA09, vAEE+09, vAEE+10]. Hierbei tragen insbesondere die ereignisbasierte Kommunikation und die individuelle Verarbeitung von Ereignissen im Rahmen von ereignisgesteuerten Architekturen zur Realisierung einer losen Kopplung zwischen ursächlichen Ereignissen und den dazugehörigen Anpassungen einer prozessorientierten Anwendung bei. In diesem Abschnitt der Arbeit werden die wichtigsten Voraussetzungen und Möglichkeiten solcher ereignisgesteuerter Ansätze vorgestellt und diskutiert.

3.7.1 Ereignisse

Ereignisgesteuerten Systemen unterliegt aus der Sicht der Softwaretechnik ein Programmierkonzept, bei dem ein Softwareprogramm nicht (ausschließlich) linear durchlaufen wird, sondern vordefinierte Programmfragmente (Ereignisbehandlungsroutinen) ausgeführt werden, sobald ein bestimmter Vor-

gang beobachtet wird. Ein *Ereignis* (engl. *Event*) stellt dabei die softwaretechnische Repräsentation einer bestimmten Aktivität eines Systems dar [Luc02]. Eine *Ereignis-Quelle* generiert bei Änderung des *Zustands* eines Objekts im System eine *Benachrichtigung* (*Event Notification*), welche von mehreren *Ereignis-Senken* innerhalb oder außerhalb des Systems empfangen und verarbeitet werden kann. Ereignis-Quellen und Ereignis-Senken sind dabei lose miteinander gekoppelt, so dass Ereignisse ohne Kenntnis der Art und Anzahl der (potentiellen) Ereignis-Senken generiert werden können. Das zugrunde liegende softwaretechnische Konzept entspricht dem Entwurfsmuster des *Beobachter-Musters* (*Observer-Pattern*) [DGH03, GHJJ10].

Um Ereignisse zur Steuerung des Kontrollflusses zu nutzen, müssen die relevanten Zustände der Objekte eines Systems zunächst einmal wahrgenommen, d. h. durch physikalische oder logische Sensoren (vgl. Abschnitt 3.5.1) erhoben werden. Die Beobachtung selbst sollte dabei das Verhalten des Systems nicht beeinflussen [Luc02]. Weicht während der Beobachtung der Zustand des Systems in einer festgelegten Art und Weise (z. B. auf Basis von zuvor definierten Schwellwerten) von dem vorhergehenden Zustand ab, wird die Beobachtung in ein Ereignis transformiert. Nach Luckham besitzt ein Ereignis im Allgemeinen drei zu berücksichtigende Aspekte [Luc02]:

- **Struktur**: Die Struktur eines Ereignisses umfasst dessen Aufbau aus (Daten-)Elementen und Attributen. Beispiele sind Identifikatoren für die Art des Geschehnisses und das von der Zustandsänderung betroffene Objekt sowie die Beschreibung von Zeitpunkt und Ort der wahrgenommenen Aktivität.
- **Bedeutung:** Ein Ereignis repräsentiert eine bestimmte Aktivität bzw. Zustandsänderung. Die Zuordnung des Ereignisses zu dem ursächlichen Geschehnis gibt somit dessen (semantische) Bedeutung an.
- **Relativität:** Die Relativität eines Ereignisses beschreibt dessen Beziehungen zu anderen Ereignissen hinsichtlich Zeit, Kausalität und Aggregation. Ereignisse besitzen dabei die gleichen Beziehungen zueinander wie die ihnen zugrunde liegenden Geschehnisse.

In verteilten Systemen werden Ereignisse in der Regel durch *Nachrichten* übertragen [Ham05, Luc02]. Der folgende Abschnitt zeigt ein auf dem Beobachter-Muster aufbauendes Kommunikationsmodell, um Ereignis-Quellen und Ereignis-Senken zeitlich und räumlich voneinander zu entkoppeln und somit ein hohes Maß an Flexibilität sowohl bei der Art und Anzahl der emittierten Ereignisse auf der Seite des Senders als auch bei deren Auswahl und Verarbeitung auf der Seite des Empfängers zu ermöglichen.

3.7.2 Ereignisbasierte Kommunikation

Die Kommunikation zwischen Ereignis-Quellen und Ereignis-Senken wird in Anlehnung an entsprechende Vorgehensweisen der Realwelt in der Regel über

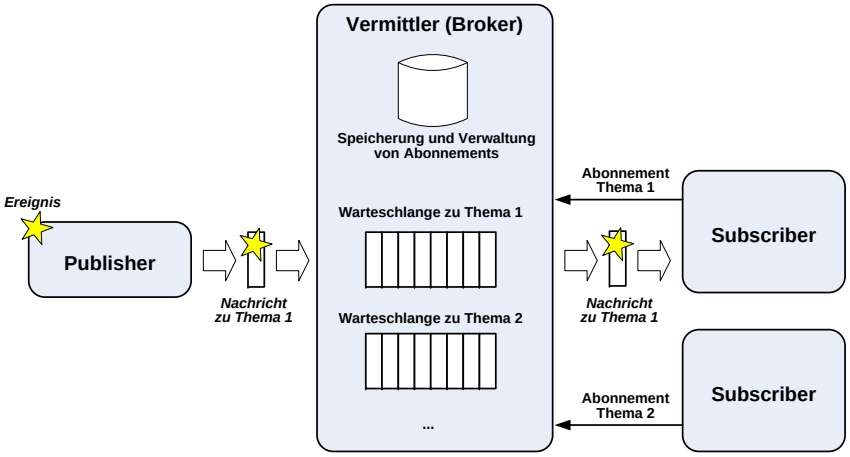


Abbildung 3.16: Abonnementsystem zur ereignisbasierten Kommunikation (nach (EFGK03, Ham05))

ein *Abonnementsystem* (*Publish-Subscribe-System*) [EFGK03, Ham05, TS08] geregelt. Basierend auf einem asynchronen Interaktionsmodell können hierbei konsumierende Teilnehmer (*Subscriber*) Nachrichten zu einem bestimmten Thema (*Topic*) abonnieren, welche sukzessiv durch publizierende Teilnehmer (*Publisher*) generiert werden. Wird durch einen publizierenden Teilnehmer ein neues Ereignis veröffentlicht, so werden nachfolgend alle für diese Art von Ereignis registrierten Konsumenten benachrichtigt. Die Koordination der Nachrichtenvermittlung wird dabei von einem Vermittler (*Broker*) übernommen, welcher die Abonnements der Konsumenten verwaltet, Nachrichten von publizierenden Teilnehmern entgegen nimmt, den Inhalt der Nachrichten ggf. in ein bestimmtes Thema einordnet und an die registrierten konsumierenden Teilnehmer weiterleitet [EFGK03, Ham05].

Zur Umsetzung einer vermittelnden Infrastruktur kommt in der Regel eine *nachrichtenorientierte Middleware* (*Message Oriented Middleware*, kurz *MOM*) mit themenspezifischen Warteschlangen zum Einsatz (vgl. [Ham05, DEF+08]), so dass Ereignisse reihenfolgeabhängig zwischengespeichert werden können. Dies erlaubt neben der referentiellen Entkopplung von Sender und Empfänger zusätzlich eine zeitliche Entkopplung der Kommunikation, so dass Sender und Empfänger nicht zwingend gleichzeitig aktiv sein müssen [TS08]. Abbildung 3.16 zeigt das Modell der ereignisbasierten Kommunikation mit einem solchen Vermittler auf Basis eines themenspezifischen Warteschlangensystems. Um den Anforderungen komplexer Prozessmanagementsysteme zu entsprechen, erweitern höherwertige Architekturen das dargestellte einfache Kommunikationsparadigma um weitere Verwaltungskomponenten, wie zum Beispiel zur Korrelation, Katalogisierung oder zur längerfristigen Speicherung und Analyse von Ereignissen (vgl. [BH05]).

Durch die dargestellte Entkopplung können sich ereignisgesteuerte Systeme gut an Veränderungen anpassen. Ein wichtiger Aspekt der ereignisbasierten Kommunikation ist hierbei jedoch die Auswahl und Zustellung von *relevanten*

Ereignissen. Dabei hat sich in einigen Fällen die Klassifikation von Ereignissen nach bestimmten vordefinierten Themen und deren Anordnung in hierarchischen Themenbäumen als zu ausdrucksschwach und unflexibel erwiesen [EFGK03]. Erweiterte Modelle bieten daher die Möglichkeit einer *inhaltsgesteuerten Zuordnung* (*Content-based Publish-Subscribe*). Hierbei werden Ereignisse nicht nach einem externen Kriterium klassifiziert (wie zum Beispiel dem Namen eines Themas), sondern werden auf Basis ereignisinterner Attribute, Datenstrukturen oder Metadaten ausgewählt. Die konkrete Zuordnung durch den Vermittler wird in diesem Fall durch vom Konsumenten beschriebene Filter realisiert, für die eigene Beschreibungssprachen existieren (vgl. [EFGK03]). Im Gegensatz zur Nutzung vordefinierter Themen ergibt sich hierdurch für den Konsumenten ein höherer Grad an Individualität bei der Auswahl von Ereignissen, so dass keine für den Nutzer uninteressanten Ereignisse übertragen und auf Empfängerseite erneut gefiltert werden müssen. Aus Effizienzgründen werden jedoch oft auch hybride Ansätze eingesetzt, welche individuelle Filtermechanismen auf Ereignisse vordefinierter Themen in Form von Typhierarchien anwenden (*Type-based Publish-Subscribe*) [EFGK03].

3.7.3 Ereignisgesteuerte Architekturen

Basierend auf dem asynchronen Versenden von Ereignissen nach dem Publish-Subscribe-Prinzip stellt eine *ereignisgesteuerte Architektur* (*Event-driven Architecture*, kurz *EDA*) eine verteilte Softwarearchitektur dar, bei der das Zusammenspiel der lose gekoppelten Komponenten auf dem Erkennen, Verarbeiten und Behandeln von Ereignissen beruht [DEF+08, BD10]. Die Ereignisse stellen somit in diesem Architekturstil das Mittel der Kommunikation dar (vgl. Abschnitt 3.7.2). Ereignisse werden hierbei von Sensoren festgestellt, durch einen Ereignis-Produzenten veröffentlicht, durch ein Regelwerk individuell von Ereignis-Konsumenten verarbeitet und bei Bedarf zu einer passenden Reaktion abgeleitet. Dabei ist jede teilnehmende Komponente in Bezug auf das Versenden, Empfangen und Verarbeiten von Ereignissen autonom [DEF+08, BD10].

Zu einer ereignisgesteuerten Architektur gehören zusammengefasst drei wesentliche Systemkomponenten, welche vollständig voneinander entkoppelt sein können und somit ein hohes Maß an Flexibilität bereitstellen [Cha06, RT06, Rom08]:

- Komponenten zur Erfassung von internen und externen Ereignissen (*Event Sensors*),
- Komponenten zur Auswertung und Aggregation von Ereignissen (*Event Processors*),
- Komponenten zur Reaktion auf Ereignisse (*Event Responders*).

Eine der Hauptanforderungen an eine ereignisgesteuerte Architektur ist dabei der Umgang mit einer großen, ggf. ungeordneten und unvollständigen Menge von heterogenen Ereignissen, welche sich sowohl positiv als auch negativ

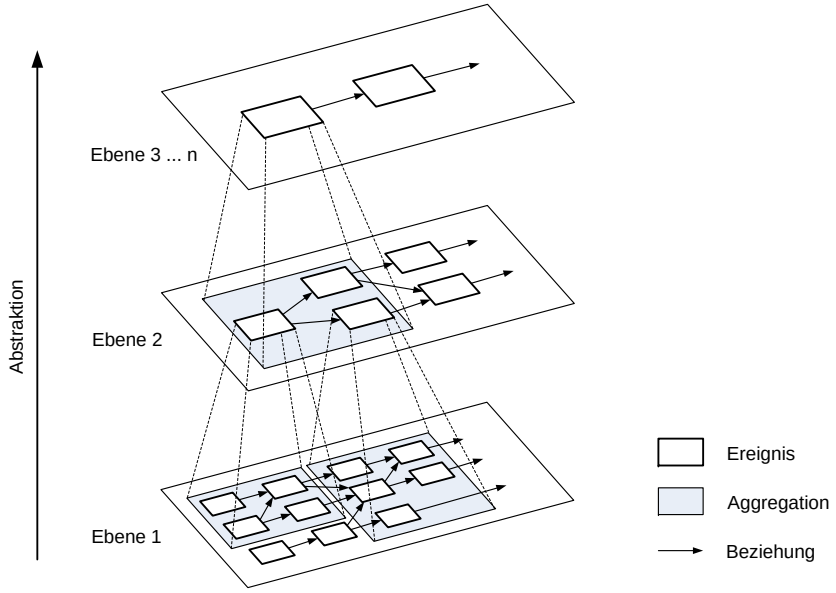


Abbildung 3.17: Ereignis-Hierarchie (nach (Luc02, Job10))

auf die (prozessorientierte) Anwendung auswirken können. LUCKHAM [Luc02] bezeichnet diese Menge von Ereignissen daher auch als sogenannte *Ereigniswolke* bzw. *Event Cloud*. Die zumeist große Menge von Ereignissen resultiert dabei zum einen aus der fachlichen Komplexität der Anwendung, der technischen Komplexität der unterliegenden IT-Infrastruktur, der Kommunikation mit der physikalischen Umwelt sowie aus der Granularität der Ereignisse [BD10]. Übergeordnetes Ziel einer ereignisgesteuerten Architektur ist es daher, die Ereignisse in einer systematischen Art und Weise zu identifizieren und sie den individuellen Konsumenten zeitnah und in einem geeigneten Abstraktionsniveau bereitzustellen. Hierfür werden vier Aspekte der Ereignisverarbeitung adressiert [BD10]:

- **Ereignismuster:** Das Erkennen von komplexen Mustern zwischen Ereignissen (sog. *Event Pattern*) umfasst die Identifikation ihrer Zusammenhänge, Abhängigkeiten und Korrelationen in Hinblick auf temporale oder räumliche Bedingungen.

- **Abstraktion:** Eine Betrachtung von Ereignissen auf unterschiedlichen Abstraktionsebenen ermöglicht die Verdichtung von Informationen auf einem problemadäquaten Abstraktionsniveau. Dabei lassen sich Ereignisse von einfachen (zumeist technischen) *Low Level Events* auf der Sensor-, Netzwerk- oder Middleware-Ebene bis hin zu höherwertigen, komplexen *High Level Events* mit unmittelbarer fachlicher Bedeutung auf der Ebene der prozessorientierten Anwendung klassifizieren. Die resultierenden Abstraktionsebenen werden als *Ereignis-Hierarchie* [Luc02] bezeichnet (vgl. Abbildung 3.17).

- **Beziehungen:** Die Ereignisse auf den verschiedenen Hierarchie-Ebenen besitzen ursächliche bzw. kausale Zusammenhänge, die von dem Auftreten einfacher Ereignisse, über das Erkennen eines Ereignismusters zu der Erzeugung eines (neuen) komplexen Ereignisses führen (*Aggregation* [Luc02]). Das Nachverfolgen von kausalen Beziehungen zwischen Ereignissen ist daher relevant, um eine Erklärung für das Entstehen von Ereignissen bzw. für deren Ausbleiben zu erhalten.

- **Aktionen:** Wenn bestimmte Muster von Ereignissen erkannt werden, können bei Bedarf geeignete Aktionen abgeleitet und initiiert werden. Diese Aktionen können dabei sowohl fachliche Konsequenzen in Form von auszuführender Prozesslogik als auch die Erzeugung von Ereignissen einer höheren Abstraktionsebene umfassen.

Die ereignisgesteuerte Architektur ergänzt die dienstorientierte Architektur, indem Dienste durch Ereignisse ausgelöst bzw. aufgerufen werden können. Zudem können Dienste untereinander sehr lose gekoppelt werden, wenn sie ausschließlich über Ereignisse miteinander kommunizieren [PvdH07]. Ein wesentlicher Nachteil besteht jedoch in der Zuverlässigkeit der Kommunikation, da ein Teilnehmer nicht davon ausgehen kann, dass die von ihm publizierten Ereignisse überhaupt abonniert und empfangen werden [Rom08, DEF+08, BD10].

Die individuelle Verarbeitung von Ereignissen erfolgt im Allgemeinen durch Regeln [Etz05] (vgl. Abschnitt 3.6). Hierfür können verschiedene Strategien zum Einsatz kommen. Zunächst können einfache Ereignisse genutzt werden, um relevante Informationen zu Geschehnissen zeitnah und zielgerichtet weiterzuleiten und einfache nachgelagerte Aktionen auszuführen (*Simple Event Processing*). Bei einer größeren Mengen von Ereignissen aus unterschiedlichen Quellen, deren Relevanz noch nicht festgestellt wurde, muss der Strom von eingehenden Ereignissen in geeigneter Weise gefiltert und verwaltet werden (*Event Stream Processing*). Schließlich kann die Erkennung von komplexen Ereignissen aus mehreren relevanten Ereignissen einer tieferen Abstraktionsebene auf Basis einer Übereinstimmung mit vorgegebenen Ereignismustern unterstützt werden (*Complex Event Processing*) [Cha06, RT06]. Analog zur Einteilung nach fachlicher und technischer Perspektive (vgl. Abschnitt 3.3) zeigt Abbildung 3.18 eine Übersicht des wesentlichen Nutzenpotentials von ereignisgesteuerten Architekturen. Im Folgenden wird die Bedeutung der Ereignisverarbeitung für die Flexibilisierung prozessorientierter Anwendungen genauer untersucht.

3.7.4 Nutzung einfacher Ereignisse

Auf Basis des vorgestellten Interaktionsmodells und den in Abschnitt 2.4.4 vorgestellten Möglichkeiten zur Modellierung von ereignisbedingten Reaktionsmöglichkeiten innerhalb des Prozesskontrollflusses lassen sich einfache Ereignisse sowohl in die Prozessbeschreibung integrieren als auch zur Entschei-

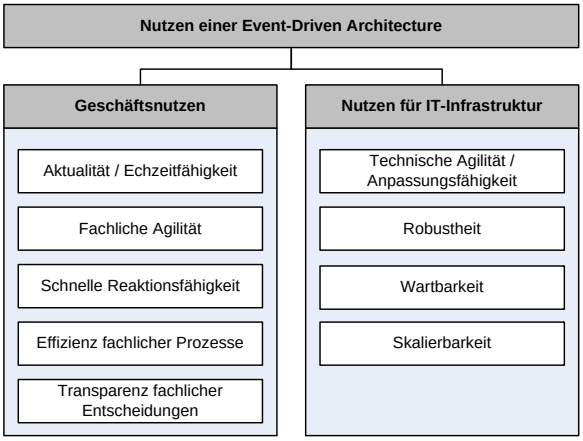


Abbildung 3.18: Nutzenpotential einer ereignisgesteuerten Architektur (BD10)

dung über die (nicht-funktionale) Ausführung von Prozessen nutzen. Im Kontext von prozessorientierten Anwendungen repräsentiert ein *einfaches Ereignis* eine für die Prozessausführung relevante Änderung einer bestimmten Eigenschaft. Dabei können Ereignisse sowohl im Prozess selbst auftreten, die Ausführungsumgebung (d. h. das Prozessmanagementsystem) bzw. die prozessausführende Organisation betreffen oder eine Änderung des extrinsischen Kontextes anzeigen (vgl. Abschnitt 3.5.2).

Bei Ereignissen im Kontext prozessorientierter Anwendungen kann zwischen *normalen* und *abnormalen Ereignistypen* unterschieden werden. Normale Ereignisse treten planmäßig im regulären Betrieb eines Systems auf und können in der Regel problemlos verarbeitet werden. Ein Beispiel ist das Empfangen einer erwarteten Antwortnachricht während der Kooperation mit einem entfernten Dienst. *Antizipierte abnormale Ereignisse* gehören nicht zum planmäßigen Betrieb eines Systems, jedoch wird ihr mögliches Eintreten erwartet und es existieren Möglichkeiten zur Erkennung einer bestimmten Situation und zu deren Behandlung (zum Beispiel eine Lieferverzögerung innerhalb eines Bestellprozesses, für welche entsprechende Fehlerbehandlungsmaßnahmen spezifiziert wurden). *Nicht-antizipierte abnormale Ereignisse* können hingegen in der Regel nicht ohne weiteres wahrgenommen und verarbeitet werden. Ein Beispiel ist der böswillige Angriff auf die genutzte IT-Infrastruktur [Cha06, Rom08].

Eine Entkopplung des Prozessmodells von der Ereignisspezifikation erlaubt eine relativ flexible Zuweisung von den für das Ereignis ursächlichen Zustandsänderungen zu Reaktionen innerhalb der Prozessbeschreibung [vAEE+10]. Für eine funktionale Reaktion auf Ereignisse müssen diese jedoch trotzdem in geeigneter Weise in den Prozess selbst integriert werden. Gegebenenfalls ist hierfür eine Erweiterung der genutzten Prozessbeschreibungssprache um die entsprechenden Konstrukte notwendig (vgl. auch 2.4.4) [vAEE+10]. Die einfachste fachliche Reaktion auf ein Ereignis im Kontext prozessorientierter Anwendungen besteht durch die Instantiierung eines Prozesses. Zum

Beispiel kann durch das Ereignis „Unfallmeldung“ der Prozess „Schadensbearbeitung“ angestoßen werden. Instantiierende Ereignisse müssen daher bereits beim Deployment des Prozesses beachtet werden, indem die spezifizierten Ereignisse bei der angegebenen Ereignis-Senke bzw. bei einem Vermittler abonniert werden [BDG07]. Zur Beeinflussung des Kontrollflusses stellen z. B. auch die Definition von Timeouts und der Austausch von Nachrichten *innerhalb* des Prozesses Ereignisse dar, die durch die Prozess-Engine verwaltet und daher spätestens bei dessen Instantiierung abonniert werden müssen [BDG07, vAEE+10]. Während der Prozessausführung kann der Prozess dabei entweder auf ein bestimmtes Ereignis warten und bei Eintritt des Ereignisses fortgeführt werden (z. B. beim Eingang einer erwarteten Nachricht) oder durch ein eingehendes Ereignis unterbrochen werden und eine Ereignisbehandlungsroutine ausführen. Letzteres hat den Vorteil, dass nicht mehr nur eine sequentielle Bearbeitung des Prozesskontrollflusses erfolgt, sondern verschiedene oder sogar beliebige Einsprungspunkte in Ausführungsroutinen für den Prozess ermöglicht werden. Die Behandlung von nicht-antizipierten abnormalen Ereignissen auf der fachlichen Ebene ist hierdurch jedoch im Allgemeinen nicht möglich [Rom08]. Hierfür ist eine weitergehende Analyse und Erkennung von (komplexen) Ereignismustern notwendig [BDG07, DGB07], wie sie im nachfolgenden Abschnitt vorgestellt wird.

Nicht-funktionale Anpassungen der Prozessausführung können durch die Nutzung von Ereignissen für ein echtzeitnahes Monitoring (vgl. Abschnitt 2.7) unterstützt werden [AESW09]. Ein Beispiel ist der Austausch von Ressourcen bei abfallender Leistung oder bei Nicht-Verfügbarkeit. Diese Art von Anpassungen erfordern eine Integration von (vornehmlich technischen) Ereignissen der unterliegenden Infrastruktur in die Ausführungsumgebung für Prozesse. Im Kontext einer dienstorientierten Architektur betrifft dies in erster Linie die Beobachtung und Anpassung der involvierten Dienstinstanzen [PvdH07].

Voraussetzungen für den Austausch und die effiziente (automatische) Verarbeitung von Ereignissen ist die Kenntnis der verwendeten Syntax und ein einheitliches Verständnis über die Bedeutung eines Ereignisses [DEF+08]. Aus diesem Grund wird eine geeignete Standardisierung für den Austausch und die Verarbeitung von Ereignissen angestrebt. Ein Beispiel ist die Spezifikation *Common Base Event (CBE)* [IBM04] von IBM, welche eine einheitliche Struktur von Ereignissen und ihren Metadaten definiert.

3.7.5 Verarbeitung komplexer Ereignisse

Insbesondere die Verarbeitung und Nutzung komplexer Ereignisse hat im Rahmen des Prozessmanagements in den letzten Jahren stark an Bedeutung gewonnen, um eine zeitnahe Anpassung der prozessorientierten Anwendung auf sich ändernde Umgebungsparameter zu ermöglichen. So ist das als *ereignisgesteuertes Prozessmanagement* (*Event driven (Business) Process Management*) benannte Forschungsgebiet als Verknüpfung dienst- bzw. prozessorientierter

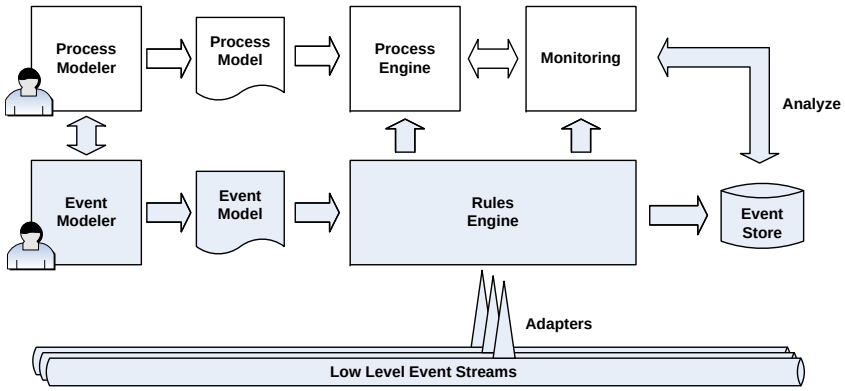


Abbildung 3.19: Modell des Zusammenwirkens zwischen Prozessmanagement und Ereignisverarbeitung (tlw. nach (vAEG+08))

Architekturen mit der Disziplin des *Complex Event Processing (CEP)* hervorgegangen [AESW09, vAEE+10].

Das Complex Event Processing beschäftigt sich allgemein mit der Erkennung, Analyse, Gruppierung und Verarbeitung voneinander abhängiger Ereignisse [Luc02, EB09]. Durch die Korrelation von einzelnen atomaren Ereignissen zu komplexen Ereignissen können diese für verschiedene Nutzergruppen und Bedürfnisse spezifiziert und miteinander in Verbindung gesetzt werden. Da Änderungen an den Regeln zur Zusammensetzung dieser komplexen Ereignisse und ihrer Reaktionen prinzipiell sogar zur Laufzeit der Anwendung möglich sind, resultiert hieraus eine besonders hohe Flexibilität [Luc02]. Auf Basis der Strukturierung der Ereignisse in technische und fachliche Abstraktionsebenen (vgl. Abbildungen 3.17 und 3.18) ermöglicht das Complex Event Processing somit eine dauerhaft flexible Ableitung von individuell relevanten Ereignissen für einen bestimmten Anwendungsfall und somit eine angepasste, echtzeitnahe Überwachung des Prozesses selbst und der für seine Ausführung wesentlichen Aspekte [AESW09].

Eine mögliche Umsetzung der Nutzung komplexer Ereignisse für prozessorientierte Anwendungen wird durch die lose Kopplung von Systemen zum Prozessmanagement und zur Ereignisverarbeitung vorgeschlagen. Hierbei agiert ein zum Prozessmanagementsystem unabhängig parallel laufendes Complex-Event-Processing-System, welches alle relevanten Ereignisse analysiert und verarbeitet, die im Kontext der laufenden prozessorientierten Anwendungen entstehen oder von diesen selbst ausgelöst werden [AESW09]. Das Prozessmanagementsystem (d. h. insbesondere die Prozess-Engine und die Komponenten zum Monitoring) kommunizieren hierbei durch Ereignisse, die vom Prozess selbst und den mit den einzelnen Aktivitäten des Prozesses verknüpften Softwarekomponenten (z. B. Dienste einer dienstorientierten Architektur) erzeugt werden. Abbildung 3.19 zeigt eine entsprechende Unterstützung des Prozessmanagements durch das Erfassen und die Verarbeitung relevanter Ereignisse aus mehreren Ereignisströmen (vgl. [vAEG+08]). Parallel zur Modellierung des Prozesses erfolgt hier die Modellierung eines *Ereignis-Modells*, welches die

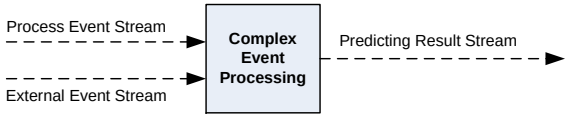


Abbildung 3.20: Erkennung von Ereignismustern zur Berücksichtigung externer und prozessinterner Einflüsse (nach (AESW09))

erforderlichen Ereignistypen definiert und festlegt, welche Warnungen an welche Gruppen in der betreffenden Organisation zu senden sind sowie welche Aktionen automatisch zu starten sind, falls ein bestimmtes Ereignismuster erkannt wird. Ziel ist es hierbei, die Ereignisverarbeitung und -auswertung von der fachlichen Logik der prozessorientierten Anwendungen weitestgehend zu entkoppeln. Dies ermöglicht es, die zu einem Ereignis auf fachlicher Ebene der Ereignishierarchie führenden Regeln zur Aggregation von elementaren Ereignissen zu modifizieren, ohne dass Änderungen am Prozessmodell notwendig werden. Die Zusammensetzung von Ereignissen kann somit sich ändernden Umgebungsbedingungen flexibel angepasst und sogar zur Laufzeit einzelner Prozessinstanzen ausgetauscht werden [Etz05].

Die Integration von Komponenten des Complex Event Processing und des Prozessmanagementsystems geschieht in der Regel über eine gemeinsame Infrastruktur (z. B. über einen *Enterprise Integration Backbone (EIB)* [EB09, Luc02]). Hierbei können einerseits Prozesse durch Ereignisse gesteuert werden oder andererseits selber Ereignisse auslösen.

Durch die Korrelation von einzelnen atomaren Ereignissen zu komplexen Ereignissen können Probleme frühzeitig erkannt und Warnungen oder proaktive Maßnahmen veranlasst werden [AESW09]. Die gezielte Verarbeitung von Ereignisströmen und komplexen Ereignissen kann somit auch zur *Vorhersage* und damit zur *proaktiven Anpassung* eines Prozesses genutzt werden. Die Autoren VON AMMON et al. [AESW09] stellen im Kontext von Prozessen im Logistikbereich ein allgemeines Ereignismuster vor, welches den Einfluss externer (antizipierter) Ereignisse auf die erfolgreiche Ausführung eines Prozesses ausdrückt. Dazu werden besondere Ereignisse der Umwelt (z. B. ein Verkehrsstau auf einer bestimmten Route) und prozessinterne Ereignisse (z. B. das Ausliefern von Waren auf dieser Route) gemeinsam verarbeitet, um die wahrscheinlichen Konsequenzen des Zusammenspiels dieser Ereignisse zu berechnen (z. B. eine resultierende Lieferverzögerung) [AESW09]. Abbildung 3.20 verdeutlicht das Zusammenspiel der beiden Ereignisströme zur Gewinnung prozessrelevanter Vorhersagedaten.

Ereignismuster werden mit Hilfe spezieller regelbasierter Beschreibungssprachen (*Event Processing Languages (EPL)*) ausgedrückt [DEF+08]. Eine besondere Herausforderung besteht dabei in dem Umgang mit einer großen Menge an Ereignissen, welche in der Regel schnell an Wert verlieren, wenn sie nicht zeitnah und zielgerichtet ausgewertet werden. Für die Aggregation von Ereignisdaten hat es sich daher als besonders vorteilhaft herausgestellt, nicht die Ereignisse selbst komplett (z. B. in einer Datenbank) zu speichern, son-

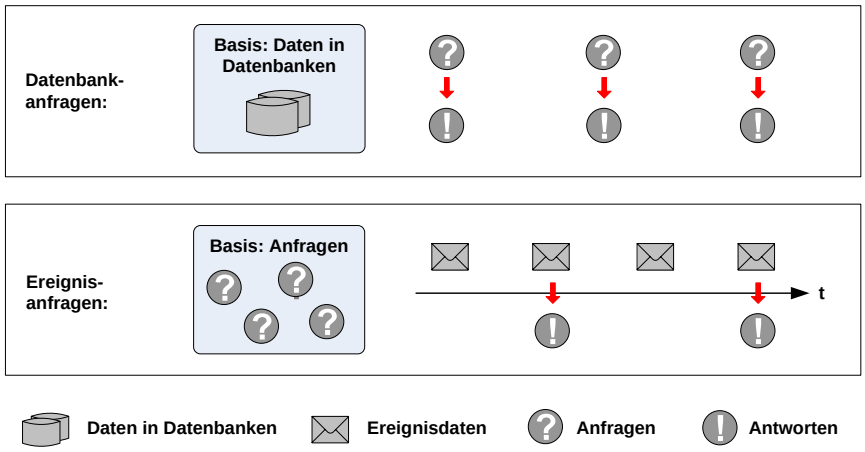


Abbildung 3.21: Anfrage von Ereignisströmen in Vergleich zu Datenbankabfragen (EB09)

dern die spezifizierten Regeln zur Auswertung der Ereignisse direkt auf den Ereignisstrom und/oder eine begrenzte Menge von gespeicherten Ereignissen anzuwenden. Hierzu dienen sogenannte *Datenstrom-Anfragesprachen* (z. B. *Continuous Query Language (CQL)* [ABW06]), welche auf der Datenbankabfragesprache SQL basieren. Bei einer Anfrage werden Datenströme, die Ereignisse als Tupel enthalten, in Relationen umgewandelt. Auf diesen Relationen wird dann eine reguläre SQL-Anfrage ausgewertet und die Ergebnis-Relation zurück in einen Datenstrom gewandelt. Konzeptionell wird dieser Vorgang an jedem Zeitpunkt einer vorgegeben diskreten Zeitachse durchgeführt [EB09]. Abbildung 3.21 verdeutlicht diese Vorgehensweise im Vergleich zu einer normalen Datenbankabfrage. Messungen hinsichtlich der Skalierbarkeit und des Durchsatzes einer solchen Ereignisverarbeitung belegen, dass auf diese Weise auch komplexe Anfragen in einem angemessenen Zeitraum durchgeführt werden können (vgl. [WDR06, Esp07, MBM09]).

3.8 Flexibilität durch Autonomic Computing

Basierend auf den genannten grundlegenden Eigenschaften der losen Kopplung von dienstorientierten Softwaresystemen, der Wahrnehmung von Kontextinformationen und Ereignissen sowie deren Verarbeitung durch regelbasierte Systeme können höherwertige Paradigmen und Ansätze genutzt werden, um prozessorientierte Anwendungen nicht nur prinzipiell anpassungsfähig zu gestalten, sondern auch selbständig Anpassungen durch das System durchführen zu lassen (vgl. Abschnitt 3.1). Die automatische Anpassung von Systemen oder Teilsystemen führt dabei nicht nur zur Entlastung menschlicher Akteure und einem geringeren Wartungsaufwand, sondern insbesondere auch zu einer besseren Beherrschbarkeit der Komplexität und in der Regel zu einer schnelleren Reaktionszeit [KC03, Mur04].

Konzepte zur *Selbstverwaltung* (bzw. zum *Self-Management* [KC03]) von einzelnen Systemen bzw. Systemkomponenten sollen insbesondere den Admini-

strator bei Routineaufgaben wie der (Re-)Konfiguration von Parametern oder bei der Fehlerbehandlung unterstützen. Das Paradigma des *Autonomic Computing* stellt hierfür ein allgemeines Konzept dar, welches auch auf prozessorientierte Anwendungen und Prozessmanagementsysteme angewendet werden kann. In diesem Abschnitt wird die Umsetzung von autonomem Verhalten nach dieser Sichtweise vorgestellt und mit dem Kontext prozessorientierter Anwendungen in Verbindung gebracht.

3.8.1 Autonomic Computing

Das Paradigma *autonomer Computersysteme* bzw. des *Autonomic Computing* basiert auf dem Vorbild des menschlichen autonomen Nervensystems, welches in der Lage ist, Körperfunktionen des Menschen für diesen unbewusst anhand der Wahrnehmung äußerer Umstände zu steuern [KC03]. Bei einem Computersystem wird diese Selbstverwaltung durch Kontrollschleifen realisiert, welche über Sensorschnittstellen Fakten über zu verwaltenden Ressourcen sammeln und entsprechend darauf reagieren [KC03, Mur04]. Im Kontext des Autonomic Computing werden vier Arten von autonomem Verhalten unterschieden [KC03, Par07, HM08]:

- **Self-Configuration:** Ein System wird als *selbst konfigurierend* bezeichnet, wenn es möglich ist, neue Systemkomponenten in eine bestehende Umgebung einzufügen, ohne dass eine manuelle Einrichtung der Komponenten notwendig ist. Ein bekanntes Beispiel hierfür sind *Plug-and-Play*-Komponenten.

- **Self-Optimization:** Das System ist *selbst optimierend*, wenn die Allokation und Nutzung von Ressourcen in Hinblick auf die Bedürfnisse des Benutzers automatisch effizient durchgeführt werden. Beispiele sind das Auslagern von Speicher oder die Anwendung dynamischer Dateisysteme.

- **Self-Healing:** Sich *selbst heilende* Systeme verfügen über die Möglichkeit, Fehler und Störungen innerhalb des Systems zu entdecken, die Ursache hierfür zu diagnostizieren und den Fehler mit geeigneten Maßnahmen zu beheben. Ein Beispiel ist der Austausch von fehlerhaften (Software-)Komponenten.

- **Self-Protection:** Kann ein System böswillige Angriffe bzw. Schäden von solchen Angriffen erkennen, wird ihm die Eigenschaft zugesprochen, sich *selbst schützen* zu können. Einfaches Beispiel ist die Erkennung von autorisierten Benutzern und die Bereitstellung der vom Benutzer zugreifbaren Daten.

Zur Umsetzung dieser Eigenschaften verfügen autonome Computersysteme wie allgemeine regelbasierte Systeme (vgl. Abschnitt 3.6.1) über eine *Wissensbasis*, welche Entscheidungen auf der Basis von benutzerdefinierten Vorgaben ermöglicht. Hierzu wird eine Menge von Regeln (*Policies*) verwaltet, welche

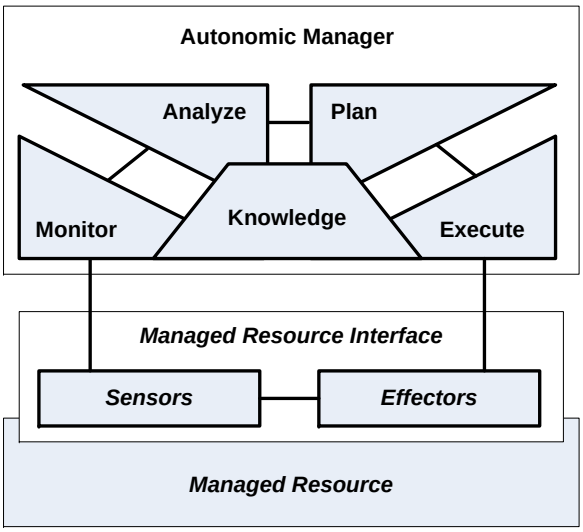


Abbildung 3.22: Kontrollschleife des Autonomic Computing (nach (KC03, DHP+05, MNS+05, HM08))

a-priori bekannte Abhängigkeiten zwischen Systemkomponenten ausdrückt und potentielle Lösungen in Form von Reaktionen (*Anpassungsstrategien*) mit Prioritäten versieht. Die Architektur autonomer Systeme stützt sich dabei auf einen grundlegenden Kontrollzyklus, welcher im folgenden Abschnitt kurz vorgestellt wird.

3.8.2 Kontrollschleifenbasiertes Architekturmodell

Die dargestellte Vision des Autonomic Computing stützt sich im Wesentlichen auf ein Architekturmodell, welches die automatische Verwaltung einer (beliebigen) gegebenen Systemressource auf der Basis benutzerdefinierter Vorgaben erlaubt. Die betrachtete Systemressource wird dabei als sogenannte *verwaltete Ressource* (*Managed Resource*) angesehen, welche durch eine zusätzliche, eine Kontrollschleife implementierende Systemkomponente verwaltet wird. Diese Komponente stellt den sogenannten *Autonomic Manager* dar und ist nach dem Prinzip der funktionalen Dekomposition in die vier Funktionen *Monitoring*, *Analysis*, *Planning* und *Execution* aufgeteilt, welche nacheinander durchlaufen werden. Diese Kontrollschleife des Autonomic Computing wird auf Basis der Initialen der vier Funktionen typischerweise als sogenannte *MAPE-Kontrollschleife* (*MAPE-Loop*) bezeichnet [KC03, Mur04, MNS+05, LSPF07].

Abbildung 3.22 zeigt eine zusammenfassende Darstellung der wesentlichen Zusammenhänge. Dabei basieren die vier Funktionen der Kontrollschleife auf einer gemeinsamen Wissensgrundlage (*Knowledge*), welche die benutzerdefinierten Regeln als Richtlinien für das autonome Verhalten beinhaltet [MNS+05]. Die Funktionen haben jedoch streng voneinander getrennte Aufgabenbereiche:

- Die für das *Monitoring* zuständige Komponente sammelt zunächst Informationen der zu verwaltenden Ressource und ihrer relevanten Um-

welt. Dies kann beliebige durch Sensoren erhobene (Kontext-)Daten, einfache oder komplexe Ereignisse sowie benutzerdefinierte, konfigurierbare Eigenschaften umfassen [Par07]. Die Monitoring-Funktion ist zudem für die Aggregation, Filterung und Aufbereitung dieser Daten verantwortlich [MNS+05]. Wird ein bestimmtes *Symptom* erkannt (z. B. eine erhöhte Verarbeitungszeit), wird dieses an die Analysefunktion weitergereicht [Par07].

- Im *Analyseteil* der Kontrollschleife werden Mechanismen bereitgestellt, um Symptome zu prüfen, deren Relevanz anhand der Gesamtsituation zu beurteilen und zu entscheiden, ob eine Anpassung des Systems als Reaktion auf das Symptom notwendig ist [MNS+05, Par07]. Dabei kann die Analysefunktion durchaus auch Vorhersagen über mögliche zukünftige Kontexte einbinden, um eine proaktive Anpassung zu ermöglichen. Wird zum Beispiel eine erhöhte Verarbeitungszeit und ein ausgefallener Server erkannt, so kann eine Zuweisung der kritischen Aufgabe an einen neuen Server notwendig werden, um die zukünftige Verletzung von antwortzeitbezogenen Dienstgütevereinbarungen zu verhindern [Par07]. Die Analysefunktion kann in so einem Fall entweder eine entsprechende *Änderungsanforderung* (*Change Request*) an die nachfolgende Planungsfunktion stellen oder bei der Monitoring-Komponente weitere Detailinformationen abrufen [MNS+05]. Die Kontrollschleife ist daher nicht zwingend als gerichteter Kreislauf anzusehen.

- Die für die *Planung* zuständige Komponente erzeugt oder selektiert die für die Durchführung der gewünschten Änderung erforderlichen Maßnahmen in Form von Plänen auf der Basis von benutzerdefinierten Regeln [MNS+05]. Ein Plan kann dabei aus einem einfachen Änderungsbefehl an die lokal verwaltete Ressource bestehen oder einen komplexen Änderungsprozess enthalten, welcher sich durchaus auch über weitere verwaltete Ressourcen fortpflanzen kann [DHP+05, Par07]. Der resultierende Plan wird schließlich an die Ausführungsfunktion übergeben.

- Die *Ausführungsfunktion* ist für die Durchführung der identifizierten Anpassungen verantwortlich. Die mit der Konsequenz einer Regel verbundene Aktion wird hier ausgeführt [DHP+05, Par07].

Die Analysefunktion und die Ausführungsfunktion haben dabei direkten Zugriff auf die verwaltete Ressource, weshalb diese eine hierfür geeignete Schnittstelle (*Managed Resource Interface*) implementieren muss (vgl. Abbildung 3.22). Da diese Schnittstelle auch für die Verwaltung von Ressourcen zum Prozessmanagement relevant ist, soll die verwaltete Ressource mit einer solchen Schnittstelle im Folgenden kurz genauer betrachtet werden.

3.8.3 Verwaltbare Ressourcen

Eine durch einen Autonomic Manager verwaltete Ressource wird im Kontext des Autonomic Computing als *Managed Resource* bezeichnet [KC03, Mur04, DHP+05, MNS+05, Par07]. Um jedoch durch das oben erläuterte kontrollschleifenbasierte Architekturmodell beobachtet und angepasst werden zu können, sind zunächst die prinzipielle Erhebung und Mitteilung von relevanten Informationen über die Ressource und die prinzipielle Anpassbarkeit der Ressource notwendig. Dabei sollte die Änderung eines Parameters der Ressource durch die Ausführungsfunktion des Autonomic Managers im Idealfall wieder als Information von dessen Monitoring-Funktion wahrgenommen werden können [Par07]. Diese wesentlichen Voraussetzungen der Ressource werden als *Verwaltbarkeit* bzw. als *Manageability* bezeichnet. Eine Ressource, welche diese grundlegenden Voraussetzungen erfüllt, heißt daher auch *verwaltbare Ressource* bzw. *Manageable Resource* [BV06a, PTDL07, Par07].

Eine verwaltbare Ressource kann verschiedene Granularitäten annehmen und einerseits auf der Ebene der übergeordneten Entität (z. B. Hardware-Server) oder als feingranulares Element einer Entität (z. B. Speicher, Prozessor oder Peripheriegerät) angesehen werden. Unabhängig von der Granularität erfolgt die Kontrolle der Ressource über eine möglichst für alle verwalteten Ressourcen einheitliche Schnittstelle (*Manageability Interface*), welche im Detail über eine Sensorschnittstelle (*Sensor*) und eine Effektorschnittstelle (*Effector*) verfügt [MNS+05, Par07, HM08] (vgl. Abbildung 3.22). Die Sensorschnittstelle stellt Informationen einerseits auf Abruf bereit und erlaubt andererseits das Abonnement von Ereignissen. Beispiele sind die Abfrage von Zustandsinformationen, die Bereitstellung von Operationen zur Identifikation von verwalteten Ressourcen und Mechanismen zur Durchführung von Messungen. Auf Seiten der Effektorschnittstelle werden Operationen angeboten, welche Änderungen am Zustand der verwalteten Ressource erlauben, zum Beispiel an Konfigurationsparametern [MNS+05, Par07]. Die Kombination von Sensor- und Effektorschnittstelle bildet den erforderlichen Berührungspunkt (*Touchpoint Building Block*) zwischen Autonomic Manager und verwaltbarer Ressource.

Eine wesentliche Herausforderung der Verwaltung mehrerer (heterogener) Ressourcen stellt eine einheitliche Umsetzung der Sensor- und Effektorschnittstellen und der nach Möglichkeit standardisierte Zugriff auf die bereitgestellte Management-Funktionalität dar. Im Rahmen von dienstorientierten Architekturen stellen hierbei Web Services (vgl. Abschnitt 3.4.3) einen geeigneten Ansatz dar, um unabhängig von der tatsächlichen Implementierung einen standardisierten Zugriff auf die Funktionen von Sensoren und Effektoren zu ermöglichen [MNS+05, Par07, PTDL07]. Auf der Basis des *Web Service Resource Frameworks* (*WSRF*) [CFF+06] hat OASIS daher den Standard *Web Services Distributed Management* (*WSDM*) veröffentlicht, welcher ein Modell für den Zugriff auf Verwaltungsfunktionen von verwaltbaren Ressourcen durch Web Services (*Management Using Web Services*, kurz *MUWS*) [BV06a, BV06b] beschreibt und gleichzeitig die Verwaltung eines Web Services als eigene verwalt-

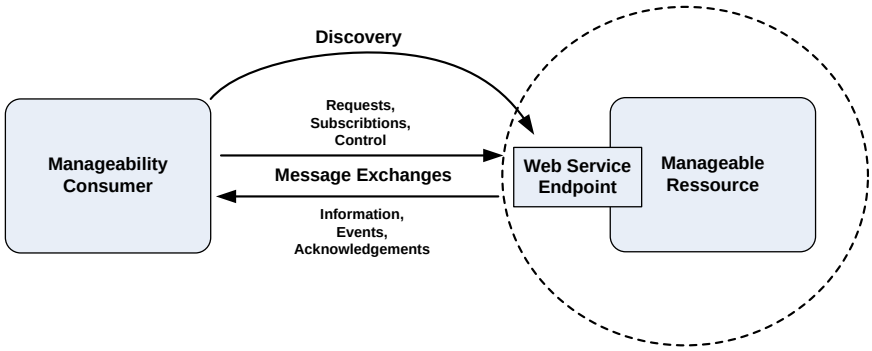


Abbildung 3.23: Grundlegendes Modell von WSDM (nach (BV06a))

bare Ressource (*Management Of Web Services*, kurz *MOWS*) [WS06] aufgreift (vgl. [BMW06]). Abbildung 3.23 zeigt das WSDM zugrunde liegende Konzept des Zugriffs einer verwaltbaren Ressource über Web Services. Hierbei kann die von einer Ressource bereitgestellte Management-Schnittstelle über bestehende Discovery-Mechanismen dienstorientierter Architekturen dynamisch aufgefunden (vgl. Abschnitt 3.4.3) und die angebotene Management-Funktionalität beliebigen (sowohl lokalen als auch entfernten) Konsumenten zur Verfügung gestellt werden.

In einem erweiterten Szenario des Autonomic Computing steht nun die hierarchische Verwaltung von mehreren Ressourcen durch mehrere Autonomic Manager im Vordergrund. In diesem Fall verfügen die Manager der einzelnen Ressourcen eines komplexeren Systems selbst über Sensoren und Effektoren, welche es erlauben, übergeordneten Verwaltungskomponenten ihren Zustand mitzuteilen und Änderungsanforderungen von diesen entgegen zu nehmen. Dabei können globale und lokale Richtlinien die Verbreitung von Informationen und die Ausführung von Zustandsänderungen innerhalb der lokalen Komponenten, aber auch auf der Ebene des Gesamtsystems kontrollieren und benutzerdefiniert einschränken [KC03, MNS+05, Par07]. Es entsteht nach dieser Vision ein autonomes Gesamtsystem, welches mit Ausnahme der vorgegebenen Regeln als Teil der Wissensbasis ohne weiteren Eingriff menschlicher Benutzer geeignete Anpassungen an seinen Subsystemen vornehmen kann.

Ob eine Ressource eine im erweiterten Kontext des Autonomic Computing verwaltbare Ressource darstellt, hängt neben der Bereitstellung einer geeigneten Schnittstelle oft von Sicherheitseinstellungen und der erlaubten Freigabe der Ressource in einem übergeordneten System dar. In vielen Fällen (z. B. bei einer unternehmensübergreifenden Kooperation) ist es eventuell gar nicht gewünscht, dass andere Systemkomponenten Einfluss auf die Parametrisierung oder die Steuerung einer (privaten) Ressource nehmen. Die Ressource wird dann oft als *autonom* im Sinne von „unabhängig“ und nicht im Sinne von „selbständig ohne Interaktion des menschlichen Benutzers“ bezeichnet. Falls notwendig, wird daher in dieser Arbeit bei der Verwendung des Autonomiebegriffs auf dessen Bedeutung im jeweiligen Kontext hingewiesen.

3.8.4 Autonomes Prozessmanagement

Die mit der Entwicklung und Überarbeitung prozessorientierter Anwendungen sowie der Konfiguration von Prozessmanagementsystemen verbundenen Aufgaben stellen für den Systemadministrator in der Regel einen großen Aufwand dar. Ursache hierfür sind zum einen die zugrunde liegenden Informations- und Kommunikationssysteme, welche oft aus sehr vielfältigen, heterogenen Komponenten bestehen und einzeln angepasst werden müssen. Zum anderen werden die Prozesse selbst durch den fortwährenden Änderungsbedarf in der Regel immer umfangreicher und komplexer. Eine manuelle Konfiguration des Gesamtsystems in Hinblick auf Optimierung oder Fehlerbehandlung ist daher insbesondere bei umfangreichen Ausführungsumgebungen als schwierig einzustufen. Neben der oft unzureichenden Effizienz sind manuelle Änderungen zudem eine häufige Ursache für Fehler oder unerwünschte Seiteneffekte, die sich aus der Veränderung einzelner Parameter ergeben und oft weitreichende Folgen (wie z. B. das Scheitern einer Prozessausführung) haben können [SY06, LSPF07]. Die Übertragung der Konzepte des Autonomic Computing auf den Bereich des Prozessmanagements verspricht daher neben der Unterstützung menschlicher Akteure und der Beschleunigung von notwendigen Anpassungen auch die Erhöhung der Robustheit von prozessorientierten Anwendungen. Analog zu der in Abschnitt 3.3 getroffenen Klassifizierung der Flexibilität nach technischen und fachlichen Gesichtspunkten können im Rahmen des *autonomen Prozessmanagements* zwei Schwerpunkte identifiziert werden:

- Die Prozess-Engine, das Prozessmanagementsystem oder die gesamte Ausführungsumgebung mit allen technischen Ressourcen stellt eine *verwaltbare Ressource* im Sinne des Autonomic Computing dar (*technische Perspektive*) [HPA05, HPA06].
- Die inhaltliche Anpassung der prozessorientierten Anwendung bzw. die Abbildung eines abstrakten, strategischen Prozesses auf einen konkreten ausführbaren Prozess wird durch Konzepte des Autonomic Computing unterstützt (*fachliche Perspektive*) [SY06].

Abbildung 3.24 zeigt zunächst einen Überblick über den Ansatz des technischen Managements, wobei typische Parameter des technischen Prozessmanagements identifiziert und den vorgestellten Funktionen der MAPE-Kontrollschleife (vgl. Abschnitt 3.8.2) zugeordnet werden. Die Monitoring-Phase umfasst hierbei im Wesentlichen das Erfassen von Laufzeitinformationen. Die Autoren LEE et al. [LSPF07] unterscheiden zwischen *prozessbezogenen Zustandsinformationen* (z. B. Fortschritt oder Beendigung einer Aktivität bzw. des Gesamtprozesses), dem *Zustand von Ressourcen* (z. B. Verfügbarkeit von Diensten oder Daten) und dem *Zustand der Ausführungsumgebung* (z. B. Verfügbarkeit und Auslastung der Komponenten einer Prozess-Engine oder der verwendeten Netzwerkverbindungen). Nach einer entsprechenden Verdichtung der Daten werden diese Informationen in der Analyse-

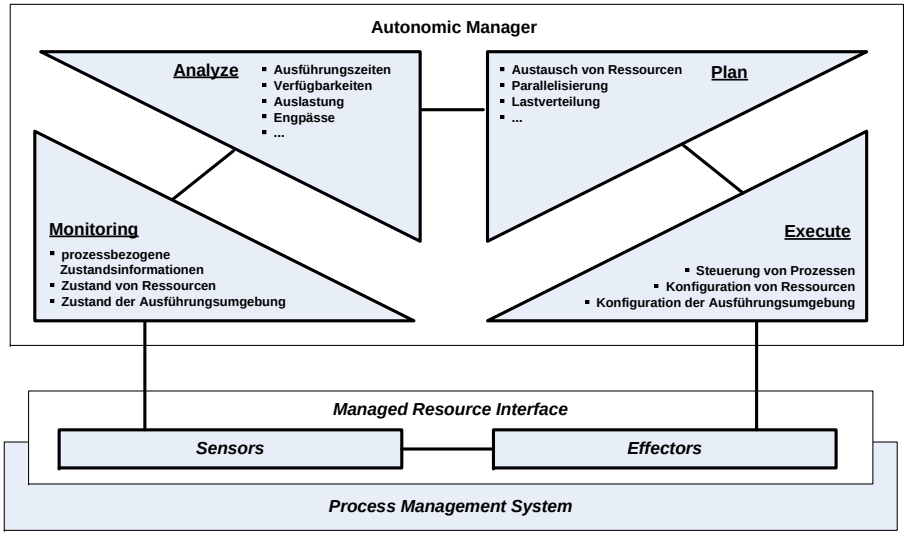


Abbildung 3.24: Autonomes Management technischer Aspekte der Prozessausführung (tlw. nach (HPA05, LSPF07))

Phase miteinander in Verbindung gebracht, so dass einerseits *Probleme* (z. B. eine ungünstige Lastverteilung oder das Überschreiten der geplanten Ausführungszeit eines Prozesses) oder andererseits *Chancen* (z. B. die Verfügbarkeit neuer, günstigerer Ressourcen oder das Erkennen von Systemen im Leerlauf) identifiziert werden können [LSPF07]. In der Planungsphase wird anhand einer Menge von Strategien entschieden, durch welche Maßnahmen das Prozessmanagement korrigiert bzw. optimiert werden kann. Zu den möglichen Strategien gehören z. B. die Erhöhung des Parallelisierungsgrades, das Umplanen bzw. der Austausch von Ressourcen oder das Verschieben von Ausführungskomponenten zur Nutzung von Netzwerkverbindungen mit höherer Bandbreite [HPA05, LSPF07]. In der letzten Phase des MAPE-Kontrollflusses werden die gewählten Pläne schließlich ausgeführt. Die hierfür notwendigen Effektoren werden in diesem Kontext einerseits durch die Steuerungsfunktionen des Prozessmanagementsystems (z. B. zum Anhalten, Abbrechen oder Neustarten von Prozessen) bereitgestellt [SLI08] oder sind durch die Konfigurationsmöglichkeiten der Ausführungsumgebung und der Ressourcen determiniert (z. B. durch das Zuweisen von zusätzlichem Speicher, der Verteilung von Systemkomponenten auf mehrere physikalische Einheiten oder die Anpassung von (Multi-)Threading-Anwendungen).

Ein Beispiel für eine konkrete Umsetzung des autonomen Prozessmanagements liefern die Autoren HEINIS, PAUTASSO und ALONSO [HPA05, HPA06]. Vor dem Hintergrund, dass die Auslastung (*Workload*) von Prozessmanagementsystemen weitestgehend nicht vorhersagbar ist, schlagen sie lastabhängige Strategien zur Selbstkonfiguration, Selbstheilung und Selbstoptimierung einer Prozess-Engine vor. Die spezielle Prozess-Engine besteht hierzu aus mehreren Komponenten, welche bei Bedarf auf verschiedene Hardware-Systeme verteilt werden können, um sowohl gleichzeitig durch den

Kontrollfluss mehrerer Prozesse navigieren zu können als auch die parallele Ausführung von Aktivitäten eines oder mehrerer Prozesse zu optimieren [HPA05, HPA06]. Die Prozess-Engine wird hierzu als *Verwaltbare Ressource* ausgestaltet, welche es erlaubt, die aktuellen Zustände bezüglich der Systemlast zu beobachten (*Sensoren*) und bei Anpassungsbedarf eine entsprechende Anzahl an Threads zur Aktivitäts- bzw. Prozessbearbeitung zu starten (*Effektoren*). Die Anpassungslogik mit den nutzerdefinierten Strategien wird hierbei komplett an den *Autonomic Manager* des Systems ausgelagert (vgl. Abbildung 3.24) und kann somit zusätzlich unabhängig vom Prozessmanagementsystem verändert oder erweitert werden.

Im Gegensatz zu einer solchen technischen bzw. organisatorischen Anpassung von Prozessmanagementsystemen ist eine autonome inhaltliche Anpassung von Prozessen ohne die Beteiligung menschlicher Akteure nur unter bestimmten Voraussetzungen und oft nur begrenzt möglich. Beispiele für eine automatische inhaltliche Anpassung von Prozessen sind die automatische Verfeinerung von abstrakten Aktivitäten des Prozesses zu konkreten Aktivitäten oder detaillierten Subprozessen (z. B. durch eine Konkretisierung von kontextsensitiven Bereichen, vgl. Abschnitt 3.5.3), das Hinzufügen oder Entfernen von konkreten Implementierungen [LSPF07] oder die inhaltliche Änderung des Kontrollflusses (z. B. durch Parallelisierung von zuvor sequentiellen Aktivitäten). In Anlehnung an die von IBM definierten *Autonomic Maturity Levels* (vgl. [MNS+05]) haben die Autoren STROHMEIER und YU [SY06] fünf Stufen der Autonomie von Prozessmanagementsystemen abgeleitet, wobei sie die Perspektive der fachlichen Anpassung von Prozessen fokussieren:

- **1. Basic Level:** Auf dieser ersten Stufe werden die Anpassungen durch den menschlichen Anwendungsentwickler modelliert, analysiert und bei Bedarf manuell angepasst. Unterstützung erfolgt nur durch wenige technische Hilfsmittel wie zum Beispiel durch Werkzeuge zur Überprüfung der Syntax.
- **2. Managed Level:** Auf der zweiten Stufe wird der manuelle Anpassungsprozess durch den Anwendungsentwickler durch die Bereitstellung von statistischen Daten zum Laufzeitverhalten der Prozesse unterstützt.
- **3. Predictive Level:** Die dritte Stufe erlaubt die Simulation von Prozessabläufen sowie die Verwaltung und Auswertung verschiedener Prozessvarianten (*Process Mining*), so dass der Anwendungsentwickler bei der Modellierung oder der Anpassung von Prozessen zielgerichtet aus mehreren Alternativen eine Auswahl treffen kann.
- **4. Adaptive Level:** Auf der vierten Stufe werden einige Anpassungen bereits durch das Prozessmanagementsystem übernommen. Als Beispiele nennen STROHMEIER und YU automatisierte Fehlerbehandlungsmaßnahmen und Aktivitäten zur Optimierung der Prozesse (vgl. [SY06]).
- **5. Autonomic Level:** Die fünfte Stufe stellt den höchsten Grad an Autonomie dar, wobei der Anwendungsentwickler nur noch über die Defi-

nition von Zielvorgaben mit dem Prozessmanagementsystem interagiert. Auf Basis dieser vorgegebenen Ziele und den eigenständig erhobenen Daten über sich selbst und seine Umwelt kann das Prozessmanagementsystem selbständig Entscheidungen bezüglich der Ausführung von einzelnen Prozessvarianten treffen.

Eine derart zielorientierte Anpassung, wie sie auf der höchsten Autonomiestufe gefordert wird, kann durch das klassische Prozessmanagement bislang kaum unterstützt werden. Daher existieren hierfür andere Softwareentwicklungsansätze, welche der Forderung nach größeren Freiräumen zur Zielerreichung eher entsprechen, wie z. B. die *agentenorientierte Softwareentwicklung* (vgl. [WJ06]). Im folgenden Abschnitt werden die wesentlichen Grundlagen und eine Auswahl von agentenbasierten Ansätzen zur Flexibilisierung von prozessorientierten Anwendungen und ihrer Ausführung kurz vorgestellt.

3.9 Flexibilität durch agentenorientierte Ansätze

Die Unbestimmtheit vieler Anwendungsfälle und die Notwendigkeit zur Unterstützung von eher unstrukturierten Prozessen können in einigen Fällen die Anwendung anderer Softwareparadigmen vorteilhaft machen, die es erlauben, die Arbeitsweise menschlicher Benutzer durch die Automatisierung intelligenten Verhaltens nachzubilden oder sogar gänzlich zu ersetzen. Als ein solches Paradigma hat sich die *agentenorientierte Softwareentwicklung* (*AOSE*) aus dem Forschungsgebiet der (verteilten) künstlichen Intelligenz heraus entwickelt, wobei bei der agentenorientierten Softwareentwicklung die Modellierung und Realisierung des Zusammenspiels autonomer Softwareeinheiten zur interaktiven Ausführung von vorgegebenen Aufgaben im Vordergrund steht [WJ06]. Hieraus ergeben sich verschiedene Einsatzmöglichkeiten für autonome Komponenten in Form von *Softwareagenten*, um zur Flexibilisierung von automatisiert auszuführenden Aufgaben beizutragen.

Nach einer kurzen Einführung werden in diesem Abschnitt zunächst die prinzipiellen Möglichkeiten zur Integration von prozessorientierten Anwendungen und Softwareagenten erarbeitet, bevor im Speziellen die Konzepte der mobilen Agenten, der vorausschauenden Anpassung von Prozessen und des zielorientierten Prozessmanagements als Beispiele für eine Flexibilisierung prozessorientierter Anwendungen vorgestellt werden.

3.9.1 Softwareagenten

Nach den Autoren WOOLDRIDGE und JENNINGS [WJ95, Woo09] stellt ein (Software-)*Agent* im Allgemeinen ein abgrenzbares Computerprogramm dar, welches in einer festgelegten Umgebung eigenständig Aktionen durchführen kann, um vorgegebene Ziele zu erreichen. Hierzu ist ein Agent in der Lage, seine Umgebung wahrzunehmen, selbständig Änderungen in seiner Umgebung zu initiieren oder auf relevante Änderungen seiner Umgebung zu reagieren.

Für die Ausführung seiner Aktionen bzw. Reaktionen kann er sowohl mit Benutzern als auch mit anderen Agenten interagieren. Ein System aus mehreren kollaborierenden Agenten zur Erreichung eines gemeinsamen Ziels wird dabei als *Multi-Agenten-System* (*MAS*) bezeichnet [TvS03, Woo09].

Softwareagenten können nach WOOLDRIDGE und JENNINGS [WJ95, Woo09] zusammengefasst über die folgenden grundlegenden Eigenschaften definiert werden:

- **Autonomie:** Der Agent ist zu eigenständigem Verhalten fähig, d. h. dass abhängig vom Zustand des Agenten bzw. seiner Umgebung ein autonomer Verarbeitungsvorgang abläuft, welcher nicht von außen initiiert oder gesteuert werden muss.
- **Soziale Fähigkeit:** Agenten interagieren über den Austausch von Nachrichten mit anderen Agenten oder mit menschlichen Benutzern. Die Fähigkeit zur Interaktion kann dabei entweder der *Kooperation* oder der *Kompetition* dienen [WJ06].
- **Reaktivität:** Agenten nehmen ihre Umgebung wahr und können zeitnah auf relevante Änderungen dieser Umgebung und in Hinblick auf die Erfüllung ihrer Aufgabe reagieren.
- **Proaktivität:** Agenten können selbst in begrenztem Maße initiativ auf ihre Umgebung einwirken, um zur Erfüllung ihrer Aufgabe Änderungen der Umgebung herbeizuführen.

Neben dieser sogenannten *schwachen Charakterisierung* [WJ95] rückt das Halten von mentalen Zuständen als weitere Eigenschaft eines Agenten die Analogie zum Menschen stärker in den Vordergrund [WJ06]. Vor dem Hintergrund der künstlichen Intelligenz ergibt sich damit nach WOOLDRIDGE und JENNINGS eine *starke Charakterisierung* [WJ95]. Typische Beispiele für mentale Zustände sind Wissen, Intentionen, Ziele oder künstliche Emotionen. Darüber hinaus können einzelne Klassen von Agenten weitere Eigenschaften besitzen, wie zum Beispiel Lernfähigkeit oder Mobilität [TvS03] (vgl. tlw. auch Abschnitt 3.9.4).

Trotz individueller Unterschiede schlägt die *Foundation for Intelligent Physical Agents* (*FIPA*) als Ausgangspunkt für die Bereitstellung allgemein verwendbarer Middleware-Komponenten ein standardisiertes Referenzmodell für die Verwaltung von Software-Agenten mittels *Agenten-Plattformen* vor [TvS03, FIP04]. Eine Agenten-Plattform stellt danach neben grundlegenden Funktionen zur Erzeugung, Registrierung und zum Löschen von Agenten einen (lokalen) Verzeichnisdienst sowie einen Kommunikationskanal bereit, welche zusammen das Veröffentlichen von Eigenschaften, das Suchen und Finden von Agenten sowie deren Nachrichtenaustausch auf fachlicher Ebene durch tlw. standardisierte Agentensprachen unterstützen [FIP04].

Während das Modell kooperierender Agenten durchaus Ähnlichkeiten zur Kooperation in dienstorientierten Architekturen aufweist (vgl. Abschnitt 3.4),

zeichnen sich Agenten im Vergleich zu den eher passiven Diensten durch ihr aktives Verhalten aus, z. B. indem sie aktiv Wissen über ihre Umgebung sammeln und von sich aus die Durchführung von geeigneten Aktionen initiieren [Mas07]. Im folgenden Abschnitt wird daher die Integration von (dienstbasierten) Prozessen und Agenten als mögliche Basis für die Realisierung flexibler prozessorientierter Anwendungen untersucht.

3.9.2 Integration von Agenten-, Dienst- und Prozessorientierung

Prozessorientierte Anwendungen sind in der Regel durch die Kooperation zwischen ansonsten weitestgehend autonomen Ressourcen gekennzeichnet. Sowohl bei organisationsinternen als auch bei organisationsübergreifenden Kooperationen muss dabei oft mit einer hohen Komplexität umgegangen werden, welche neben der Prozessorientierung als zentrales Paradigma der Abbildung eines fachlichen Anwendungsvorgangs die Integration anderer Softwareparadigmen attraktiv macht. In seiner Arbeit adressiert GREFEN [Gre06] daher das prinzipiell mögliche Zusammenspiel von Prozessen (zur strukturierten Spezifikation der durchzuführenden Aufgabe) mit dienstorientierten Architekturen (zur Interoperabilisierung und losen Kopplung von Anwendungskomponenten) und agentenorientierten Konzepten (zur Integration von lokal intelligentem Verhalten).

Abbildung 3.25 zeigt vier mögliche Integrationsvarianten zur Kombination der genannten Softwareparadigmen nach GREFEN [Gre06]. Die erste Möglichkeit setzt die Dienstorientierung (hier in Form von Web Services) als grundlegende Sichtweise voraus und positioniert das Prozessmanagement und die Agenten zusammen auf der darüber liegenden Ebene. Hierbei werden Prozesse und Agenten als Dienste gekapselt, was der weitestgehend vorherrschenden dienstorientierten Sichtweise entspricht [Gre06] (vgl. Abbildung 3.25a). In Hinblick auf die Flexibilisierung prozessorientierter Anwendungen bedeutet dies, dass Dienste mit intelligentem Verhalten existieren bzw. Agenten über Dienstschnittstellen als Ressourcen in einen Prozess eingebunden werden können. Hierdurch kann zum Beispiel eine nur abstrakt beschriebene Aktivität des Prozesses durch einen Agenten in Hinblick auf dessen Wissen über sich und seine Umgebung angepasst bearbeitet werden.

Als zweite Möglichkeit sieht GREFEN die Multi-Agenten-Systeme als grundlegende Sichtweise, so dass Prozessmanagement bzw. Prozesse und Dienste in Form von Agenten miteinander interagieren [Gre06] (vgl. Abbildung 3.25b). Ein Beispiel hierfür ist, den Prozess als Vorgehensweise für das Verhalten eines Agenten zu implementieren. Dadurch wird der Prozess selbst von einer passiven zu einer aktiven Komponente.

Die dritte Möglichkeit betrachtet sowohl das agenten- als auch das dienstorientierte Paradigma als gleichwertige grundlegende Sichtweisen. Hieraus ergibt sich eine kombinierte Infrastruktur aus Diensten und Agenten, welche beide nebeneinander auf jeweils angepasste Weise durch ein übergeordnetes Prozessmanagementsystem genutzt werden können [Gre06] (vgl. Abbildung

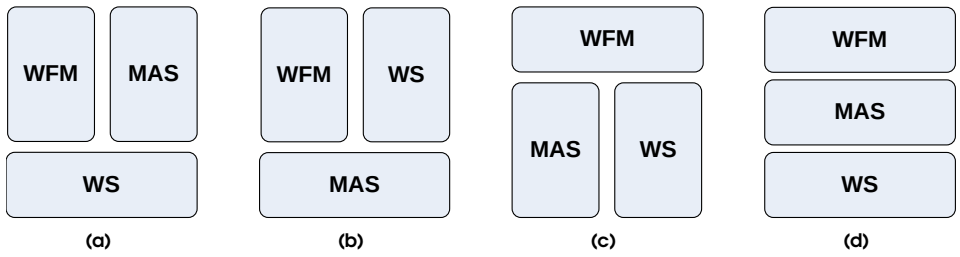


Abbildung 3.25: Integrationsvarianten von Multiagentensystemen (MAS), Web Services (WS) und Workflow Management Systemen (WFM) (Gre06)

3.25c). Ein Beispiel hierfür ist der Einsatz von Agenten zur Aushandlung von Dienstgütevereinbarungen, so dass diese in Abhängigkeit der aktuellen Umgebungsparameter für die Ausführung von dienstbasierten Prozessen individuell angepasst werden können (vgl. z. B. [GALH00]).

Als vierten Ansatz beschreibt GREFEN eine strikte Schichtenarchitektur, welche zunächst eine dienstorientierte Architektur zugrunde legt, darauf ein Multi-Agenten-System aufbaut und als oberste Schicht das Prozessmanagement ansiedelt [Gre06] (vgl. Abbildung 3.25d). Hierdurch kann unter anderem ein zielorientiertes Verhalten der Prozess-Engine erreicht werden, indem Agenten zum Beispiel auf Basis vorgegebener Rahmenbedingungen für die aktive Auswahl und Einbindung von Diensten oder zur Fehlerbehandlung eingesetzt werden (z. B. [FUYP04, GRCB05]). Somit kann auf inhaltlicher und/oder technischer Ebene eine weitere Abstraktionsebene zwischen dem Prozessmanagement und den Ressourcen erreicht werden.

Im Folgenden werden einige für die Flexibilisierung von Prozessen interessante agentenorientierte Ansätze als konkrete Beispiele der genannten Integrationsvarianten genauer betrachtet.

3.9.3 Vorausschauende Anpassung von Prozessen

Eine für die Anpassung von Prozessen besonders interessante Eigenschaft von Software-Agenten ist ihre potentielle Fähigkeit zu proaktivem Verhalten (vgl. Abschnitt 3.9.1). Eine möglichst frühzeitige Erkennung von Anpassungsbedarf und die Durchführung der entsprechenden Anpassung sind im Kontext prozessorientierter Anwendungen insbesondere für die Erkennung und Behandlung von Fehlersituationen relevant. Dabei kann es zum Beispiel aufgrund von fachlichen Ausnahmesituationen notwendig werden, den Kontrollfluss des Prozesses während der Ausführung an die gegebenen Bedingungen anzupassen, z. B. durch das Einfügen oder Entfernen von Aktivitäten oder die Änderung der Transitionsbeziehungen. Eine vorausschauende Anpassung kann dabei zum Beispiel verhindern, dass die verfügbaren Ressourcen zu später unnötigen Aktivitäten zugeordnet werden oder für kurzfristig eingefügte Aktivitäten nicht rechtzeitig zur Verfügung stehen [MGR04]. Um möglichst keine Verzögerungen durch ein manuelles Eingreifen hinnehmen zu

müssen, wird eine Automatisierung des Anpassungsvorgangs durch den Einsatz agentenorientierter Konzepte angestrebt.

Die Autoren MÜLLER, GREINER und RAHM [MGR04, GMR+05] differenzieren zwischen zwei verschiedenen Anpassungsstrategien: Eine *reaktive Anpassung* findet demnach statt, wenn der von einem Fehler betroffene Abschnitt eines Prozesses bereits ausgeführt wird. Dies bedeutet im Speziellen, dass vor jeder auszuführenden Aktivität geprüft wird, ob diese aufgrund der Ausnahmesituation entfernt, verschoben oder ersetzt werden soll [MGR04]. Eine *prädiktive Anpassung* kann vorgenommen werden, wenn das Auftreten eines Fehlers festgestellt oder vorhergesehen werden kann, bevor der betreffende Abschnitt des Prozesses ausgeführt wird. Für eine Automatisierung dieser Anpassungsvorgänge ist zunächst einmal die Erkennung einer fachlichen Ausnahmesituation relevant. Darauf basierend muss festgestellt werden, welcher Teil des Prozesses von der Ausnahmesituation betroffen ist und in welchem Zeitraum die Anpassung durchgeführt werden muss, so dass entschieden werden kann, ob eine prädiktive Anpassung möglich ist oder auf eine reaktive Anpassung zurückgegriffen werden muss. Schließlich muss die Anpassung durch die entsprechenden Operationen durchgeführt und die Durchführung durch geeignete Mechanismen überwacht werden [MGR04].

Für die genannten Aufgaben schlagen MÜLLER, GREINER und RAHM mit *AgentWork* eine Architektur aus vier Agenten vor, welche ein unterliegendes Prozessmanagementsystem um anpassungsfähiges Verhalten erweitern (vgl. Abbildung 3.26). Der *Event Monitoring Agent* entscheidet hierbei auf Basis vorgegebener ECA-Regeln, beim Auftreten welcher Ereignisse es sich um eine relevante fachliche Ausnahmesituation handelt und welche Operationen zur Behandlung dieser Ausnahme durchgeführt werden sollen. Der *Adaptation Agent* entscheidet auf Basis dieser Daten, ob eine reaktive oder prädiktive Anpassung vorgenommen werden soll. Im Fall einer vorausschauenden Anpassung wird der durch die Änderung betroffene Bereich des Prozesses identifiziert. Nach der Bestätigung eines menschlichen Systemadministrators wird die Änderung dann durch den Agenten durchgeführt. Der *Process Monitoring Agent* kontrolliert schließlich, ob die zeitlichen Annahmen des *Adaptation Agents* mit der tatsächlichen Ausführung des Prozesses übereinstimmen. Werden Abweichungen festgestellt, kommt es zu einem erneuten Anpassungsdurchgang. Falls notwendig, können zudem Fehlerbehandlungsmaßnahmen auf andere Prozesse ausgeweitet werden, welche auf entfernten Prozessmanagementsystemen ablaufen. Für eine Entscheidung über eine derartige Eskalation ist der *Interprocess Agent* zuständig.

Der Ansatz setzt im Wesentlichen das in Abschnitt 3.8 beschriebene kontrollschleifenbasierte Architekturmodell autonomer Systeme durch eine Agentenarchitektur um, wobei der wesentliche Unterschied darin besteht, dass bei der von MÜLLER, GREINER und RAHM vorgeschlagenen Architektur insbesondere Ausnahmesituationen auf *fachlicher* Ebene erkannt und *vorausschauend* behandelt werden können. Dabei steht in beiden Fällen die Automatisierung des Anpassungsvorgangs und die Entlastung menschlicher Systemadministra-

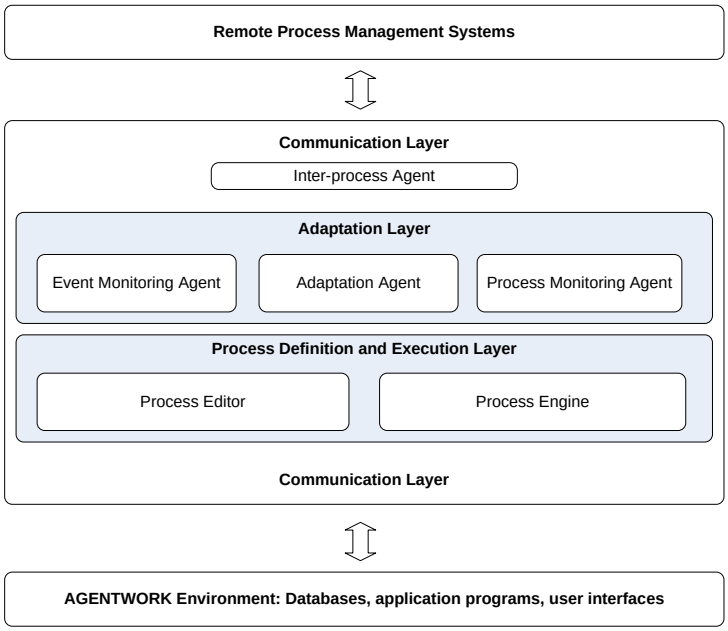


Abbildung 3.26: Architektur von AgentWork (MGR04)

toren und Prozessteilnehmer bzw. die Nachbildung deren Verhaltens bei Entscheidungen bezüglich der Anpassung von prozessorientierten Anwendungen im Vordergrund.

3.9.4 Mobile Agenten

Neben den zuvor genannten Integrationsvarianten besteht zudem die Möglichkeit, dass der Prozess selbst „intelligent“ sein kann. Dies bedeutet insbesondere, dass seine fachliche Logik nicht in Form einer von anderen aktiven Elementen auszuführenden passiven Aufgabenspezifikation beschrieben ist, sondern aktive Inhalte in Form eines ausführbaren Computerprogramms besitzt, welches eigenständig auf den Ressourcen einer bestimmten Ausführungsumgebung arbeiten kann. Diese Sichtweise hebt die in Kapitel 2 getroffene Unterscheidung von explizit beschriebenen fachlichen Prozessen auf Anwendungsebene und beliebigen (autonomen) Computerprogrammen niedrigerer Abstraktionsebenen teilweise auf.

Eine Möglichkeit zur Realisierung einer solchen erweiterten Prozesssicht ist die Ausführung der durch den Prozess vorgegebenen Aufgaben durch einen *mobilen Agenten*. Die Mobilitätseigenschaft eines Agenten erlaubt im Allgemeinen, dass dieser zwischen verschiedenen Computersystemen verschoben werden kann bzw. sich selbst eigenständig auf diese Systeme transferiert [Woo09]. Ein häufiges Anwendungsbeispiel hierfür ist die Suche nach Informationen in heterogenen Datenquellen wie z. B. in verschiedenen Datenbanken oder im Internet [TvS03], wo eine Verschiebung der auszuführenden Funktion (*Code Shipping*) einer Verschiebung der Daten (*Data Shipping*) aufgrund der zu durchsuchenden Datenmenge vorzuziehen ist [Bow01, Bra03].

Das mobilen Agentensystemen zugrunde liegende Prinzip ist das Prinzip der *Code-Migration* [FPV98, MDP+00, TS08]. Als Code-Migration (auch *Code-Mobilität*) wird die Fähigkeit verstanden, dynamisch die Bindungen zwischen Code-Fragmenten und ihrem Ausführungsort ändern zu können [CPV97]. Die Autoren FUGGETTA et al. [FPV98] beschreiben hierfür ein grundlegendes Modell, welches aus einem *Code-Segment* als Menge der Anweisungen des auszuführenden Programms, einem *Ressourcensegment* als Menge der Verweise auf benötigte Ressourcen und einem *Ausführungssegment* als aktuellem Ausführungszustand des Programms aus privaten Daten, Stack und Programmzähler besteht [FPV98, TS08]. Bei einer sogenannten *schwachen Mobilität* [FPV98, IWKK00] wird in der Regel nur das Code-Segment auf ein anderes Ausführungssystem übertragen, so dass das Programm immer von seinem Ausgangszustand gestartet werden muss. Ein Beispiel hierfür sind Java-Applets [TS08]. Bei einer *starken Mobilität* wird zudem gefordert, dass ein Programm während der Ausführung unterbrochen und auf einem anderen Ausführungssystem fortgeführt werden kann [FPV98, IWKK00, TS08]. Hierfür ist der aktuelle Ausführungszustand zu berücksichtigen, so dass neben dem Code-Segment auch das Ausführungssegment zum Zielsystem übertragen werden muss. In beiden Fällen muss zudem geprüft werden, ob es notwendig ist, vom Programm benötigte lokale Ressourcen zu verschieben, zu kopieren, zu ersetzen oder durch einen systemweiten Verweis zugänglich zu machen [FPV98, TS08].

Aufgrund der Autonomie und der Interaktivität von Agenten wird bei der Realisierung mobiler Agentensysteme häufig eine Unterstützung der starken Mobilität bedingt [TS08]. Ein mobiler Agent kann dabei in der Regel im Rahmen seiner Vorgaben selbständig über eine Migration und ihren Zielort entscheiden [MDP+00, DGH03]. Soll eine Migration des Agenten von der aktuellen Ausführungsplattform zu einer Zielplattform stattfinden, so wird der aktuelle Zustand des Agenten zusammen mit seinen Daten und Umgebungsinformationen erhoben, der Anwendungsprozess auf der Quellplattform beendet und die erhobenen Daten in einer Nachricht an die Zielplattform versendet. Dort wird anhand dieser Daten der aktuelle Zustand rekonstruiert, so dass der Agent seine Ausführung dort entsprechend fortsetzen kann [GCK+02, DGH03, TvS03].

Bei der Anwendung der Konzepte von mobilen Agentensystemen auf den Bereich prozessorientierter Anwendungen werden einzelne Prozessinstanzen durch mobile Agenten repräsentiert. Diese Agenten migrieren in Abhängigkeit ihrer Prozessbeschreibung und ihrem internen Zustand von einer Ausführungsumgebung zur nächsten, wo ihnen jeweils spezielle Ressourcen und Kommunikationsmechanismen zur Verfügung stehen [Sch01]. Aufgrund ihrer Eignung für insbesondere verteilte ausgeführte Prozesse werden einige auf mobilen Agenten bzw. auf der Code-Migration basierende Ansätze zur Flexibilisierung der *verteilten Prozessausführung* in Abschnitt 5.5 vorgestellt.

Mobile Agenten bieten aufgrund ihrer inhärenten Eigenschaften ein hohes Maß an Flexibilität. So kann ein Agent als aktiver Vertreter seiner Ziele direkt auf relevante Parameter einer lokalen Umgebung Einfluss nehmen, so dass sowohl eine zeitnahe Ableitung von Anpassungsbedarf als auch deren Durchführung aufgrund lokaler Entscheidungen jederzeit möglich ist. Auf der anderen Seite benötigt die Realisierung der Migration auf jedem teilnehmendem Knoten ein geeignetes Laufzeitsystem, um den Agenten auf unterschiedlichen Systemplattformen eine einheitliche Umgebung bereitzustellen [DGH03]. Ein besonders schwerwiegendes Problem resultiert zudem aus einem uneingeschränkten Zugriff des Agenten auf die Ressourcen der lokalen Ausführungssysteme, da hierbei durch den mobilen Agenten unter Umständen auch vertrauliche Daten ausgespäht oder Systemressourcen manipuliert werden können [TS08, Woo09]. Das unkontrollierte aktive Verhalten von (mobilen) Agenten durch die Ausführung von beliebigem Programmcode ist daher nicht für alle Anwendungsfälle geeignet oder muss durch entsprechende Sicherheitsvorkehrungen begrenzt werden.

3.9.5 Zielorientiertes Prozessmanagement

Wie bereits in Abschnitt 3.3.1 klassifiziert wurde, kann das allgemeine Konzept der *Abstraktion* angewendet werden, um eine fest vorgegebene Struktur für fachlich weniger stark determinierte Prozessen zu vermeiden. Als Beispiel solcher Prozesse nennen BURMEISTER et al. [BSBB06] komplexe Entwicklungsvorgänge im Bereich des Fahrzeugbaus, welche durch eine Mischung von kreativen Aufgaben, kooperativen Arbeiten und strukturiert-wiederkehrenden Aktivitäten gekennzeichnet sind, so dass viele alternative Ausführungspfade und Prozessabschnitte nicht im Voraus planbar sind. Die Idee von *zielorientierten Ansätzen* ist es, die genaue Durchführung von solchen komplexen und in der Regel lang andauernden Prozessen zunächst offen zu lassen und stattdessen das gewünschte übergeordnete Ergebnis der Prozessausführung in den Vordergrund zu stellen [BSBB06, GR07, BPJ+10].

Bei einer *zielorientierten Modellierung* von Prozessen werden die Ziele der Prozessausführung schrittweise durch die Dekomposition einer Zielhierarchie beschrieben, so dass ein übergeordnetes Ziel jeweils durch die Erfüllung einer Reihe von Teilzielen erreicht werden kann. Für jedes (Teil-)Ziel wird eine Menge von Plänen in Form von Modulen spezifiziert, deren Ausführung prinzipiell zu einer Erfüllung des Ziels führt. Die Auswahl eines konkreten Moduls kann hierzu von bestimmten Bedingungen in Bezug auf den aktuellen Kontext der Prozessausführung abhängig gemacht werden, so dass eine situationsgerechte Anpassung zur Laufzeit des Prozesses möglich ist [BSBB06]. Auf der untersten Ebene der Modellierung werden detailliert und ggf. direkt ausführbare Prozessmodelle beschrieben. Diese können entweder statisch vorgegeben sein oder bei Bedarf erzeugt und eingesetzt werden [BSBB06] (vgl. auch *Late Modelling* in Abschnitt 3.3.1). Abbildung 3.27 verdeutlicht diese hierarchischen

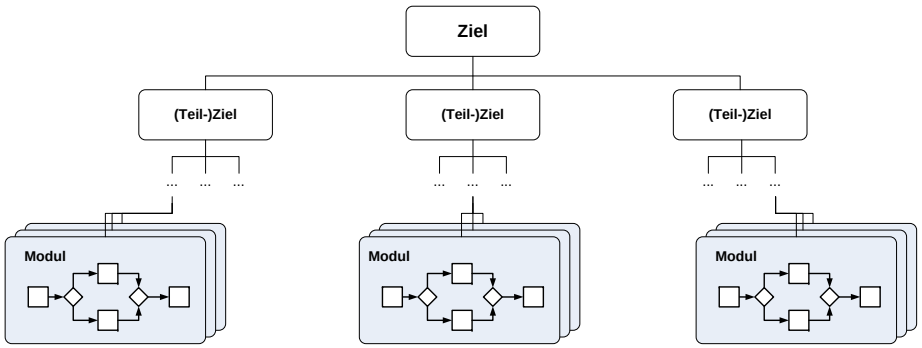


Abbildung 3.27: Zielorientierte Modellierung von Prozessen (nach (BSBB06))

Beziehungen zwischen Zielen und Teilprozessen als Abbildung eines gesamten prozessorientierten Anwendungsfalls.

Auf konzeptioneller Ebene folgt die dargestellte Art der Modellierung und insbesondere der Vorgang der Zieldeliberation dem Vorbild der Architektur von *BDI-Agenten* [BSBB06, GR07]. Bei einem BDI-Agenten handelt es sich um einen Software-Agenten mit mentalen Zuständen, welche durch das Wissen des Agenten über sich und seine Umgebung (*Beliefs*), durch seine Wünsche als erstrebenswerte Zustände (*Desires*) und durch seine Intentionen als Absicht zur Verfolgung bestimmter Ziele (*Intentions*) beschrieben werden können [Bra87, RG95]. Zur Verfolgung seiner Ziele besitzt der BDI-Agent *Pläne*, welche anhand von bestimmten Aktionen beschreiben, wie diese Ziele erreicht werden können. Auf der Basis der ihm vorliegenden Umgebungsinformationen kann der Agent somit Entscheidungen darüber treffen, welche Aktion als nächstes auszuführen ist, um die ihm vorgegebenen Ziele zu erreichen. Im Fall des zielorientierten Prozessmanagements kann der BDI-Agent (unter Umständen gemeinsam mit anderen Agenten) die Entscheidung treffen, welches Modul zur Erfüllung welcher Ziele herangezogen werden soll [BSBB06].

Durch eine Modularisierung des Gesamtprozesses und die Separierung der Zielspezifikation von der Spezifikation ausführbarer (Teil-)Prozesse ergibt sich ein hohes Maß an Flexibilität auf fachlicher Ebene des Prozesses. Zum einen kann die Auswahl der konkret durchzuführenden Aufgaben von aktuellen Umgebungsparametern abhängig gemacht werden und das intelligente Verhalten von Agenten genutzt werden, um den als nächstes auszuführenden Teilprozess zur Erreichung des übergeordneten Ziels zu bestimmen. Zum anderen wird durch die Modularisierung die Wartbarkeit einzelner Teilprozesse verbessert. Durch die Separierung können neben Ausführungssystemen für BDI-Agenten und/oder fachlichen Regeln zudem die bereits bestehenden Prozessmanagementsysteme zur Ausführung der ausgewählten Teilprozesse auf der untersten Ebene dieser Modellierungshierarchie weiter verwendet werden [BSBB06]. Dies birgt vor allem hinsichtlich der praktischen Einsetzbarkeit dieses Konzepts große Vorteile.

3.10 Zusammenfassung

In diesem Kapitel wurde die Flexibilität und Anpassbarkeit der in Kapitel 2 allgemein eingeführten prozessorientierten Anwendungen untersucht. Flexibilität ermöglicht in dieser Hinsicht Anpassungen auf Änderungen der Ausführungsumgebung einer prozessorientierten Anwendung, wobei neben extrinsischen Faktoren auch die prozessorientierte Anwendung selbst, das ausführende Prozessmanagementsystem und die beteiligten Ressourcen Teil des relevanten Kontextes eines Prozesses sein können. Relevante Umgebungsänderungen können technisch über Ereignisse kommuniziert und durch verantwortliche (Teil-)Systeme verarbeitet werden. Die Art der Umgebungsänderung (bzw. des Ereignisses) bestimmt dabei die erforderliche Reaktion des betrachteten Systems und damit die Art der hierfür benötigten Flexibilität:

- Vorkommnisse mit fachlichen Auswirkungen erfordern inhaltliche Anpassungen am Prozessmodell oder an einzelnen Prozessinstanzen, welche jedoch oft auch auf organisatorischer bzw. technischer Ebene eine Neuzuordnung von Ressourcen und ggf. eine Anpassung der unterliegenden IT-Infrastruktur nach sich ziehen.

- Vorkommnisse mit organisatorischen oder technischen Konsequenzen machen eine Anpassung der Art und Weise der Ausführung unter Beibehaltung der Prozessinhalte notwendig.

In beiden Fällen ist eine flexible Basisstruktur für Informations- und Kommunikationssysteme erforderlich, welche möglichst a-priori die Möglichkeit zur fortwährenden Anpassung der prozessorientierten Anwendung und ihrer Ausführung erlaubt. Als zentrale Konzepte zur Bildung einer flexiblen Infrastruktur wurden aktuelle Methoden zur Entkopplung von fachlicher (Geschäfts-)Logik der Prozesse und technischer (Management-)Logik der involvierten Informations- und Kommunikationssysteme identifiziert. Durch dienstorientierte und ereignisgesteuerte Architekturen sowie durch die Anwendung regelbasierter Systeme werden *Abstraktion*, *lose Kopplung* und *variable Bindungszeitpunkte* als besonders wirkungsvolle Methoden zur Erhöhung von Flexibilität im Kontext prozessorientierter Anwendungen propagiert. Des Weiteren wird der Erkennung von Änderungsbedarf, der Ermöglichung von (echt-)zeitnahen Reaktionen oder sogar von proaktiven Anpassungen sowie der weitestgehenden Automatisierbarkeit der Erkennung und Anpassung eine große Bedeutung zugesprochen.

Abbildung 3.28 enthält eine ablauforientierte Übersicht der wichtigsten identifizierten Flexibilisierungsanforderungen und Anpassungsmöglichkeiten für prozessorientierte Anwendungen im Allgemeinen und eine grobe Zuordnung der genannten Forschungsgebiete. Die Flexibilität der Informations- und Kommunikationssysteme stellt hier die Basis für die Ermöglichung von Anpassungen sowohl der funktionalen (fachlichen) als auch der nicht-funktionalen

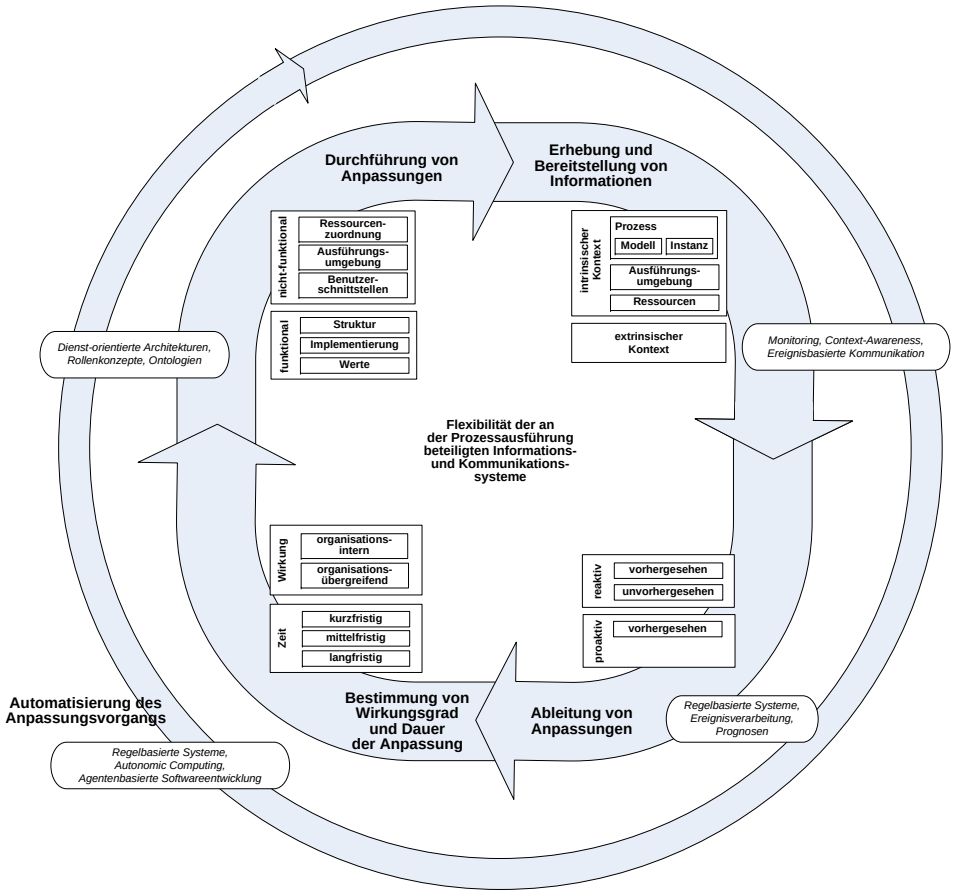


Abbildung 3.28: Flexibilität als Basis für (automatisierte) Anpassungsvorgänge prozessorientierter Anwendungen

(organisatorisch/technischen) Ebene dar. Dabei können Anpassungen – in Abhängigkeit der Dynamik der Umgebung – von verschiedener Dauer sein (d.h entweder Prozessmodelle oder Prozessinstanzen betreffen) und sich organisationsintern oder organisationsübergreifend auswirken. Voraussetzung für eine erfolgreiche Anpassung ist jedoch die vorhergehende Erhebung und Bereitstellung von relevanten Informationen, um überhaupt Änderungsbedarf erkennen und eine konkrete Art der Anpassung ableiten zu können. Die Möglichkeit, diesen gesamten Anpassungsvorgang durch Informations- und Kommunikationstechnologie zu unterstützen (und damit zu automatisieren), bringt letztendlich die angestrebten Vorteile in Bezug auf Reaktionsgeschwindigkeit, Fehlervermeidung und Entlastung menschlicher Prozessverantwortlicher.

Eine organisationsübergreifende Anpassung von prozessorientierten Anwendungen wurde bislang nur in Hinblick auf die Integration und den Austausch von einzelnen Ressourcen externer Organisationen bzw. Organisationseinheiten betrachtet. Dabei ist es im Rahmen von dienstorientierten Architekturen möglich, je nach Bedarf interne und externe Ressourcen in Form von elektronischen Diensten dynamisch aufzufinden und in die betreffende prozessorientierte Anwendung einzubinden. Somit kann bereits ein gewisses Fle-

xibilisierungspotential auf der Ebene *innerhalb eines Systems* und *außerhalb eines einzelnen Systems in der Interaktion mit anderen Systemen* erreicht werden. Ein weiterer Schritt in diese Richtung stellt die Abbildung von organisationsübergreifenden Prozessen dar, welche *über mehrere Systeme verteilt ausgeführt* werden.

Im weiteren Verlauf dieser Arbeit wird die Verteilung der Prozessausführung als Objekt der Flexibilisierung untersucht. Die Verteilung der Kontrolle über die Ausführung eines Prozesses und die aus der Kooperation mit anderen Organisationen resultierende Dynamik, die auf die Ausführung des Prozesses einwirkt, machen eine Überprüfung der Flexibilisierungsanforderungen und -möglichkeiten konventioneller (nicht-verteilter) Prozesse notwendig. Im folgenden Kapitel werden daher verteilt ausgeführte Prozesse als Kernthema dieser Arbeit genauer betrachtet. Die in diesem Kapitel erworbenen Erkenntnisse über die Flexibilität und Anpassungsfähigkeit von prozessorientierten Anwendungen im Allgemeinen werden dann auf die Flexibilität und die weitere Flexibilisierung von verteilten Prozessen im Speziellen übertragen und auf ihre Anwendbarkeit überprüft.

4 Dynamisch verteilt ausgeführte Prozesse

Der Anwendungsbereich und die Anforderungen prozessorientierter Anwendungen haben sich insbesondere in den letzten Jahren stark erweitert. Zum einen hat sich die Automatisierung von Vorgängen der Realwelt von einem betriebswirtschaftlich geprägten Kontext auch in andere Anwendungsdomänen fortgepflanzt. Zum anderen ist durch die Verbreitung von Informations- und Kommunikationstechnologien – insbesondere durch die rasche Ausbreitung des Internets und der Mobilkommunikation – die ubiquitäre Verfügbarkeit von Daten und Funktionalitäten stark angestiegen. Durch die zunehmende Vernetzung von Hard- und Softwarekomponenten wird die Nutzung von verteilten technischen Infrastrukturen und Ressourcen erleichtert und dadurch auch die Kooperation zwischen verschiedenen Organisationen oder Organisationseinheiten ermöglicht. Die Abbildung und technische Unterstützung solcher systemübergreifender Vorgänge stellt daher einen wesentlichen Teil der Forschung auf dem Gebiet prozessorientierter Anwendungen dar (vgl. [JSH+01, All05, HSH+06, Wes07, Gad10]).

Die Verteilung von Prozessen und die Abbildung bereits von Natur aus verteilter Abläufe zwischen unterschiedlichen Systemen umfasst ganz unterschiedliche Anwendungsdomänen und ist häufig von einer hohen Dynamik geprägt. Bereits in länger zurückliegenden Arbeiten betont VAN DER AALST [vdA99, vdA00] die zunehmende Relevanz von organisationsübergreifenden Prozessen im Rahmen des *Electronic Commerce* und im Kontext *virtueller Unternehmen*, welche durch häufige Änderungen und Anpassungen an dynamische Marktsituationen gekennzeichnet sind. Die Abbildung von Prozessen zwischen unterschiedlichen staatlichen Einrichtungen im *E-Government*, die Integration von *mobilen Geräten* sowie moderne Strategien zur *Lastverteilung* oder zur *Optimierung* der Prozessausführung machen ebenfalls häufig eine flexibel verteilte Ausführung von Prozessen erforderlich. Entsprechend zur Laufzeit in Bezug auf ihre Verteilung an die vorherrschenden Bedingungen und Anforderungen angepasste Prozesse werden in dieser Arbeit als *dynamisch verteilt ausgeführte Prozesse* bezeichnet und in diesem Kapitel genauer untersucht.

Als Unterscheidung zu anderen Arten der Verteilung im Kontext prozessorientierter Anwendungen wird zunächst die *verteilte Ausführung* eines (inhaltlich strukturierten und stabilen) Prozesses als zusätzlicher Mechanismus zur Anpassung der Ausführung prozessorientierter Anwendungen abgegrenzt (vgl. Abschnitt 4.1). Im Anschluss daran werden zur Analyse der konkreten Anforderungen einer dynamisch verteilten Prozessausführung die Ziele und Eigenschaften derartiger Prozesse genauer untersucht. Hierzu werden sowohl konkrete Fallbeispiele (vgl. Abschnitt 4.2) als auch eine Literaturrecherche zu klassischen Verteilungsmodellen für eine Klassifizierung verteilt ausgeführter

Prozesse herangezogen (vgl. Abschnitt 4.3). Aus den erarbeiteten Erkenntnissen werden im Folgenden Anforderungen abgeleitet, welche schließlich auf ein erweitertes Lebenszyklusmodell verteilt ausgeführter Prozesse abgebildet werden, um den relevanten Anpassungsbedarf den entsprechenden Lebenszyklusphasen prozessorientierter Anwendungen zuzuordnen (vgl. Abschnitt 4.4). Das Kapitel schließt mit einer Zusammenfassung der erarbeiteten Erkenntnisse (vgl. Abschnitt 4.5).

4.1 Einführung und Begriffsklärung

Wie in Abschnitt 2.7.1 erläutert wurde, können prozessorientierte Anwendungen bereits als von sich aus verteilt betrachtet werden. Die entsprechenden Ausführungssysteme (wie Prozess-Engine, Integrationsdienste und kommunikationsorientierte Komponenten) lassen sich dabei der Klasse der Middleware zuordnen (vgl. auch [Jab97, CRW98, JSH+01]). In der weiterführenden Literatur geht das Verständnis von *Verteilung* in Bezug auf Prozesse, deren Ausführung und der daran beteiligten Systeme jedoch weit auseinander. An dieser Stelle soll daher auf Basis der grundlegenden Definition eines *verteilten Systems* der Begriff der *verteilten Prozessausführung* geklärt werden. Dabei wird insbesondere der für diese Arbeit wichtige Unterschied zwischen einer *zentral verwalteten* Ausführung von Prozessen und einer *verteilt verwalteten Ausführung* von Prozessen herausgearbeitet.

4.1.1 Verteilte Systeme

Ein verteiltes System stellt eine Menge voneinander unabhängiger Funktionseinheiten [Ben04] (bzw. Computer oder Computerkomponenten [TS08]) dar, welche sich dem Benutzer oder Anwendungsentwickler als ein einzelnes, zusammenhängendes System darstellen [TS08]. Die beteiligten Einheiten verfügen dabei in der Regel nicht über einen gemeinsamen Speicher und kommunizieren miteinander über Nachrichten [CDK05, TS08]. Sie realisieren zusammen eine Funktion, welche in Umfang oder Qualität nicht von einer Einheit des Systems alleine erbracht werden kann [Ben04].

Die Eigenschaft der *Autonomie* der beteiligten Funktionseinheiten, die *Kooperation* zum Erreichen eines gemeinsames Ziels (z. B. durch den Zugriff auf gemeinsame Ressourcen) sowie die *Transparenz der Verteilung* stellen dabei wesentliche Charakteristika verteilter Systeme dar. Im Kontext verteilter Prozesse stellt dabei insbesondere die Transparenz der Verteilung einen wichtigen, weiter zu untersuchenden Aspekt dar. Nach TANENBAUM [TS08] können sieben unterschiedliche Arten von Verteilungstransparenzen identifiziert werden (vgl. auch [Ham05, Ill07]):

- **Ortstransparenz**: Der Benutzer oder Anwendungsentwickler hat keine oder nur eingeschränkte Kenntnis über den physikalischen Aufenthaltsort einer Funktionseinheit. Der Zugriff auf Ressourcen (d. h. Daten oder Funktionen) erfolgt mittels eines eindeutigen Namens.

- **Zugriffstransparenz**: Für den Benutzer oder Anwendungsentwickler sind lokale und entfernte Zugriffe identische Mechanismen. Es existiert kein spürbarer Leistungsunterschied zwischen einer lokalen und einer entfernten Bearbeitung (*Leistungstransparenz* [Ill07]).
- **Replikationstransparenz**: Es ist nicht sichtbar, ob – und falls ja, wieviele – Replikate eines Objekts im verteilten System existieren.
- **Migrationstranparenz**: Ressourcen können physikalisch bewegt werden, ohne dass ihr Name oder die Art und Weise ihres Zugriffs verändert werden muss.
- **Relokationstransparenz**: Ressourcen können physikalisch bewegt bzw. verschoben werden, während sie genutzt werden.
- **Nebenläufigkeitstransparenz**: Mehrere Benutzer können gemeinsame Objekte verwenden, ohne dass es zu Inkonsistenzen kommt. Aktivitäten können parallel ablaufen, ohne sich gegenseitig zu stören.
- **Fehlertransparenz**: Das Gesamtsystem wird in der Regel nicht von dem Ausfall einer einzelnen Teilkomponente beeinflusst.

Es ist wichtig anzumerken, dass es sich bei den angegebenen Transparenzarten jeweils um graduelle Eigenschaften handelt, welche nicht immer vollständig erreicht werden oder auch nicht immer vorteilhaft sein müssen [TS08]. So ist zum Beispiel der Standort eines Webservers für den Besucher einer Webseite im Internet nicht relevant, während der geographische Aufenthaltsort eines RFID-fähigen Systems zur Lokalisierung einer Warenlieferung sogar den wesentlichen Anwendungszweck dieser Funktionseinheit darstellt.

Weitere wichtige Eigenschaften bzw. Entwicklungsziele eines verteilten Systems sind das Vorhandensein offener Schnittstellen (*Offenheit*) und darauf aufbauend eine einfache und effiziente Erweiterbarkeit (*Skalierbarkeit*) des Gesamtsystems [TS08]. Insbesondere in Hinblick auf die Skalierbarkeit und die Ausfallsicherheit verteilter Anwendungen stellt die Systemarchitektur verteilter Funktionseinheiten einen wesentlichen Aspekt dar. Man unterscheidet zwischen *zentralisierten Architekturen* (z. B. dem klassische Client-Server-Modell) und *dezentralisierten Architekturen* (z. B. dem Peer-to-Peer-Modell) [Ham05, TS08]. In vielen Fällen wird dabei die Skalierbarkeit von dezentralisierten Architekturen besser bewertet, da durch die Nutzung zentraler Komponenten bei einer Erhöhung der Teilnehmerzahl an dieser Stelle oft Engpässe entstehen können (*Bottlenecks*) [Ben04, CDK05]. Der Preis für die Vermeidung solcher Engpässe ist in der Regel jedoch ein höherer Verwaltungs- und Kommunikationsaufwand [TS08].

Der Begriff der *Verteilung* wird im Wesentlichen verwendet, um entweder den *statischen Zustand* auszudrücken, dass Funktionseinheiten verschiedenen Orten zugeordnet sind, oder um den *Vorgang der Zerlegung* und der Zuordnung der entstehenden Teileinheiten zu beschreiben [Sch01]. Im Folgenden werden

beide Formen der Verteilung im Kontext prozessorientierter Anwendungen untersucht.

4.1.2 Verteilung im Kontext prozessorientierter Anwendungen

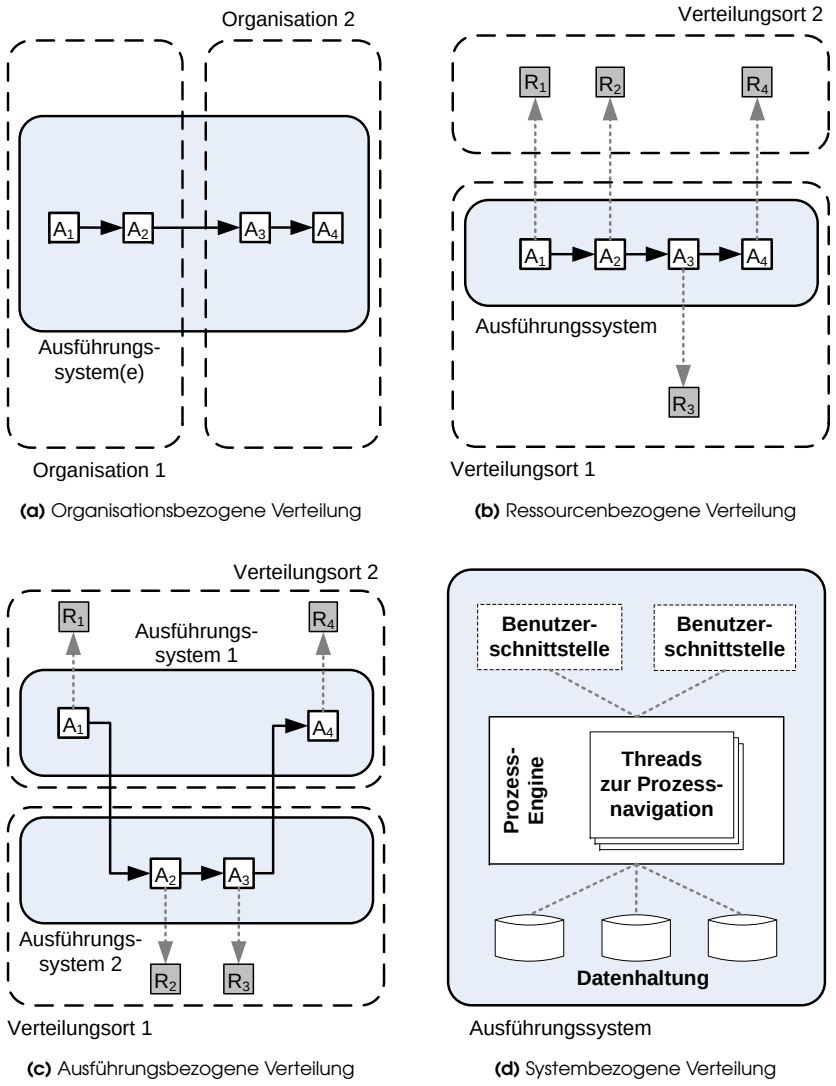
Prozessorientierte Anwendungen und verteilte Systeme stehen inhärent und unmittelbar miteinander in Beziehung [JSH+01]. Die in Abschnitt 2.7 genannten Eigenschaften der allgemeinen Prozessausführung entsprechen den Kernanforderungen an verteilte Systeme nach *Verteilungstransparenz* und einem *gemeinsamen Zugriff auf Ressourcen*. Die variable Einbindung von verteilten Ressourcen (sowohl innerhalb als auch außerhalb einer Organisation bzw. einer Organisationseinheit) kann durch die Bereitstellung *offener Schnittstellen* zur Nutzung dieser Ressourcen zum Beispiel im Rahmen von dienstorientierten Architekturen realisiert werden. Eine potentielle Verteilung der Ressourcen oder der Softwarekomponenten einer Ausführungseinheit ist dabei ebenfalls transparent und hat auf die Ausführung und Anpassung einer ausgeführten prozessorientierten Anwendung in der Regel keinen funktionalen Einfluss.

Für ein Verständnis des Begriffs der *verteilten Prozessausführung* ist daher zunächst einmal eine genaue Abgrenzung des Begriffs von anderen Arten der Verteilung im Kontext prozessorientierter Anwendungen erforderlich. Folgende Arten von Verteilung in Bezug auf Prozesse können in der eingesehenen Literatur identifiziert werden (vgl. auch tlw. [JSH+01, Sch01]):

- **Organisationsbezogene Verteilung:** Ein Prozess überschreitet die Grenzen einer Organisation, wenn mindestens eine Aktivität des Prozesses von einer anderen Organisation ausgeführt oder verwaltet wird [KWA99a]. VAN DER AALST spricht hier allgemein auch von *organisationsübergreifenden Prozessen* (*cross-organizational* bzw. *interorganizational processes*) [vdA99, vdA00, Wes07], wobei die Art und Weise der Verteilung nicht näher defniert wird (vgl. Abbildung 4.1a).

- **Ressourcenbezogene Verteilung:** Hierbei sind die Ressourcen eines Prozesses (physikalisch, logisch oder organisatorisch) verteilt, wobei die Kontrolle einem *zentralen Ausführungssystem* zugeordnet ist (vgl. Abbildung 4.1b) [vdA99, vdA00]. Diese Art der Verteilung erfolgt zum Beispiel durch die Komposition von elektronischen Diensten im Rahmen von Orchestrierungen (vgl. Abschnitt 3.4.4).

- **Ausführungsbezogene Verteilung:** Die Ausführung des Prozesses erfolgt verteilt über mehrere Ausführungssysteme, wobei der Prozess bzw. die Ausführung des Prozesses partitioniert wird (vgl. Abbildung 4.1c). Diese Art der Verteilung erfolgt zum Beispiel durch die Abbildung von organisationsübergreifenden Kommunikationsvorgängen als Choreographie (vgl. Abschnitt 3.4.5).



(a) Organisationsbezogene Verteilung **(b)** Ressourcenbezogene Verteilung

(c) Ausführungsbezogene Verteilung **(d)** Systembezogene Verteilung

Abbildung 4.1: Verteilung im Kontext prozessorientierter Anwendungen

- **Systembezogene Verteilung:** Auch ein einzelnes Ausführungssystem (in der Regel das Prozessmanagementsystem) kann verteilt werden, so dass die entsprechenden Komponenten der softwaretechnischen Realisierung auf unterschiedlichen Hardwaresystemen angeordnet sind (z. B. [HPA05, HPA06]) (vgl. Abbildung 4.1d). Dieses Verständnis der Verteilung umfasst auch den gängigen Aufbau klassischer Workflow-Managementsysteme als Drei-Schichten-Architektur aus Datenhaltungskomponente, Workflow-Engine und Workflow-Client (vgl. [GG99, JSH+01, Gad10]).

Der Fokus dieser Arbeit liegt auf der **ausführungsbezogenen Verteilung**, wobei die Kombination aus einem Ausführungssystem mit allen Ressourcen, welche von der jeweiligen Ausführungseinheit zur

Ausführung von Aktivitäten prozessorientierter Anwendungen zugegriffen werden können, jeweils als *Ausführungsumgebung* bezeichnet wird. Dabei können Ressourcen, wie zum Beispiel global zugreifbare Dienste, durchaus von verschiedenen Ausführungseinheiten aufgerufen und somit mehreren Ausführungsumgebungen zugeordnet sein. Die ressourcenbezogene Verteilung wird daher durch die ausführungsbezogene Verteilung subsumiert.

In Hinblick auf die Flexibilität von prozessorientierten Anwendungen ist vor allem das Verhalten von Prozessen zur Laufzeit und seine dynamische Anpassungsfähigkeit interessant. Wird im Folgenden von einem Prozess gesprochen, so ist daher stets die ausführbare Variante des technischen Prozessmodells bzw. seiner Prozessinstanzen gemeint (vgl. Abschnitt 2.3). Auf Basis bisheriger Flexibilisierungsansätze (vgl. Kapitel 3) wird davon ausgegangen, dass Ressourcen lose gekoppelt und in der Art einer dienstorientierten Architektur jeweils durch eine lokale Softwareanwendung gekapselt sein können. Ressourcen können somit zur Entwicklungs- oder zur Laufzeit automatisiert aufgefunden und dynamisch in einen Prozess eingebunden werden. Zudem können die Softwarekomponenten einer Ausführungsumgebung durchaus selbst verteilt realisiert sein, z. B. durch die Nutzung verteilter Datenbanken. Eine organisationsübergreifende Verteilung muss schließlich insbesondere in Bezug auf die etwaige *Autonomie* teilnehmender Parteien berücksichtigt werden. Die Variante der ausführungsbezogenen Verteilung kann daher nicht isoliert betrachtet werden, sondern wird im Folgenden immer im Kontext der anderen genannten Verteilungsarten gesehen.

4.1.3 Dynamisch verteilte Prozessausführung

In dieser Arbeit soll der spezielle Fall betrachtet werden, dass die *Ausführung* eines (von Struktur und Inhalt vorgegebenen) Prozesses verteilt erfolgt, d. h. dass potentiell mehrere (lokale oder entfernte) Ausführungssysteme an der Navigation des Kontrollflusses und somit an der Auswahl und der Einbindung von Ressourcen beteiligt sind. Die für die Ausführung von Aktivitäten des Prozesses ausgewählten Ressourcen (d. h. Softwareanwendungen, Maschinen oder Personen) können dabei ebenfalls verteilt sein. Ressourcen und Ausführungssysteme können bei diesem Szenario zudem unterschiedlichen Organisationen bzw. Organisationseinheiten angehören, wobei eine Organisation bzw. Organisationseinheit jeweils auch mehrere Ausführungssysteme betreiben kann:

> *Ein **verteilt ausgeführter Prozess** ist ein Prozess auf Anwendungsebene, bei dessen Ausführung verschiedene Hard- und/oder Softwaresysteme für die automatisierte Verwaltung des Prozesses, d. h. für die Bearbeitung des Kontrollflusses und den Aufruf von Ressourcen, verantwortlich sind.*

Im Kontext verteilt ausgeführter Prozesse können dabei auch die für verteilte Systeme im Allgemeinen genannten unterschiedlichen Systemarchi-

tekturen zum Einsatz kommen: Die Prozessausführung kann verteilt erfolgen, aber dennoch einer *übergeordneten zentralen Steuerung* unterliegen. Zum anderen kann die verteilte Prozessausführung (vollständig) *dezentral* erfolgen. In beiden Fällen müssen die Interaktionen zwischen den beteiligten Ausführungssystemen einheitlich geregelt und die Schlüsselschnittstellen der Hardware- bzw. Softwarekomponenten offen gelegt sein. Die verteilte Ausführung eines Prozess *kann* – aber *muss* nicht zwingend - für den Prozessinitiator oder für die beteiligten Ressourcen *transparent* erfolgen (vgl. Abschnitt 4.1.1). Wie im nächsten Abschnitt gezeigt werden wird, kann die Lokalität der Ausführung in einigen Fällen eine wichtige Eigenschaft oder sogar den Auslöser für die Verteilung von Prozessen darstellen.

Die Verteilung als Vorgang des Aufteilens eines Prozesses zur Zuordnung an verschiedene Ausführungseinheiten wird als *Partitionierung* bezeichnet und die resultierenden Prozessabschnitte stellen entsprechend *Partitionen* des Prozesses dar. Zentraler Aspekt dieses Vorgangs ist die Entscheidung über die *Granularität* der Verteilung und die *Zielorte* der Ausführung. Die Granularität der Verteilung beschreibt, ob ein Prozess in seiner Gesamtheit einem Ausführungssystem zugeteilt wird (*zentrale Ausführung*), ob der Prozess in einzelne Teilprozesse mit einer jeweils beliebigen Anzahl von Aktivitäten zerlegt wird, oder ob jede einzelne Aktivität einer anderen Ausführungseinheit zugewiesen wird (*maximal verteilte Ausführung* bzw. *Vollverteilung*) [Sch01]. Zwischen den Extremfällen der zentralen Ausführung eines ganzen Prozesses und einer maximal verteilten Ausführung muss in Hinblick auf ein bestimmtes Optimierungskriterium eine sinnvolle Partitionierung bestimmt werden. Diese Partitionierung kann dabei auf der Ebene einzelner *Prozessmodelle* (d. h. zur Entwicklungszeit) und/oder auf der Ebene einzelner *Prozessinstanzen* (d. h. zur *Laufzeit*) stattfinden. Die Verteilung der Prozessausführung kann diesen Verteilungszeitpunkten entsprechend *statisch* oder *dynamisch* festgelegt werden. Bei einer statischen Verteilung werden Prozesspartitionen bereits zur Entwicklungszeit bestimmt und beim Deployment der prozessorientierten Anwendung den potentiellen Zielorten der Verteilung (d. h. konkreten Ausführungseinheiten) zugeordnet. Bei einer dynamischen Verteilung geschieht dieser Vorgang erst während der Ausführung einer individuellen Prozessinstanz (vgl. [Sch01]):

> *Ein **dynamisch verteilt ausgeführter Prozess** ist ein Prozess auf Anwendungsebene, bei dem die Partitionierung jeder einzelnen Prozessinstanz und die für die Ausführung der resultierenden Prozesspartitionen verantwortlichen Hard- und/oder Softwaresysteme zur Laufzeit des Prozesses bestimmt werden.*

Dynamisch verteilt ausgeführte Prozesse versprechen ein großes Maß an Anpassungsfähigkeit an veränderte Umgebungsbedingungen, verfügen dabei jedoch selbst über besonders hohe Flexibilitätsanforderungen. Im Folgenden werden daher die Relevanz und die Hintergründe von dynamisch verteilt aus-

geführten Prozessen anhand von verschiedenen Fallbeispielen motiviert, um deren Anforderungen zu erfassen und diese später mit bereits bestehenden Ansätzen für verteilt ausgeführte Prozesse abzugleichen.

4.2 Anwendungsgebiete und Fallbeispiele

Eine Verteilung der Prozessausführung kann zum einen durch die Abbildung eines natürlich verteilten Prozesses der Realwelt begründet sein oder künstlich geschaffen werden, um bestimmte Rahmenbedingungen der Ausführung zu schaffen oder zu optimieren. In vielen Anwendungsfällen sind die Verteilung eines Prozesses bzw. dessen Interdependenzen zu anderen (Teil-)Prozessen statisch festgelegt oder ändern sich im Rahmen der Prozessevolution innerhalb des normalen Lebenszyklus der prozessorientierten Anwendung nur selten. Oft wird die Verteilung eines Prozesses jedoch auch durch das plötzliche Eintreten unerwarteter Ereignisse oder das Auftreten besonderer Umgebungsänderungen ausgelöst, welche bei der Modellierung des Prozesses noch nicht berücksichtigt wurden oder deren vorausschauende Berücksichtigung zu aufwendig wäre. In diesem Abschnitt wird anhand von verschiedenen Fallstudien aus Literatur und Praxis gezeigt, dass insbesondere bei zu verteilten Prozessen führenden Anwendungsfällen oft eine besonders hohe Dynamik auftritt und damit eine hohe Anpassungsfähigkeit des Prozesses erforderlich ist. Dies liegt zum einen an den speziellen Anwendungsdomänen verteilt ausgeführter Prozesse, wie den Bereichen virtueller Unternehmen, der Integration mobiler Geräte oder der Optimierung nicht-funktionaler Eigenschaften. Zum anderen entstehen durch die Verteilung selbst spezielle Anforderungen, welche die Anpassung von Prozessen und ihrer Verteilung beeinflussen. Durch die Betrachtung der folgenden Fallbeispiele kann dabei auf viele relevante Eigenschaften geschlossen werden, welche einer flexiblen Verteilung der Prozessausführung zugrunde liegen müssen.

4.2.1 Unternehmensübergreifende Geschäftsprozesse

Eine typische Anwendungsdomäne für die verteilte Ausführung von prozessorientierten Anwendungen ist die Zusammenarbeit zwischen verschiedenen Unternehmen zur Fertigung und zum Vertrieb eines Produkts oder zum Angebot einer (komplexen) Dienstleistung. Hierfür ist in der Regel eine technische Kopplung der beteiligten Unternehmensysteme sowie eine organisatorische Abstimmung der betrieblichen Abläufe erforderlich [All05]. Die mit Hilfe von Informations- und Kommunikationstechnologien (insbesondere von Internettechnologien) durchgeführten unternehmensübergreifenden Geschäftsprozesse werden dabei auch unter dem Stichwort *Electronic Business* (*E-Business*) zusammengefasst [vdA99, All05, Gad10].

Eine computergestützte Durchführung von unternehmensübergreifenden Prozessen orientiert sich häufig entlang der Lieferkette von Waren und Dienstleistungen, die zur Erstellung höherwertiger Produkte benötigt werden (*Sup-*

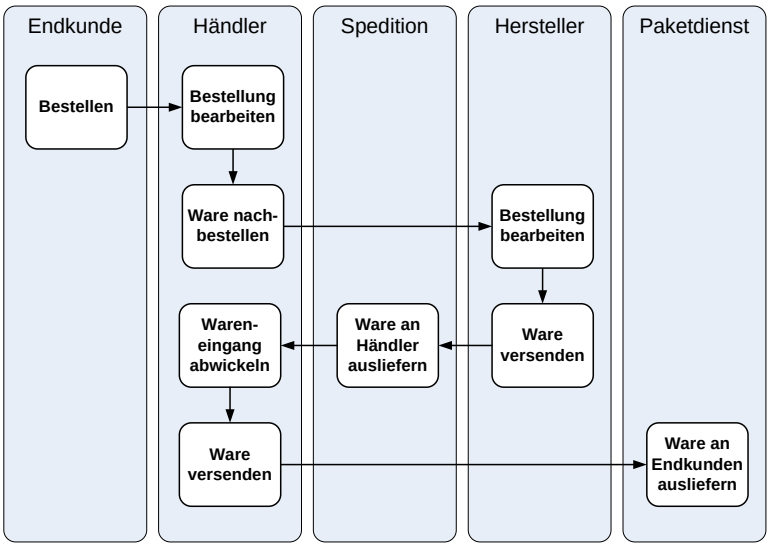


Abbildung 4.2: Unternehmensübergreifende Abwicklung eines Bestellprozesses bei einem Versandhändler (All05)

ply Chain Management). Als Beispiel zeigt Abbildung 4.2 einen vereinfachten Prozess zur Abwicklung eines Kundenauftrags bei einem Versandhändler, welcher zur Durchführung seiner Leistung Waren beim Hersteller bezieht und für die An- bzw. Auslieferung der Waren eine Spedition und einen Paketdienst involviert (vgl. [All05]). In modernen Unternehmen kann die Verwaltung eines solchen Vorgangs fast vollständig automatisiert bzw. weitestgehend durch Informations- und Kommunikationstechnologie unterstützt werden. Der Kunde kann Waren über ein Unternehmensportal im Internet bestellen, ein computergestütztes Warenwirtschaftssystem kann bei Bedarf selbständig Ware nachbestellen und Hersteller, Spedition und Paketdienst können z. B. über elektronische Dienstschnittstellen beauftragt werden. Der Versand bzw. die Auslieferung der Ware kann zudem durch mobile Technologien wie RFID und mobile Eingabegeräte unterstützt werden. Vorteile einer solchen Automatisierung des Prozessablaufs sind eine höhere Geschwindigkeit der Prozessausführung, geringere Kosten, weniger Fehlerquellen sowie eine bessere Auskunftsfähigkeit der Kooperationspartner untereinander [All05, Gad10].

Eine wichtige Voraussetzung für eine reibungslose, (teil-)automatisierte Zusammenarbeit zwischen verschiedenen Parteien ist ein Mindestmaß an *Interoperabilität*, d. h. das Vorhandensein von gemeinsamen Kommunikationsprotokollen und Datenformaten sowie ein gemeinsames Verständnis über die beteiligten Rollen und die jeweils zu erbringenden Dienstleistungen [All05]. Zudem werden geeignete Verfahren zur *Überwachung der verteilten Prozessausführung* benötigt [CCSD04]. Ein einfaches Beispiel für das in Abbildung 4.2 dargestellte Szenario ist das Bereitstellen von technischen Maßnahmen zur Verfolgung von Lieferungen von Seiten der Transportdienstleister (*Paketverfolgung*), welche sowohl dem Händler und dem Hersteller als auch dem Endkunden Einblick in die Ausführung von (Teil-)Prozessen der betei-

ligten Geschäftspartner erlauben und somit die *Nachvollziehbarkeit* der Prozessausführung im Ganzen erhöhen. Wird dabei eine *Abweichung vom erwarteten Verhalten* festgestellt, so können Anpassungen notwendig werden, welche sich – neben inhaltlichen oder vertraglichen Konsequenzen – auch auf die Verteilung der Prozessausführung auswirken können. Ein Beispiel ist das *Austauschen eines temporär überlasteten Geschäftspartners* durch einen Partner mit freien Kapazitäten.

Durch die Beteiligung verschiedener Unternehmen mit unterschiedlichen Zielen und Arbeitsweisen sowie einer potentiell sehr heterogenen Struktur an (stationären und mobilen) Informations- und Kommunikationssystemen ist die Gestaltung solcher unternehmensübergreifenden Abläufe in der Regel sehr aufwendig und lohnt sich in der Regel nur bei oft in gleicher Weise wiederholten Prozessen (vgl. Abschnitt 2.2). Ein dynamischer Austausch der Kooperationspartner zur Laufzeit setzt zudem voraus, dass alle potentiellen Anbieter der Dienstleistung über eine einheitliche Schnittstelle zur Übernahme eines bestimmten Teilprozesses verfügen. Zum Beispiel wäre in dem dargestellten Szenario die Einbindung verschiedener Transportdienstleister möglich, je nachdem ob der Kunde einen Standard- oder Expressversand vornehmen möchte oder in Abhängigkeit der bestellten Ware diese als Paket-, Sperrgut- oder Gefahrentransport versendet werden muss [Wes07]. Die für die Prozessausführung erforderlichen (speziellen) Schnittstellen sowie die sich im Hintergrund befindlichen Implementierungen müssen (ggf. in verschiedenen Ausführungen für verschiedene Prozesse) durch alle potentiellen Lieferanten bereitgestellt werden, auch wenn es nur selten oder nie zu einem Aufruf durch einen Kooperationspartner kommt.

Die *Autonomie* der beteiligten Geschäftspartner verbietet zudem in den meisten Fällen eine hierarchisch übergeordnete Instanz, welche die Modellierung und technische Umsetzung des unternehmensübergreifenden Gesamtprozesses organisieren oder überwachen kann. Flexibilitätspotentiale werden daher in diesem Bereich verstärkt durch den Einsatz *dezentraler Organisationsformen* gesehen [KB04a]. Eine flexible Änderung der Prozessausführung ist hier jedoch in der Regel nicht möglich, da die Schnittstellen der beteiligten Unternehmen zur Durchführung elektronisch gestützter Prozessabschnitte zunächst mit allen relevanten Kooperationspartnern ausgehandelt und technisch aufwendig umgesetzt werden müssen [All05].

In vielen dynamischen Situationen ist (im Gegensatz zu dem bisher betrachteten eher statischen Szenario) auch oft ein *spontaner Zusammenschluss* von Unternehmen erforderlich. Beispiele sind Situationen, in denen sich Organisationen nur zur Bearbeitung von einzelnen Aufgaben zusammenfinden (z. B. im Rahmen *virtueller Unternehmen* [Koc00, HLGG00]) oder in denen die beteiligten Parteien einer Kooperation noch keine etablierte Geschäftsbeziehung zueinander aufweisen (z. B. in Szenarien des *Open E-Commerce*, vgl. [vdA99]). Aber auch in Ausnahmesituationen und bei Auftreten unerwarteter Ereignisse kann die Nutzung unternehmensfremder Ressourcen kurzfristig Abhilfe schaffen. Bei Kooperationen außerhalb der typischen Unterstützung von

Lieferketten wird dabei in vielen Fällen auch die Abgrenzung zwischen den beteiligten Unternehmen zunehmend variabel. Ein Beispiel sind zwischenbetriebliche Kooperationen auf horizontaler Ebene, bei denen die beteiligten Unternehmen in der Kooperation schnell und flexibel auf die ungenutzten Ressourcen ihrer Kooperationspartner zugreifen können [KB04a], so dass diese eine nur unregelmäßig benötigte Quantität an Ressourcen nicht ständig selbst vorhalten muss. Als Folge müssen Prozesse daher auch oft unvorhergesehen über mehrere Organisationen verteilt werden. Die Abgrenzung von internen Geschäftsprozessen und organisationsübergreifenden Abläufen sowie deren Unterstützung durch Informations- und Kommunikationstechnologien wird daher zunehmend schwieriger.

Bei der Anpassung von Geschäftsprozessen ist insbesondere die Dauer des Anpassungsvorgangs ein wettbewerbsrelevanter Faktor [KB04a]. Ziel ist daher eine kostengünstige Abwicklung von E-Business-Szenarien möglichst ohne Vorbereitung [All05], was in der Regel nur durch eine einheitliche und gemeinsame technische Infrastruktur erreicht werden kann. Neben der Durchsetzung einheitlicher Vorgaben durch einen dominanten Kooperationspartner geht der Trend zur Durchführung unternehmensübergreifender Prozesse deshalb in Richtung *Standardisierung* von Schnittstellen und Prozessbeschreibungssprachen sowie zu einer gemeinsamen Nutzung von elektronischen Marktplätzen als unabhängige Plattformen zur dynamischen Anbindung von potentiellen Kooperationspartnern [All05]. Die gemeinsame Nutzung von Standardsoftware (z. B. zum *Enterprise Resource Planning (ERP)*, *Supply Chain Management (SCM)* sowie zum entsprechenden *Supplier Relationship Management (SRM)*) wird jedoch der Individualität von Unternehmen und Unternehmenskooperationen oft nicht gerecht [BK06, ENS07]. Es gibt daher insbesondere im Bereich des E-Business einen Bedarf an Lösungsmöglichkeiten, welche bei Beibehaltung der Autonomie aller beteiligten Parteien auch eine dynamische Verteilung von Geschäftsprozessen im Rahmen von (spontanen) Unternehmenskooperationen erlauben.

4.2.2 Organisationsübergreifende Prozesse im E-Government

Bei einer organisationsübergreifenden Zusammenarbeit ist oftmals auch die Auswahl eines geeigneten Kooperationspartners und somit die Verteilung des Prozesses selbst Teil des Anwendungsfalls. Die folgenden beiden Anwendungsszenarien aus dem Bereich E-Government zeigen die dynamische Auswahl von Kooperationspartnern am Beispiel zweier langandauernder organisationsübergreifender Prozesse aus dem Kontext des europäischen Forschungsprojekts *Research for E-Government* (*R4eGov*)[1].

[1] R4eGov – Towards e-Administration in the Large. EU 6th Framework Program, www.r4egov.eu, 2006-2009.

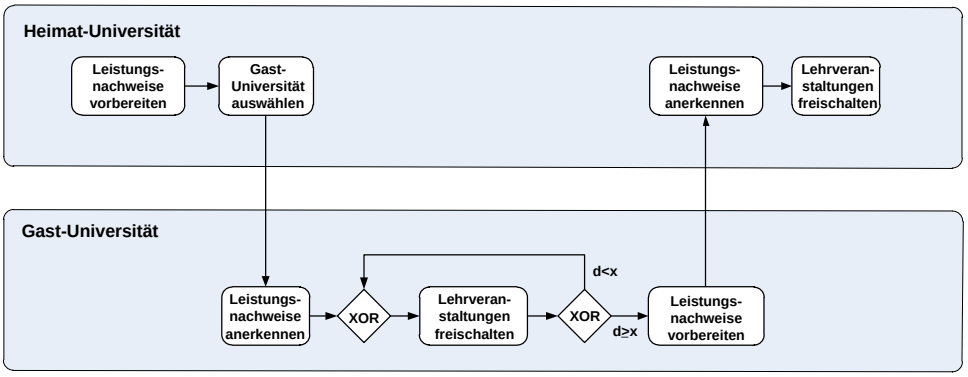


Abbildung 4.3: Beispielprozess zum eErasmus-Projekts (ZBH+10)

Fallstudie: eErasmus eHigher Education (eEH)

In dem untersuchten fachlichen (Teil-)Projekt *eErasmus eHigher Education (eEH)* [LV06] wird eine elektronische Unterstützung für das grenzüberschreitende Austauschprogramm *Erasmus*[2] entwickelt, welches die Vermittlung von Auslandsaufenthalten von Studierenden zwischen Hochschulen innerhalb der Europäischen Union (EU) vermittelt und organisiert. Ziel ist es, den Vorgang des Austausches von Studierenden innerhalb dieses Programms weitestgehend zu vereinheitlichen, so dass auf Basis der bislang erbrachten Studienleistungen eines Studierenden der Zugang zu geeigneten Lehrveranstaltungen an einer Gast-Universität im Ausland freigegeben und die dort erbrachten Studienleistungen bei der Rückkehr des Studierenden an der jeweiligen Heimat-Universität angerechnet werden können [LV06].

Abbildung 4.3 zeigt ein Anwendungsbeispiel aus dem eErasmus-Projekt. Der abgebildete Prozess stellt den (vereinfachten) Vorgang eines standardisierten Ablaufs der Organisation des Auslandsaufenthalts eines Studierenden mit Anrechnung der erbrachten Studienleistungen dar. Zunächst wird hierbei auf Basis der Wünsche des Studierenden und der bereits erbrachten Leistungen eine geeignete Gast-Universität ausgewählt. Nach einer entsprechenden Prüfung wird die Ausführung des Teilprozesses der Freischaltung geeigneter Lehrveranstaltungen sowie die Aufbereitung der dort erbrachten Leistungen an die Gast-Universität übertragen. Nach Beendigung des Auslandsaufenthalts kehrt der Kontrollfluss zur Heimat-Universität zurück, wo die Studienleistungen angerechnet werden können und der Studierende sein Studium mit entsprechend darauf aufbauenden Lehrveranstaltungen fortsetzen kann.

Bei der Betrachtung der Kooperation zwischen Heimat- und Gast-Universität wird deutlich, dass die Auswahl der Gast-Universität dynamisch zur Laufzeit des Prozesses geschieht. Die konkrete Gast-Universität ist demnach zur Entwicklungszeit des Prozesses bzw. vor Beginn dessen Ausführung noch nicht endgültig bekannt. Die Information, welche Universität für den Auslandsaufenthalt ausgewählt wurde, kann in einem solchen Szenario zum

[2]Deutscher Akademischer Austauschdienst e.V. (DAAD): eu.daad.de

Beispiel als Prozessvariable festgehalten werden, welche bei der Verteilung des Prozesses ausgewertet wird (vgl. auch [BD00]). Ähnlich wie eine dynamische Auswahl von Diensten im Umfeld dienstorientierter Architekturen ist in diesem Fall somit auch das Auffinden und Einbinden von Kooperationspartnern zur Ausführung bestimmter Prozessschritte erforderlich. Für die Realisierung der Verteilung müssen daher sowohl die Heimat-Universität als auch die Gast-Universität über geeignete Schnittstellen verfügen, um die Übergabe des Kontrollflusses und eine etwaige Interaktion während der Prozessausführung vornehmen zu können.

Des Weiteren wird bei diesem Anwendungsbeispiel deutlich, dass potentiell jede Instanz des Prozesses bei unterschiedlichen Parteien gestartet oder unterschiedliche Parteien involvieren kann, was die Verwaltung der insgesamt ausgeführten Prozesse erschwert. In der Annahme, dass jeder teilnehmenden Hochschule ein finanzieller Betrag für die Betreuung eines Studierenden gewährt wird, muss zum Beispiel die Dauer der extern durchgeführten Aktivitäten erhoben und (ggf. durch die Heimat-Universität) bestätigt oder verwaltet werden. Hierfür müssen geeignete Mechanismen zum Austausch der gesammelten Daten für die *Nachvollziehbarkeit* der extern ausgeführten Prozessschritte bereitgestellt werden. Ebenso sollten Änderungen an den Daten oder den Umständen der Prozessausführung möglichst an alle Beteiligten propagiert werden können. Wird zum Beispiel aufgrund von Krankheit des Studierenden die Dauer des Auslandsaufenthalts unvorhergesehen verlängert (vgl. Variable d in Abbildung 4.3), so sollte auch die Heimat-Universität darüber informiert werden, um zum Beispiel eine Exmatrikulation wegen fehlender lokaler Rückmeldung des Studierenden zu vermeiden.

Die Partitionierung und die Zuordnung von Prozessschritten, die von einem externen Partner ausgeführt werden sollen, ist in dem betrachteten Anwendungsbeispiel statisch festgelegt und wird sich hierbei in der Regel auch zur Laufzeit einer einzelnen Prozessinstanz nicht ändern. Es existieren jedoch andere Anwendungsfälle, deren Ausführung einer deutlich höheren Dynamik unterliegt und die im Folgenden betrachtet werden sollen.

Fallstudie: Europol/Eurojust

Von notwendigen Anpassungen sind insbesondere oft langandauernde Prozesse betroffen, deren Objekte und deren relevanter Kontext sich während der Prozessausführung verändern können. In der hier betrachteten Fallstudie sollen organisationsübergreifende Prozesse zwischen der europäischen Justizbehörde *Eurojust*, der europäischen Polizeibehörde *Europol* und den jeweiligen nationalen Behörden ausgeführt werden. Die Anbindung von Europol bzw. Eurojust an die nationalen Strafverfolgungsbehörden erfolgt hierbei durch lokale Verbindungseinrichtungen. Dabei steht zum einen die Umwandlung der bisher papierlastigen Kooperation in eine elektronische Interaktion und zum anderen die Unterstützung dieser Interaktion durch eine sichere Datenverbindung im Vordergrund. Das Ziel ist eine schnellere und effektivere Verfolgung

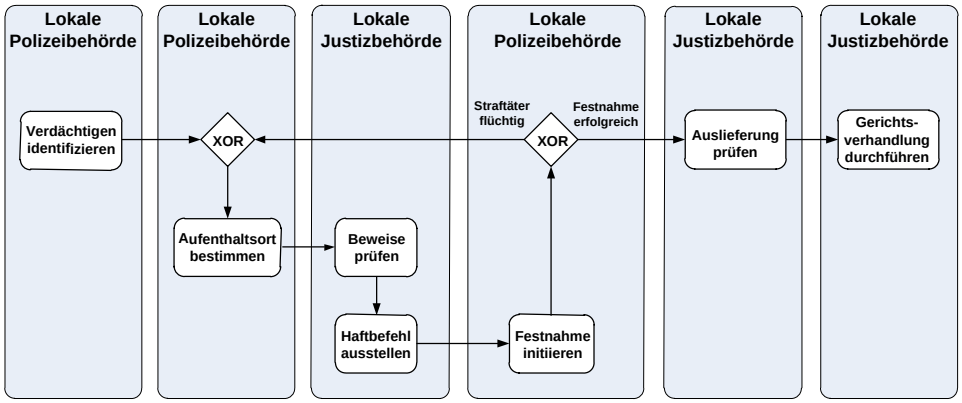


Abbildung 4.4: Beispielprozess zur grenzüberschreitenden Verfolgung von Straftätern

von Straftaten in Europa, zum Beispiel durch gemeinsame Schnittstellen zur Kontaktaufnahme, zum Zugriff auf benötigte Kriminalakten und Formulare, zur Bereitstellung von Beweismaterial oder zum Beantragen bzw. Ausstellen von Haftbefehlen (vgl. [SCB06, WPH07]).

Abbildung 4.4 zeigt einen vereinfachten und verallgemeinerten Prozess für ein erweitertes Kooperationsszenario zur Verfolgung von Straftaten in dem beschriebenen Kontext. Hierbei wird nicht etwa der Vorgang der Strafverfolgung selbst durch die Prozessausführung automatisiert, sondern es wird der ordnungsgemäße Ablauf einer ggf. grenzüberschreitenden Kooperation unter Beteiligung aller relevanter Instanzen und der ordnungsgemäße Austausch von entsprechenden Dokumenten (z. B. Vorlage von Beweismaterial oder Haftbefehlen) explizit festgeschrieben. Der Austausch von Informationen kann hierbei durch Informations- und Kommunikationstechnologien bzw. durch den prozessorientierten Aufruf der entsprechend bereitgestellten Schnittstellen unterstützt werden.

Bei der Betrachtung des Beispiels wird deutlich, dass in diesem Fall die Ausführung des Prozesses nicht durch eine inhaltliche Änderung des Prozessmodells angepasst werden darf. Die inhaltlichen Aktivitäten oder die Reihenfolge der prinzipiell auszuführenden Schritte müssen hier zur Sicherstellung eines ordnungsgemäßen Verfahrens unangetastet bleiben (z. B. darf die Beantragung eines Haftbefehls nicht ausgelassen werden, da sonst eine Verhaftung unzulässig wäre). Stattdessen wird bei diesem Szenario die tatsächliche Ausführung des Prozesses von der Ungewissheit, welche Institution in welchem Staat die Aktivität schließlich ausführen wird, beeinflusst. Welche Behörde in welchem Staat welche Aufgabe übernimmt, hängt jedoch dabei maßgeblich davon ab, in welchem Land der Straftäter ein Verbrechen verübt hat, in welches Land er geflüchtet ist bzw. während der Ausführung des Prozesses weiter flüchtet, in welchem Land er schließlich gefasst wird und ob mit diesem Land ein Auslieferungsabkommen existiert. Die Verteilung des Prozesses muss daher zur Ausführungszeit den jeweiligen Umständen angepasst werden. Zudem ist ggf. nicht im Vornherein klar, ob überhaupt eine

länderübergreifende Kooperation notwendig ist, z. B. wenn der Aufenthaltsort des Straftäters nicht im Ausland festgestellt wird.

In beiden E-Government-Szenarien sind die beteiligten Prozessteilnehmer durch ein hohes Maß an Autonomie und deren technische Infrastrukturen in der Regel durch eine hohe Heterogenität gekennzeichnet. Gleichzeitig ist für die Kooperation die Nachvollziehbarkeit der Prozessausführung durch externe Partner erforderlich, um die Einhaltung gemeinsamer Rahmenbedingungen und Ausführungsrichtlinien zu gewährleisten. Hierbei sind oft dezentrale Lösungen anzustreben, da aufgrund der Autonomie der beteiligten Parteien eine zentrale Kontrolle oft nicht realisierbar ist. Ein wichtiges Beispiel hierfür ist die fehlende Verantwortlichkeit von Prozessteilnehmern für den Gesamtprozess [WPH07].

4.2.3 Integration mobiler Endgeräte

Mobilität und Dynamik gewinnen in unserer Gesellschaft zunehmend an Bedeutung. Mobile Endgeräte, wie Notebooks, Tablet-PCs, persönliche digitale Assistenten (PDAs) und Mobiltelefone sind im Allgemeinen durch eine zunehmende Leistungsfähigkeit gekennzeichnet und bieten durch einen ubiquitären Zugriff auf Daten und durch umfangreiche Funktionalitäten Unterstützung für den mobilen Benutzer. Vor dem Hintergrund dieser Entwicklung werden daher auch die Integration von mobilen Ressourcen in prozessorientierte Anwendungen sowie die entsprechende Bereitstellung von Ausführungskomponenten für mobile Endgeräte zu immer wichtigeren Aspekten des Prozessmanagements.

Die Integration mobiler Endgeräte verspricht eine hohe Flexibilität in Bezug auf den Zugriff mobiler Ressourcen. So können bestimmte Prozesse ganz oder teilweise einem bestimmten Benutzer zur Ausführung zugeordnet werden und der entsprechende Prozess(-teil) dann wahlweise auf einem stationären Arbeitsplatzrechner oder – bei Abwesenheit des Benutzers (z. B. im Rahmen eines Auslandseinsatzes) – auf dessen Mobilgerät ausgeführt werden. Prozesse können somit einerseits schneller bearbeitet werden, da nicht auf eine Rückkehr des Benutzers gewartet werden muss. Andererseits wird eine Prozessausführung am „Ort des Geschehens“ unterstützt und erlaubt dem Benutzer somit eine nahtlose Eingabe von Daten am jeweiligen Einsatzort, ohne dass Informationen zwischenzeitlich auf Papier notiert oder auf anderen Medien zwischengespeichert werden müssen. Die Vermeidung von Medienbrüchen kann wiederum zu einer effizienteren und weniger fehleranfälligen Prozessausführung führen. Schließlich lassen sich auch viele spezielle Anwendungsfunktionalitäten mobiler Geräte nutzen, welche auf stationären Systemen nicht verfügbar sind und somit für die Prozessausführung einen zusätzlichen Mehrwert erbringen. Beispiele sind typische mobilitätsbezogene Dienste wie die Nutzung von *GPS* (*Global Positioning System*), die Auswertung

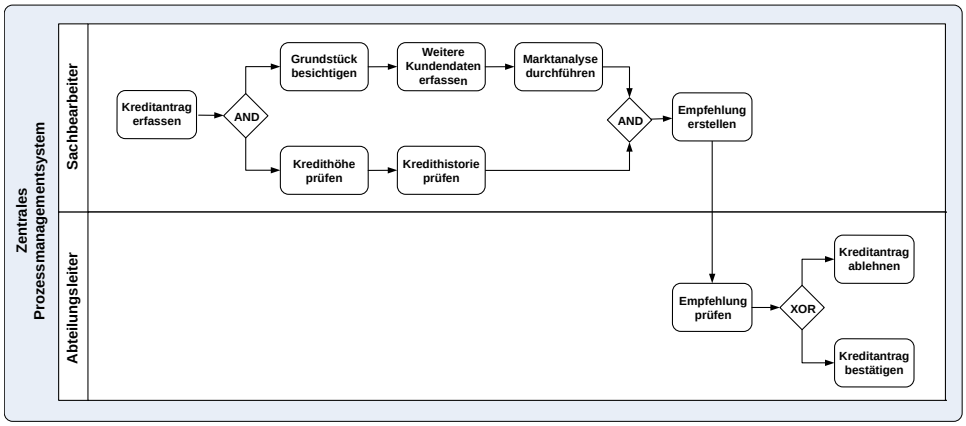
von Richtungs- oder Beschleunigungssensoren oder die Messung von physikalischen Daten wie Temperatur oder Luftfeuchtigkeit [PC07, CP07].

Aufgrund ihrer kleineren Bauweise verfügen mobile Geräte im Gegensatz zu stationären Rechensystemen nur über beschränkte Ressourcen, insbesondere hinsichtlich Rechenleistung, Speicherkapazität und Energieversorgung. Zudem ist ihre Konnektivität zu Kommunikationsnetzwerken aufgrund ihrer Mobilität in der Regel eingeschränkt, d. h., dass mobile Geräte im Allgemeinen nicht dauerhaft mit einem leistungsfähigen Netzwerk verbunden sind. Obwohl sich die Leistung mobiler Geräte und der von ihnen verwendeten Netzwerke in den letzten Jahren stark verbessert hat, ist ihnen dieses Leistungsdefizit im Vergleich zu stationären Geräten *inhärent* [Sat96]. Dies ist vor allem in der ebenfalls stets voranschreitenden Entwicklung stationärer Rechensysteme begründet. Aber auch die zunehmende Miniaturisierung mobiler Geräte (z. B. in den Bereichen RFID, Sensoren oder Chipkarten) führt zu weitergehenden Leistungseinschränkungen, welche auch in Zukunft zu berücksichtigen sein werden [AGIS05].

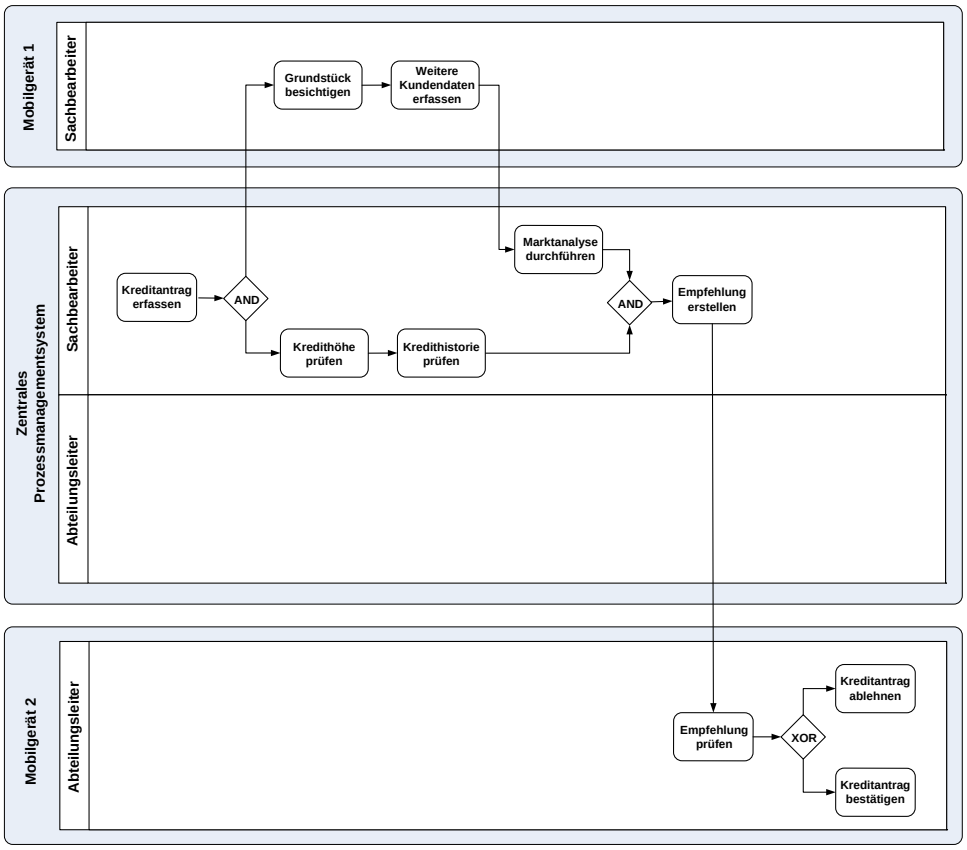
Als Konsequenz dieser Betrachtung ist es oft nicht möglich oder aus Leistungsgründen nicht vorteilhaft, eine Vielzahl von Prozessen bzw. Prozessinstanzen gänzlich auf einem mobilen Gerät auszuführen. In vielen Fällen wird der Kontrollfluss von prozessorientierten Anwendungen zentralisiert auf einem leistungsfähigen *stationären Server* verwaltet und das mobile Gerät bei Bedarf als *mobiler Client* in die Prozessausführung mit einbezogen. Durch die beschränkte Konnektivität mobiler Geräte kann es hierbei jedoch zu einer ungewollten Verzögerung der Prozessausführung kommen, wenn keine Verbindung zwischen dem mobilen Client und dem zentralen Server aufgebaut werden kann. Zudem ist der häufige Datentransfer zwischen Server und Client auch in Hinblick auf die oft beschränkte Bandbreite mobiler Kommunikationsnetzwerke und die zum Teil hohen Nutzungsentgelte (z. B. bei GPRS/UMTS) nicht immer vorteilhaft [MCA+06]. Die Integration von mobilen Geräten führt daher oft zu verteilt ausgeführten Prozessen mit *Offline-Bearbeitung* von ausgewählten Prozesspartitionen auf mobilen Geräten, wobei die Art und Anzahl der mobil auszuführenden Prozessabschnitte nicht immer zur Entwicklungszeit festgelegt werden können. Anforderungen und Herausforderungen solcher Szenarien werden im Folgenden anhand von zwei Fallstudien verdeutlicht.

Fallstudie: Offline-Bearbeitung von Prozessschritten

Konnten bis vor kurzem nur einzelne Aktivitäten auf mobilen Geräten angezeigt und durch den Benutzer über einen *mobilen Workflow Client* bearbeitet werden, während die Navigation des Kontrollflusses von Prozessen auf einem stationären Server verblieb [AGK+95, AGK+96], so verfügen selbst Mobiltelefone heute schon über weitestgehend ausreichende Ressourcen (d. h. Speicherkapazität und Rechenleistung), um einfache Prozesse oder Prozessabschnitte auf einer *mobilen Prozess-Engine* relativ eigenständig ausführen zu können [HHGR06, PC07, Kun08].



(a) Zentral ausgeführter Prozess



(b) Mögliche Auswahl von mobil ausgeführten Prozessabschnitten

Abbildung 4.5: Beispielprozess zur Integration von mobilen Geräten (vgl. tlw. (AGK+95))

Ein typisches Szenario ist die benutzergesteuerte Auswahl von Prozessschritten, welche mobil bearbeitet werden sollen, die Übertragung dieser Prozessschritte (und des entsprechenden Kontrollflusses zwischen diesen Schritten) auf ein mobiles Gerät, die autonome Bearbeitung dieser Aktivitäten und die abschließende Synchronisation der lokalen Prozessausführung mit dem zentralen Prozessmanagementsystem [PC07]. Abbildung 4.5a zeigt einen imaginären Beispielprozess aus dem Kontext eines Finanzdienstleisters, welcher mit Hypotheken als Sicherungsmittel Kredite an Grundstückseigentümer vergibt (vgl. z.T. [AGK+95]). Da hierbei unter Umständen Daten direkt vor Ort beim Kunden gesammelt werden müssen (z. B. im Rahmen der Aktivitäten *Grundstück besichtigen* und *Weitere Kundendaten erfassen*) kann die Prozessausführung dieser Schritte auf ein mobiles Gerät übertragen werden und so notfalls auch offline (z. B. bei einer fehlenden Netzwerkverbindung in ländlicher Gegend) erledigt werden (vgl. Abbildung 4.5b). Zusätzlich können die zur Prüfung des Kreditantrages auszuführenden Aufgaben *Kredithöhe prüfen* und *Kredithistorie prüfen* (ggf. durch einen weiteren Sachbearbeiter) parallel auf einem stationären System ausgeführt werden. Im Anschluss ist in jedem Fall eine Synchronisation der Prozessdaten erforderlich, um ein korrektes und einheitliches Ergebnis zu erlangen, auf dessen Basis die *Empfehlung zur Kreditvergabe* erstellt werden kann. Aufgrund der inhaltlichen Gegebenheiten der betreffenden Aktivitäten (Außendiensttätigkeiten und rechercheaufwändige Innendiensttätigkeiten) könnte in diesem Fall bereits bei der Modellierung des Anwendungsfalls auf eine potentielle Verteilung des Prozesses geschlossen werden. In anderen Fällen ist dies jedoch nicht möglich. Ein Beispiel ist die mögliche Abwesenheit des Abteilungsleiters, welcher die Kreditvergabe im Anschluss an die beschriebenen Aktivitäten bestätigen soll. Ist der Abteilungsleiter anwesend, würde er die anliegenden Aufgaben (*Empfehlung prüfen*, *Kreditantrag ablehnen* oder *Kreditantrag bestätigen*) aufgrund der größeren Bildschirmauflösung in der Regel eher auf seinem Arbeitsplatzrechner erledigen. Bei Abwesenheit (z. B. einer längeren Dienstreise) empfiehlt sich jedoch eine mobile Ausführung der Aufgaben, um die Bearbeitung des Kundenauftrags nicht weiter zu verzögern (vgl. Abbildung 4.5b).

Neben der Berücksichtigung der eingeschränkten Leistungsfähigkeit mobiler Geräte sowie der Vermeidung von Verzögerungen durch Verbindungsprobleme ist bei der Integration von mobilen Systemen auch eine kontextabhängige Verteilung von Prozessteilen zur Laufzeit des Prozesses erforderlich. Dabei können sowohl der Inhalt des Prozesses, die Beschaffenheit mobiler Geräte (z. B. deren Benutzungsschnittstellen) als auch die individuellen Anforderungen von menschlichen Prozessteilnehmern eine Rolle spielen. Die Art und Weise der Abbildung von durchzuführenden Interaktionen mit dem Prozessmanagementsystem wird im Folgenden genauer an einem weiteren Beispiel erläutert.

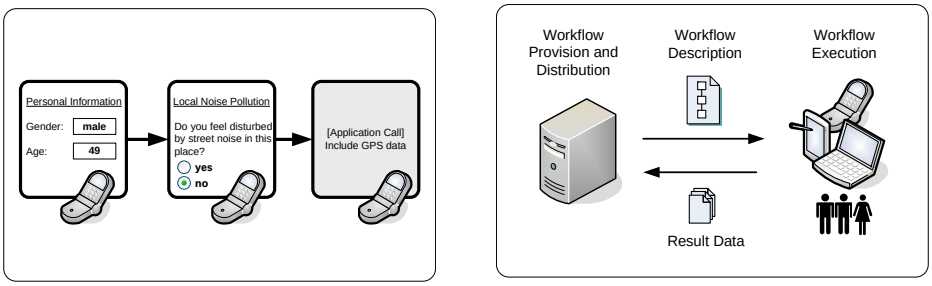
Fallstudie: Elektronischer Fragebogen

Bei einer vollständigen oder teilweisen Ausführung von Prozessen auf mobilen Geräten spielt insbesondere die Integration des Benutzers als Prozessteilnehmer eine große Rolle. Dies hat den Hintergrund, dass die Ausführung von (komplexen) automatisierten Aktivitäten aufgrund der eingeschränkten Ressourcen mobiler Geräte oft eher den leistungsfähigeren stationären Geräten vorbehalten ist. Mobil ausgeführte Prozesse bzw. Prozesspartitionen weisen daher in der Regel einen vergleichsweisen hohen Anteil an Aktivitäten mit Benutzerinteraktionen auf (vgl. [ZK07, CP07]).

Abbildung 4.6 zeigt ein Beispiel aus einem von der Universität Hamburg und einem Wirtschaftspartner durchgeführten Projekt zur Abbildung und Ausführung benutzerzentrischer Prozesse. Ziel war es, die bisher papierlastige Durchführung von Befragungen durch elektronische Fragebögen zu unterstützen, welche auf mobilen Geräten angezeigt und in Abhängigkeit der individuellen Eingabedaten der befragten Personen weitergeschaltet werden können (vgl. auch [ZBV09]). Eine verkürzte Darstellung eines solchen Fragebogens ist in Abbildung 4.6a dargestellt. Er enthält eine kurze Umfrage zur Geräuschbelästigung als Interaktion durch graphische Eingabemasken und eine automatisierte Erfassung des jeweiligen Aufenthaltsortes durch den Aufruf einer Anwendung zur Positionsbestimmung (GPS).

Zur Durchführung der Befragung können die elektronischen Fragebögen an geeignete Personen versendet oder im einfachsten Fall aus einem zentralen Repository auf die mobilen Geräte der Benutzer heruntergeladen und nach Abschluss der Befragung die entsprechenden Ergebnisdaten zurückgeführt werden (vgl. Abbildung 4.6b). Um repräsentative Ergebnisse einer Befragung zu erhalten (z. B. für Marktforschungszwecke) muss dabei eine größere Anzahl von Fragebögen verteilt und ausgefüllt werden. Aus Zeitgründen geschieht die Verteilung der elektronischen Fragebögen und damit auch die Ausführung der entsprechenden Prozessinstanzen in hohem Maße parallelisiert. Die Anschaffung und (ggf. leihweise) Vergabe von spezialisierten Geräten zur Durchführung der Befragungen ist aufgrund der großen Teilnehmerzahl, der anzustrebenden Parallelisierung und der wechselnden Zielgruppen von Befragungen jedoch in den meisten Fällen nicht wirtschaftlich. Als Konsequenz ist die Nutzung von privaten Mobilgeräten der zu befragenden Zielgruppe, insbesondere deren Mobiltelefone, in Betracht zu ziehen.

Wird eine Aktivität eines Prozesses durch die Interaktion mit dem Benutzer ausgeführt, so muss dem Benutzer in der Regel die auszuführende Aufgabe präsentiert werden, es müssen ihm entsprechende Daten zur Bearbeitung dieser Aufgabe mitgeteilt werden (Ausgabedaten) und ggf. müssen die Ergebnisse der Aufgabenbearbeitung vom Prozessmanagementsystem entgegengenommen werden (Eingabedaten) (vgl. Abschnitte 2.7.3 und 3.5.5). Bei einer nichtverteilten Ausführung von prozessorientierten Anwendungen ist die Darstellung der hierfür benötigten Benutzungsschnittstellen durch die lokal verwendeten (in der Regel homogenen) Hard- und Softwaresysteme determiniert. Die



(a) Beispielprozess mit Benutzerinteraktionen und Aufruf lokaler Anwendungen

(b) Verteilung von Prozessen an verschiedene (mobile) Endgeräte

Abbildung 4.6: Prozessorientierte Benutzerinteraktionen für verschiedene heterogene Endgeräte (vgl. (ZBV09))

Benutzungsschnittstellen können daher entweder in der Prozessbeschreibung selbst oder durch entsprechende zusätzliche (graphische) Komponenten (z. B. HTML-Formulare) in konkreter Form beschrieben und im Fall einer Interaktion direkt zur Anzeige gebracht werden. Bei einer Verteilung von Prozessen an Geräte mit sehr *heterogenen Benutzungsschnittstellen* – wie dies im Fall von mobilen Geräten häufig der Fall ist [MPS04, CELS07] – ist der Umgang mit der Auswahl und der Darstellung einer geeigneten Interaktionsmöglichkeit für den individuellen Benutzer und sein (mobiles) Endgerät jedoch eine Herausforderung. Die zum elektronischen Fragebogen durchgeführte Studie hat gezeigt, dass alleine bei den sich derzeit auf dem Markt befindlichen Mobiltelefonen die Heterogenität der Benutzungsschnittstellen so groß ist, dass selbst für einfache Benutzerinteraktionen keine gemeinsame Darstellung erreicht werden kann [ZBV09]. Des Weiteren sind noch viele ältere Varianten von Mobiltelefonen im Einsatz, welche zum Beispiel nicht über einen (geeigneten) Internet-Browser verfügen, um die Interaktion über webbasierte Schnittstellen abbilden zu können. Hinzu kommt eine große Menge an anderen mobilen Geräten (z.B PDAs, Netbooks, Notebooks oder Tablet-PCs) welche über teilweise völlig andere Eigenschaften und Möglichkeiten zur Benutzerinteraktion verfügen. Kontextabhängige Benutzerinteraktionen (wie Möglichkeiten zur Sprachausgabe im Fall eingeschränkter visueller Wahrnehmung) bleiben hierbei zudem noch völlig unberücksichtigt.

Aufgrund der vielfältigen Darstellungsmöglichkeiten ist eine vorausschauende Anpassung des elektronischen Fragebogens (bzw. des zugrunde liegenden Prozessmodells) auf alle möglichen Endgeräte nicht vorteilhaft. Die Flexibilisierung der Prozessausführung durch die explizite Modellierung möglichst vieler Varianten (vgl. Abschnitte 3.3.1 und 3.5.5) ist daher kaum realisierbar. Betrachtet man die Instanzen der elektronischen Fragebögen als verteilte Subprozesse eines übergeordneten (Marktforschungs-)Prozesses, so wird klar, dass die dynamische Verteilung von (Teil-)Prozessen an im Voraus nicht bekannte (mobile) Endgeräte im Allgemeinen besonderen Anforderungen an die Unterstützung von Benutzerinteraktionen unterliegt.

4.2.4 Kontextbasierte Kooperation

Durch die Verbreitung mobiler Geräte und die rasch voranschreitende Entwicklung im Bereich der Mobilkommunikation sind mittlerweile eine Vielzahl von heterogenen Technologien und Anwendungen entstanden, welche Daten und Funktionalitäten für einen bestimmten Anwendungsbereich bereitstellen. Beispiele reichen dabei von eher allgemein nutzbaren Funktionen, wie der Bestimmung der aktuellen Position via GPS oder dem Versenden von Kurznachrichten (SMS), bis hin zu stark spezialisierten Hard- und Softwaresystemen, wie zum Beispiel zum Erkennen von Waldbränden oder zur Beobachtung von Körperfunktionen gesundheitsgefährdeter Personen [ZKL09b]. Viele dieser Funktionalitäten sind oft nur in einem bestimmten geographischen Bereich oder in spezialisierten (mobilen) Netzwerken verfügbar, welche in der Regel einer hohen Dynamik unterliegen (z. B. *mobile Ad-hoc Netzwerke*, vgl. [RR02, CCL03]). Durch die oft nur sporadische bzw. lokale Verfügbarkeit und die geringere Leistungsfähigkeit der Geräte in Hinblick auf Rechengeschwindigkeit und Speicherkapazität, welche oft in erster Linie zur Bereitstellung der spezialisierten Funktionen aufgewendet werden, ist eine spontane Integration mobiler Systeme zur Bearbeitung einer übergeordneten Aufgabe schwierig. Die verteilte Ausführung komplexerer Aufgaben auf Systemen dieser Art ist daher ohne einen geeigneten Kooperationsmechanismus in der Regel nicht möglich (vgl. [Kun05, Kun08]).

Das Konzept der *kontextbasierten Kooperation* beruht auf der Annahme, dass voneinander unabhängige kontextbezogene Systeme ihre (ggf. spezialisierten) Funktionalitäten anderen Systemen zur Nutzung anbieten und so ihrerseits bei Bedarf von den Funktionalitäten anderer Systeme Gebrauch machen können. Ziel ist es, durch Kooperation die zuvor beschriebene technologische Isolation mobiler Geräte zu relativieren und das Potenzial des gesamten (mobilen) verteilten Systems zur Erledigung komplexerer Aufgaben verfügbar zu machen [Kun08]. Eine komplexe Aufgabe bezeichnet dabei eine Anwendungsfunktionalität, die bei einer individuellen Ausführung durch ein (mobiles) System an dessen begrenzten Ressourcen und Funktionalitäten scheitern würde oder erst durch die kollektive Ausführung einen akzeptablen oder höherwertigen Grad der Zielerfüllung erreicht. Dabei bleiben die beteiligten Systeme bei der Ausführung dieser Aufgabe autonom und besitzen in der Regel keine übergeordnete zentrale Instanz, welche die Kooperation organisiert oder verwaltet [Kun08].

Zur strukturierten Beschreibung der gemeinsam auszuführenden Aufgabe wird eine technologieunabhängige Spezifikation von Teilaufgaben, ihrer Reihenfolge und ihrer auszutauschenden Daten als direkt ausführbare Prozessbeschreibung vorgeschlagen, wobei deren Ausführung im jeweiligen Kontext an ein geeignetes (mobiles) System delegiert wird [Kun08]. In Abhängigkeit der spezifizierten Teilaufgaben (d. h. der Aktivitäten des Prozesses) wird dabei jeweils ein in der mobilen Umgebung verfügbares System zur Ausführung gewählt, welches die aktuell benötigte Funktionalität zur Ausführung der an-

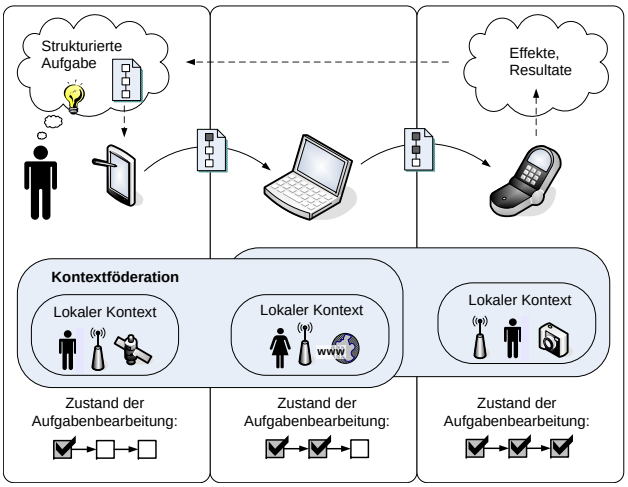
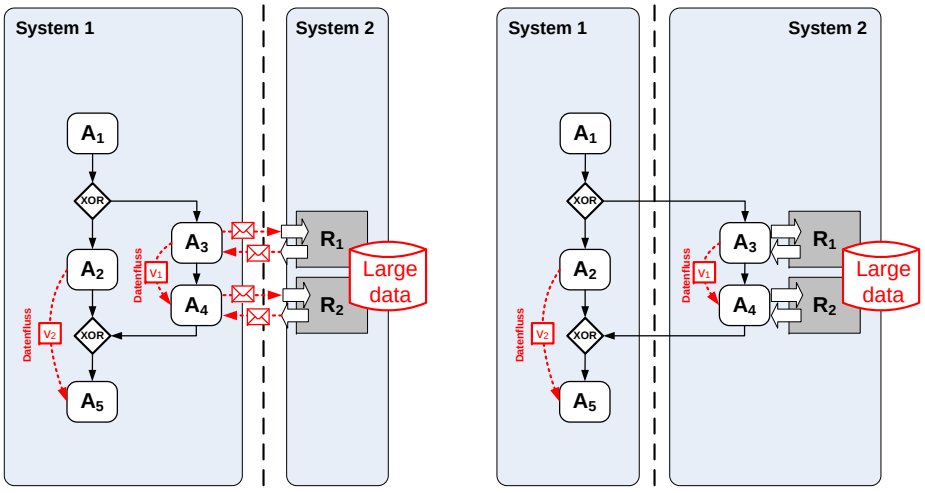


Abbildung 4.7: Kontextbasierte Kooperation (KZTL08, ZKL09b)

liegenden Teilaufgabe anbietet. Bei der grundsätzlichen Eignung mehrerer Systeme können dabei zusätzlich auch nicht-funktionale Aspekte berücksichtigt werden (vgl. [Kun05, Kun08]).

Abbildung 4.7 zeigt die prinzipielle Umsetzung einer kontextbasierten Kooperation durch einen sogenannten *mobilen Prozess*. Hierbei handelt es sich um eine Prozessbeschreibung mit den oben beschriebenen Eigenschaften, welche durch den Konsumenten einer komplexen Funktionalität modelliert und bei Bedarf instantiiert wird. Liegt zur Laufzeit eine für die aktuell auszuführende Aktivität des Prozesses benötigte Funktionalität auf dem aktuell prozessausführenden Gerät oder in dessen unmittelbarer (lokaler) Umgebung nicht vor, so kann der Prozess an ein anderes Gerät transferiert werden, um die sich ggf. dort neu eröffnenden Ausführungskontexte nutzbar zu machen. Wie gezeigt wurde, kann durch die Strategie der kontextbasierten Kooperation mit mobilen Prozessen die Wahrscheinlichkeit der erfolgreichen Ausführung einer komplexen Aufgabe in mobilen Umgebungen prinzipiell erhöht werden (vgl. [KZTL08, Kun08]).

Die verteilte Ausführung von mobilen Prozessen im Rahmen der kontextbasierten Kooperation hat aufgrund der Dynamik mobiler Systeme sehr hohe Anforderungen an eine flexible Verteilung des Prozesses. Zum einen ist weder zur Entwicklungszeit noch zum Zeitpunkt der Instantiierung eines mobilen Prozesses bekannt, welche (mobilen) Systeme tatsächlich zur Laufzeit des Prozesses an dem jeweiligen Ort der Prozessausführung verfügbar sind, welche Funktionalitäten sie bereitstellen und welche Technologien zur Kommunikation oder zum Aufruf von Anwendungsfunktionalität sie unterstützen. Zum anderen sind auch den potentiell an der Prozessausführung beteiligten Systemen die tatsächlich auszuführenden Prozesse und die von ihnen hierfür zu erbringende Art und Anzahl von Anwendungsfunktionalitäten im Voraus nicht bekannt. Sowohl die Partitionierung des Prozesses als auch die Zuordnung der resultierenden Prozesspartitionen zu (mobilen oder stationären)



(a) Zentralisierte Prozessausführung **(b)** Verteilte Prozessausführung

Abbildung 4.8: Verteilung zur Vermeidung der Übertragung großer Datenmengen (Beispiel) (WML09b, Wut10)

Ausführungseinheiten müssen daher kontextabhängig zur Laufzeit bestimmt werden und sind nicht vorab festlegbar.

4.2.5 Optimierung der Prozessausführung

Neben den genannten funktionalen Hintergründen für eine verteilte Prozessausführung spielen in vielen Fällen auch dynamische Leistungs- oder Qualitätseigenschaften der Prozessausführung eine große Rolle. In einem von den Autoren WUTKE, MARTIN und LEYMANN vorgestellten Anwendungsbeispiel [WML09a, WML09b, Wut10] (ähnlich auch bei [CRW98]) wird dabei der Datenfluss einer Dienstkomposition in Bezug auf die zu übertragende Datenmenge untersucht. Abbildung 4.8a zeigt ein entsprechendes Szenario, wobei ein durch *System 1* repräsentierter Anbieter eines (komplexen) Dienstes die Kontrolle über die Ausführung des dazugehörigen Prozesses besitzt. Abhängig von verschiedenen Faktoren (wie z. B. Eingabedaten) kann der Verlauf der hiervon initiierten Prozessinstanzen entweder die Ausführung der Aktivitäten A_1,A_2,A_5 oder A_1,A_3,A_4,A_5 umfassen, wobei typischerweise unterschiedliche Ressourcen in Form von elektronischen Diensten aufgerufen, unterschiedliche Daten abgerufen und im Anschluss weiterverarbeitet werden. In dem dargestellten Szenario werden die Prozessdaten je nach Prozessverlauf in den Variablen v_1 oder v_2 gespeichert. Diese Variablen werden dabei zunächst mit den Ausgabedaten eines elektronischen Dienstes belegt, welche im jeweils nächsten Schritt einem anderen externen Dienst als Eingabedaten übergeben werden. Konkretes Beispiel hierfür könnte der Abruf einer Liste von Dokumenten sein, welche einem Folgebearbeiter zur Durchsicht vorgelegt werden.

Werden mehrere Aktivitäten in Folge von den Ressourcen eines externen Systems bearbeitet (wie hier im Beispiel *System 2*), so kann es unter Umständen

effizienter sein, die Prozessausführung ganz oder teilweise auf dieses System zu verlagern. Insbesondere bei der Übertragung großer Datenmengen in Zusammenhang mit Kommunikationsverbindungen mit nur geringer Bandbreite oder hohen Übertragungskosten ist der Transfer von durchzuführenden Operationen zum Ort der Daten (*Code Shipping*) anstelle der (mehrfachen) Übertragung der Daten zum Ausführungsort der Operationen (*Data Shipping*) eine gängige Vorgehensweise bei der Anfrageverarbeitung für Datenbanken oder bei mobilen Softwareagenten [Bow01, Bra03] (vgl. auch *Code-Migration* in [TS08]). Abbildung 4.8b zeigt eine verteilte Variante der Prozessausführung, bei welcher die Anzahl der über das Netzwerk ausgeführten Interaktionen durch die Aufteilung des Prozesses minimiert und somit die unnötige Übertragung großer Datenmengen verhindert wird (vgl. [WML09b]).

Die Erhebung von geeigneten Daten als Entscheidungsgrundlage für die Verteilung der Prozessausführung birgt auch bei diesem Fallbeispiel große Herausforderungen. Wie in dem hier gezeigten Szenario ist es vorstellbar, dass der tatsächliche Verlauf der Ausführung einzelner Prozessinstanzen im Voraus nicht bekannt ist. Zudem ist die Menge und der Umfang der zu übertragenen Daten ggf. bei der Instantiierung des Prozesses noch nicht abzusehen. Im Fall einer dynamischen Auswahl von Dienstinstanzen zur Laufzeit des Prozesses kann außerdem nur schwer vorherbestimmt werden, ob Folgeaktivitäten auf demselben (Hardware-)System ausgeführt werden. Vor dem Hintergrund einer dienstorientierten Architektur wird eine solche Optimierung der Prozessausführung noch zusätzlich durch die Kapselung der Ressourcen als einzelne Dienste und die daraus entstehende Verteilungstransparenz erschwert.

4.2.6 Process-Management-as-a-Service

Ein weiteres aktuelles Thema stellt momentan das Nutzen und Anbieten von abstrakten hardware- oder softwaretechnischen Funktionalitäten im Rahmen des *Cloud Computing* dar. Hierbei handelt es sich um verteilte Anwendungen und Dienste innerhalb eines Netzwerks, welche in der Regel durch Internetprotokolle zugegriffen werden können und auf virtualisierten Ressourcen basieren [Sos11]. Die zur Verfügung gestellten bzw. konsumierten Infrastrukturen können dabei von physikalischen Ressourcen wie Rechen- oder Speicherkapazität über Entwicklungs- und Ausführungsumgebungen bis hin zu fertigen spezialisierten Softwarefunktionalitäten reichen und in der Regel dynamisch an den Bedarf von Anbietern und Konsumenten angepasst werden. Der Vorteil für den Anbieter solcher Ressourcen besteht dabei darin, dass nur in unregelmäßigen Abständen benötigte Ressourcen in Zeiten geringerer Nutzung öffentlich bereitgestellt werden und somit (in der Regel gegen eine Nutzungsgebühr) ausgelastet werden können. Für Konsumenten der Ressourcen besteht gleichzeitig die Möglichkeit, flexibel über Hard- und Software verfügen zu können, deren individuelle Bereitstellung aufgrund von hohen Entwicklungs-, Anschaffungs- und/oder Wartungskosten für den vorgesehenen (seltenen) Einsatz nicht wirtschaftlich wäre. Vor dem Hintergrund, dass die Erstellung und

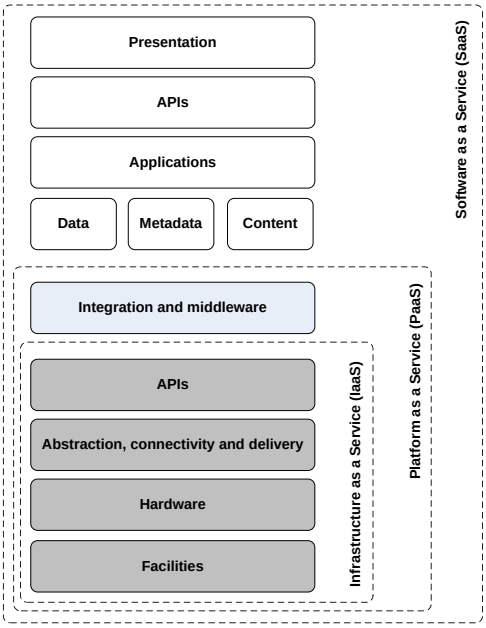


Abbildung 4.9: Cloud Computing Referenzmodell und Schichtenarchitektur (vereinfacht nach (Sos11))

Verwaltung der Ressourcen durch deren Anbieter geschieht und der damit verbundene Aufwand für den Konsumenten transparent ist, spricht man von einer *als Dienst* (*as-a-Service*) gekapselten Funktionalität, deren Bearbeitung aus Sicht des Konsumenten in einer ihm weitestgehend verborgenen „Wolke“ (*Cloud*) geschieht [BKNT09, Fin09, Sos11].

Das Anbieten und Nutzen von hardware- oder softwaretechnischen Funktionalitäten kann auf verschiedenen Ebenen geschehen. Abbildung 4.9 zeigt eine hierarchische Einordnung, wobei allen Dienstleistungen zunächst eine Bereitstellung von Hardware, hardwarenahen Diensten und Basisfunktionalitäten zur Nutzung dieser technischen Infrastruktur zugrunde liegt (*Infrastructure-as-a-Service*, kurz *IaaS*). Darauf aufbauend können höherwertige Dienste in Form von Entwicklungs- und Ausführungsumgebungen zur Verfügung gestellt werden, auf denen Anwendungen aufsetzen können (*Plattform-as-a-Service*, kurz *PaaS*). Diese Schicht ist in der Regel komponentenorientiert und basiert z. B. auf einer dienstorientierten Architektur (vgl. Abschnitt 3.4.1). Die übergeordnete Anwendungsschicht repräsentiert üblicherweise über das Internet angebotene Softwareanwendungen (*Software-as-a-Service*, kurz *SaaS*) [TBB03, Sos11].

Diesem Schichtenmodell entsprechend ist auf den verschiedenen Ebenen auch eine umfangreiche Unterstützung für das Management und die Ausführung von prozessorientierten Anwendungen möglich [Sos11]. Insbesondere für kleinere Unternehmen oder Organisationen ist die Anschaffung und der Betrieb eines komplexen Prozessmanagementsystems mit integrierter Prozess-Engine oft nur mit unverhältnismäßig großem Aufwand möglich.

Die entsprechenden Kosten für Lizenzen, Wartung und Support sowie zur Qualifizierung der eigenen Mitarbeiter rechnen sich in der Regel nur ab einer bestimmten Anzahl von tatsächlich auszuführenden Prozessen. Bei der Entscheidung für *Make, Buy or Rent* ist daher insbesondere das *Software-as-a-Service*-Lizenzmodell für die Bereitstellung von Diensten zur Prozessautomatisierung interessant. Abhängig davon, ob das Prozessmanagement als Funktionalität oder eine einzelne prozessorientierte Anwendung als Dienst verfügbar gemacht wird, wird daher in diesem Kontext von *(Business-)Process-Management-as-a-Service* bzw. von *Process-as-a-Service* gesprochen [FG08].

Bei einem typischen Process-Management-as-a-Service-Modell werden die Hardware, die Prozess-Engine und die entsprechend assoziierten Software-Werkzeuge zur Verwaltung der Prozesse durch einen externen Anbieter über das Internet bereitgestellt, so dass dessen Dienstleistung zur Prozessautomatisierung gleichzeitig von einer Vielzahl von Kunden genutzt werden kann. Die Interaktion mit menschlichen Prozessteilnehmern kann dabei über Standardsoftware wie z. B. Webbrowser, E-Mail-Programme oder SMS realisiert werden. Zudem können benutzerfreundliche Oberflächen zur Prozessanpassung (z. B. über *fachliche Regeln*, vgl. Abschnitt 3.6), zur Dokumentation und zum Monitoring angeboten werden. Im Rahmen der Anwendungsintegration können bei Bedarf sogar Legacy-Anwendungen des Kunden über leichtgewichtige Software-Adapter integriert werden [FG08].

Im Gegensatz zur Entwicklung eigener Prozessmanagementsysteme oder dem Erwerb und Betrieb einer Standard- oder Open-Source-Software zum Prozessmanagement sind die Investitionskosten bei der Nutzung von Process-(Management)-as-a-Service-Angeboten gering. Die Abrechnung kann zum Beispiel über eine monatliche Kostenpauschale und/oder über instanzbasierte Kostenmodelle geschehen. Der große Vorteil liegt jedoch in der schnellen Umsetzbar- und Ausführbarkeit prozessorientierter Anwendungen, was einen großen Flexibilitätsgewinn darstellt. Auf der anderen Seite ist jedoch ein gewisses Maß an Vertrauen zum Dienstanbieter notwendig, insbesondere wenn es um die Ausführung von Prozessen mit sensiblem Inhalt und vertraulichen Daten geht [FG08].

Das Prozessmanagement und die Ausführung von Prozessen „als Dienst“ vereinen viele der oben genannten Anforderungen an eine flexible verteilte Ausführung. Hierbei stehen vor allem Qualitätseigenschaften der Prozessausführung (wie zum Beispiel die Verarbeitungsgeschwindigkeit) und Kostenaspekte (z. B. die Menge der vorzuhaltenden Hardware) im Vordergrund. Die effiziente Verteilung der Prozessausführung auf verschiedene Prozess-Engines ist dabei (ggf. sogar zur Laufzeit des Prozesses) ein wichtiger Mechanismus zum Lastausgleich. Des Weiteren werden geeignete Schnittstellen benötigt, um Dienstkonsumenten mit relevanten Informationen und Funktionen zur Steuerung ihrer Prozesse zu versorgen. Insbesondere ist hierbei die Einigung auf eine gemeinsame Prozessbeschreibungssprache erforderlich. Das noch junge Modell einer Prozessautomatisierung als Dienstleistung wird daher

auch durch die zunehmende Standardisierung von Prozessbeschreibungssprachen weiter vorangetrieben.

4.3 Klassifizierung verteilt ausgeführter Prozesse

Die durch die aufgezeigten Fallbeispiele identifizierten Eigenschaften und Herausforderungen verteilt ausgeführter Prozesse sollen im Folgenden mit den Ergebnissen einer Literaturrecherche zu verschiedenen Klassifikationen verteilter Prozesse zusammengeführt werden. Hierzu werden zunächst die Gründe für eine Verteilung als Auslöser für eine Anpassung der Prozessausführung betrachtet und eingeordnet. Ihnen kommt eine besondere Relevanz zu, da sie der Art und Weise der durchzuführenden Änderung zugrunde liegen. Darauf aufbauend werden im Anschluss die resultierenden Charakeristika, Varianten und Eigenschaften verteilter Prozesse als Grundlage für eine Betrachtung der Anforderungen im Allgemeinen und im speziellen Hinblick auf die Flexibilisierung prozessorientierter Anwendungen kategorisiert. Abschließend werden klassische Verteilungsmodelle für die verteilte Ausführung von Prozessen herausgearbeitet und in Hinblick auf ihre Auswirkungen in Bezug auf Flexibilität untersucht.

4.3.1 Ziele der Verteilung

Wie schon bei der Betrachtung der oben genannten Fallbeispiele auffällt, liegen einer Verteilung der Prozessausführung oft ganz unterschiedliche Anlässe zugrunde. Dabei verfolgt jeder Vorgang der Verteilung ein bestimmtes Ziel [Sch01], welches in der Regel ohne die Verteilung nicht zu erreichen wäre (vgl. auch Abschnitt 4.1.1). Ein Ziel kann dabei auch mehrere Teilziele oder Rahmenbedingungen umfassen, welche teilweise auch gegenläufig sein können. In Bezug auf die Verteilung lassen sich nicht-funktionale (d. h. anwendungsunabhängige) und funktionale (d.h anwendungsbezogene) Ziele identifizieren [Sch01], wobei diese aber zumeist technisch (z. B. in Bezug auf die Verfügbarkeit einer Ressource) und nicht inhaltlich (d. h. in Bezug auf eine inhaltlich andere Ausführung) interpretiert werden können. Die folgende Klassifikation fasst die auf der Basis von wissenschaftlicher Literatur und praktischen Beobachtungen identifizierten möglichen Ziele von verteilt ausgeführten Prozessen zusammen:

- **Kooperation:** Ein wesentliches Ziel verteilter Prozessausführung ist die Abbildung der Zusammenarbeit zwischen verschiedenen Organisationen oder Organisationseinheiten, wobei die Autonomie der einzelnen Parteien durch die Verantwortlichkeit für bestimmte Teilprozesse sowohl durch die Verwaltung des Kontrollflusses als auch durch die Auswahl und den Aufruf von geeigneten Ressourcen erhalten bleiben soll [All05, Wes07, WPH07]. Beispiele sind die Abbildung einfacher Marktbeziehungen (z. B. im Rahmen des *Supply Chain Managements*, vgl.

Abschnitt 4.2.1) oder die technische Realisierung von *(Business) Process Outsourcing* [Rie03], wobei einzelne (Geschäfts-)Prozesse an einen externen Dienstleister ausgelagert werden können. Eine Kooperation kann jedoch nicht nur wirtschaftliche Anwendungsfälle umfassen (vgl. Abschnitt 4.2.1), sondern ebenso auf einer organisatorischen oder technischen Ebene die Zusammenarbeit von verschiedenen Individuen beschreiben, zum Beispiel im Rahmen des E-Governments oder der Komposition mobiler Dienste zur Realisierung komplexer Anwendungsfunktionalitäten [HSH+06, Kun08] (vgl. Abschnitte 4.2.2 und 4.2.4).

- **Zugriff auf nur lokal verfügbare Ressourcen oder Funktionalitäten:** Bestimmte Ausführungseinheiten können exklusiven Zugriff auf nicht-öffentliche Software-Anwendungen, menschliche Prozessteilnehmer, Maschinen oder Informationen haben, welche aus organisatorischen Gründen nur innerhalb eines lokalen Netzwerks verfügbar sind und nicht von einer externen Ausführungseinheit zugänglich gemacht werden können oder sollen. Ein mögliches Beispiel ist ein Drucker (bzw. ein Druckdienst), welcher im Intranet eines Unternehmens frei zur Verfügung steht, jedoch aus dem Internet nicht zugegriffen werden kann. Durch die (teilweise) Ausführung eines Prozesses durch eine Prozess-Engine innerhalb dieses abgeschlossenen Teilsystems können somit spezialisierte, zuvor nicht verfügbare Ressourcen zugänglich gemacht werden [Zuk97, BD00, MM05c, PC07, Kun08] (vgl. Abschnitte 4.2.3 und 4.2.4).

- **Dezentralität:** Für eine dezentrale Ausführung ohne eine übergeordnete Kontrollinstanz ist eine Verteilung des Prozesses auf verschiedene Ausführungseinheiten und deren zielgerichtete Verknüpfung notwendig. Der Wunsch nach Dezentralität kann dabei aus der Gleichberechtigung aller an einem Prozess teilnehmenden Parteien resultieren [WPH07], aus Gründen einer besseren Skalierbarkeit der prozessorientierten Anwendungen bei einer großen Anzahl von Prozessinstanzen motiviert sein [Sch01] oder der Erhöhung der Verfügbarkeit des Gesamtsystems durch Vermeidung eines zentralen Engpasses und somit der Ausfallsicherheit [Sch01, MM05c, Ham05, TS08] dienen (vgl. Abschnitte 4.2.2 und 4.2.5).

- **Kontextbezogene Ausführung:** In einigen Fällen ist die Umgebung, in der ein Prozess oder ein Prozessabschnitt ausgeführt wird, für die Erzielung des beabsichtigten funktionalen oder nicht-funktionalen Ergebnisses relevant. Hierbei ist insbesondere der Ort der Ausführung als ein besonders wichtiges Kriterium zu nennen [JHH+00, Sch01, PC07, Kun08]. Ein Beispiel ist die Durchführung von prozessorientierten Datenerhebungen, durch welche Daten über eine bestimmte geographische Region gewonnen werden sollen (vgl. Abschnitt 4.2.3). Die Ausführung des Prozesses in einem anderen Umfeld würde offensichtlich den eigentlichen Sinn und Zweck dieser Prozessausführung verletzen. Ebenso können an-

dere Rahmenbedingungen der Prozessausführung, wie zum Beispiel die sequentielle Ausführung mehrerer Aufgaben in einem bestimmten Temperaturbereich oder bei einer bestimmten Geschwindigkeit relevant sein, wie zum Beispiel bei der Durchführung wissenschaftlicher Prozesse (vgl. [Pan08, LLF+09]).

- **Mobilität:** Die Ermöglichung der Prozessbearbeitung für Benutzer mit wechselnden Arbeitslokationen erfordert den Umgang mit eingeschränkter Konnektivität, was eine Übertragung von Prozessen oder Prozessabschnitten auf das (mobile) Gerät des Benutzers notwendig macht [CRW98, Sch01, MCA+06, PC07]. Speziell auf die Charakteristika mobiler Systeme abgestimmte Ausführungseinheiten können zudem die entsprechenden Probleme adressieren, welche mit häufigen Verbindungsabbrüchen, einer geringen Bandbreite der Kommunikationsverbindungen und begrenzten Energieressourcen einhergehen [JHH+00]. Zudem können spezielle mobile Dienste (wie zum Beispiel zur Positionsbestimmung) gezielt aufgerufen und in den Prozess integriert werden, welche durch ein zentralisiertes stationäres Prozessmanagementsystem unter Umständen nicht ohne weiteres zugegriffen werden können [Zuk97, MM05c, PC07] (vgl. Abschnitte 4.2.3 und 4.2.4).

- **Abbildung von natürlich verteilten Abläufen:** In vielen Fällen soll durch die verteilte Ausführung eines Prozesses ein möglichst naturgetreues Abbild der Abläufe in der Realwelt geschaffen werden. Im Unternehmenskontext umfasst dies zum Beispiel die Zuordnung von Prozessabschnitten zu organisatorischen Einheiten [Sch01], welche für bestimmte Aufgaben innerhalb der Prozessausführung verantwortlich sind. Dabei ist es das Ziel, die softwaretechnische Realisierung zur Unterstützung von Realweltvorgängen so anzupassen, wie es die Organisationsstruktur oder die natürliche Arbeitsweise von Personen vorgibt und nicht etwa aufgrund von technischen Beschränkungen eine Reorganisation von natürlich verteilten Abläufen zu erzwingen [Jör06, ENS07]. Ein anderes Beispiel ist eine länderübergreifende Verteilung der Prozessausführung, wobei es unter Beteiligung menschlicher Akteure und dem Ausnutzen von unterschiedlichen Zeitzonen automatisch zu einer Beschleunigung der Aufgabenbearbeitung kommen kann [Wes07] (vgl. Abschnitte 4.2.1 und 4.2.2).

- **Überwindung technologischer Unterschiede:** Obwohl durch das Paradigma der dienstorientierten Architekturen und insbesondere durch die Implementierung von Web Services eine hohe Interoperabilität bei der Nutzung technischer Funktionalitäten angestrebt wird, existieren noch viele heterogene Infrastrukturen, die im Allgemeinen nicht durch eine Standard-Prozess-Engine integriert werden können. Für viele Software-Anwendungen, Maschinen und Benutzungsschnittstellen sind spezialisierte Adapter und Kommunikationsprotokolle für die

Ausführungsumgebung erforderlich, welche nicht allen Prozess-Engines gleichermaßen zur Verfügung stehen. Zum Beispiel ist eine *Apache-ODE*-Prozess-Engine[3] aufgrund der Beschränkung auf Web Services in der Regel nicht in der Lage, Java-RMI-Dienste zuzugreifen. Hingegen kann zum Beispiel eine *jBPM*-Prozess-Engine[4] sowohl Web Services als auch beliebige Java-Anwendungen integrieren. Eine entsprechend abstrakte Prozessbeschreibung vorausgesetzt, ist durch die Verteilung der Prozessausführung auf verschiedene Ausführungssysteme somit eine Überwindung dieser Heterogenität möglich [Sch01] und erlaubt damit wiederum die Nutzung zuvor nicht verfügbarer Ressourcen [Kun08] (vgl. Abschnitt 4.2.4).

- **Qualitäts- und Effizienzsteigerung:** Ein wichtiges Ziel der Verteilung liegt in der Optimierung nicht-funktionaler Aspekte. Zum einen hat die Auslagerung von Prozessen im Sinne des *Outsourcing* aus betriebswirtschaftlichem Hintergrund ein großes Kostensenkungspotential [Sch01, Rie03]. Zum anderen kann die Ausführungszeit der Prozesse (insbesondere durch die gleichzeitige Ausführung von parallelen Prozesspfaden auf unterschiedlichen Systemen), die Auslastung von Ausführungssystemen und Ressourcen (insbesondere durch Lastverteilung) und der Kommunikationsaufwand zwischen den beteiligten Parteien durch die bedarfsgerechte Verteilung des Prozesses verbessert werden [MWL08, WML09b]. Hierzu zählt zum Beispiel auch die Verlagerung der Prozessausführung zum Ort der benötigten Daten (vgl. Abschnitte 4.2.5 und 4.2.6).

- **Sicherheit:** Eine Verteilung der Prozessausführung kann in vielen Fällen auch zur Durchsetzung von Sicherheitseigenschaften in Bezug auf Vertraulichkeit, Authentifizierung und Autorisierung erfolgen. Dabei können zum einen die Ressourcen innerhalb einer privaten Ausführungsumgebung die zu schützenden Objekte darstellen, auf welche nur durch eine Ausführungseinheit innerhalb des lokalen Netzes zugegriffen werden darf. Zum anderen kann der Prozess selbst vertrauliche Informationen (z. B. Kreditkartendaten), vertrauliche Kontrollflussdaten (z. B. über die Nutzung von Informationen in Nachfolgeaktivitäten) oder vertrauliche Informationen über die Identitäten von Prozessteilnehmern (z. B. Konkurrenten) enthalten, welche auf einem bestimmten Ausführungssystemen nicht bekannt werden sollen [Sch01, SO04]. Der Prozess selbst stellt in seiner Gesamtheit somit einen vertraulichen Vorgang dar, dessen bedarfsgerechte Aufteilung das *Need-to-know*-Prinzip als wichtiges allgemeines Sicherheitsziel von Softwareanwendungen direkt umsetzen kann.

- **Nachvollziehbarkeit und Zurechenbarkeit:** Aus rechtlichen Gründen muss aus der Sicht von teilnehmenden Parteien oft ein

[3] Apache ODE (Orchestration Director Engine): http://ode.apache.org/
[4] jBPM (Java Business Process Management): http://www.jboss.org/jbpm

ausreichendes Maß an Nachvollziehbarkeit der ausgeführten Prozessschritte mit den entsprechenden Kausalitäten im Kontrollfluss des Prozesses vorhanden sein. Die Zurechenbarkeit von einzelnen Leistungen zu übergeordneten Prozessen ist insbesondere bei der Beteiligung menschlicher Prozessteilnehmer ein wichtiger Aspekt für die Abrechnung und die nachträgliche Auswertung von Leistungen, aber auch für die Verantwortung und Behandlung von Fehlern im Gesamtprozess, für den eine teilnehmende Organisation ggf. verantwortlich gemacht wird. In vielen Fällen ist hierbei eine zentrale Prozessausführung nicht geeignet, da in diesem Fall nicht eindeutig geklärt ist, ob die den Kontrollfluss des Prozesses verwaltende Partei oder eine aufgerufene Ressource einer Organisation für einen Fehler innerhalb eines Prozessabschnitts verantwortlich ist [WPH07] (vgl. Abschnitt 4.2.2).

In vielen Arbeiten wird alternativ oder zusätzlich zu den hier genannten Zielen die Erlangung von Flexibilität als besonderes Ziel der Verteilung hervorgehoben. Da Flexibilität zu einem gewissen Grad von den technischen Verbindungen zwischen Organisationen abhängig ist [GP00, Nur08], entsteht durch die Möglichkeit zur beliebigen Verteilung der Prozessausführung eine neue Art von Anpassungsfähigkeit, welche der *Wirkungsdimension* der Flexibilität zugeordnet werden kann (vgl. Abschnitt 3.2). Diese Dimension beschreibt den Umfang des Bereichs, in dem die Anpassungsfähigkeit eines Systems ermöglicht wird und/oder Auswirkungen auf seine Umwelt hat. Durch die Interaktion mit anderen Systemen und die Möglichkeit zur Verteilung der Prozessausführung kommt es dabei zu einer *externen Anpassungsfähigkeit*, wodurch die Verteilung selbst zur Flexibilisierung prozessorientierter Anwendungen beiträgt [SD03]. Da Anpassungsfähigkeit jedoch in der Regel keinen reinen Selbstzweck darstellt, soll die Erlangung von Flexibilität hier nicht als weiteres Ziel der Verteilung der Prozessausführung aufgeführt werden. Wird von Flexibilität als Ziel der Verteilung gesprochen, so ist Flexibilität daher stets als Mittel zur Verfolgung der hier genannten Ziele aufzufassen.

Bei Betrachtung der identifizierten Ziele lässt sich festhalten, dass die Aufteilung des Prozesses und die Integration der an der Ausführung beteiligten Systeme, Personen und Softwareanwendungen nur unter bestimmten, für den jeweiligen Anwendungskontext relevanten Einschränkungen vorgenommen werden darf. So kann zum Beispiel eine umfangreiche Grafik nicht adäquat auf einem Mobiltelefon betrachtet werden, bestimmte Aufgaben sollen eventuell nur von einem eingeschränkten Personenkreis ausgeführt werden dürfen, oder beteiligte Systeme und Anwendungen müssen auf ihre Auslastung bedacht werden. Eine *Verteilungsstrategie* setzt hierzu die Maßnahmen um, die erforderlich sind, um die Ziele der Verteilung zu erreichen, wobei die konkreten Eigenschaften der Verteilung hinsichtlich Granularität, Verteilungszeit und Zielort der Verteilung bestimmt werden [Sch01]. Der folgende Abschnitt geht daher im Detail auf mögliche Eigenschaften einer Verteilung ein.

4.3.2 Eigenschaften der Verteilung

Wie bereits in Abschnitt 4.1.3 für die Definition dynamisch verteilt ausgeführter Prozesse vorweggenommen wurde, stellt insbesondere die *Granularität* der Verteilung eine wesentliche Eigenschaft der verteilten Ausführung von prozessorientierten Anwendungen dar. Die Granularität der Verteilung gibt an, ob ein Prozess in seiner Gesamtheit einem Ausführungssystem zugeteilt wird, ob der Prozess in einzelne Teilprozesse aufgespalten wird, für die jeweils verschiedene Ausführungseinheiten verantwortlich sein können, oder ob jede einzelne Aktivität einer anderen Kontrollinstanz zugewiesen wird [Sch01]. Wird ein ganzer Prozess einer Ausführungseinheit zugewiesen, so stellt dies durch die Bildung einer *maximalen Partition* einen Spezialfall der Verteilung dar, welcher der zentralen Prozessausführung entspricht [Wut10]. Existieren nur Prozessabschnitte mit genau einer Aktivität, so stellen diese im Gegensatz dazu *minimale Partitionen* dar. Werden die minimalen Partitionen jeweils auf verschiedene Ausführungseinheiten verteilt, so stellt dieses eine *maximal* bzw. *voll verteilte Prozessausführung* dar [Sch01]. Oft existieren in diesem Fall keine Ausführungseinheiten in Form von vollwertigen Prozess-Engines (vgl. Abschnitt 2.7), da hierbei im Extremfall jede einzelne Ressource selbst für die Ausführung ihrer Aktivität und die Weiterleitung der Prozessinstanz an den Bearbeiter der Nachfolgeaktivität verantwortlich ist. Hierdurch entsteht zwischen den prozessausführenden Systemen in der Regel ein großer Kommunikationsaufwand [WML09b, Wut10]. Zwischen den Extremfällen der zentralen Ausführung eines ganzen Prozesses und einer maximal verteilten Ausführung muss daher in Hinblick auf ein bestimmtes Optimierungskriterium (d. h. in Hinblick auf die oben genannten Ziele) eine sinnvolle Partitionierung bestimmt werden. Dies kann grundsätzlich zur Entwicklungszeit oder zur Laufzeit des Prozesses geschehen. Die tatsächliche Verteilung kann dann entweder auf der Ebene eines bestimmten *Prozessmodell* stattfinden (und damit für alle hiervon abgeleiteten Prozessinstanzen gelten) oder auf der Ebene einer einzelnen *Prozessinstanz* vorgenommen werden.

Ein weiteres Kriterium der verteilten Ausführung ist die Bestimmung und Zuordnung der *Zielorte* der Verteilung [Sch01]. WESKE [Wes07] klassifiziert diese nach der *Reichweite* der möglichen Verteilung in *organisationsinterne* und *organisationsübergreifende* Verteilung. Bei einer organisationsinternen Verteilung werden verschiedene Ausführungseinheiten ausgewählt, welche jedoch alle im direkten Einflussbereich einer Organisation bzw. einer Organisationseinheit liegen. Bei einer organisationsübergreifenden Verteilung sind Ausführungseinheiten verschiedener Organisationen beteiligt. Diese Unterscheidung ist vor allem in Hinblick auf die zu erwartenden *Heterogenität* der beteiligten Systeme relevant. So ist im Fall einer organisationsübergreifenden Verteilung oft zunächst eine Interoperabilisierung des Prozessmanagements erforderlich, zum Beispiel durch die Nutzung einer gemeinsamen oder standardisierten Prozessbeschreibungssprache. Orthogonal hierzu ist eine Betrachtung der speziellen Eigenschaften potentieller Infrastrukturen in Bezug auf

ihre *Mobilität* erforderlich. Hierbei kann zwischen *stationären*, *mobilen* und *hybriden Infrastrukturen* als Zielorte der Verteilung unterschieden werden. Die Auswahl der Ausführungseinheiten kann in allen Fällen auf der Basis verschiedener Strategien vorgenommen werden, welche wiederum aus den Zielen der Verteilung resultieren. Diese Strategien reichen dabei von einer zufälligen oder zyklischen Auswahl bis hin zu komplexen Kostenfunktionen zur Berechnung der optimalen Auslastung oder zur Minimierung des Kommunikationsaufwands durch die Berücksichtigung der Nähe von Ausführungseinheiten zu den benötigten Ressourcen [BD99, JHH+00, Sch01].

Bei der Zuordnung eines Prozessabschnitts zu einer bestimmten Ausführungsumgebung ist stets das ausgewählte System für die Ausführung dieser Partition und den Aufruf von Ressourcen verantwortlich. Betrachtet man nur die *lokale* Ausführung einer einzelnen Partition, so ist diese stets durch eine zentrale Systemarchitektur gekennzeichnet. Auf der übergeordneten Ebene können jedoch unterschiedliche Systemarchitekturen vorherrschen, welche den Vorgang der Verteilung selbst, die Übergabe des Kontrollflusses und die Synchronisation von Daten steuern. Hierbei kann zwischen einer *zentralen* und einer *dezentralen Koordination* unterschieden werden [BD99, Sch01, JSH+01]. Bei einer *zentralen Koordination* ist eine übergeordnete Instanz für die Partitionierung des Prozesses, die Auswahl und Zuweisung der Prozesspartitionen zu Ausführungseinheiten sowie für eine etwaige Zusammenführung von Prozessfragmenten verantwortlich. Die beteiligten Ausführungseinheiten kommunizieren hierbei also nicht zwingend direkt untereinander, sondern werden durch eine zentrale Instanz koordiniert. Die Rolle der zentralen Instanz kann dabei durchaus auch durch den Modellierer und/oder den Initiator der auszuführenden Prozesse wahrgenommen werden, während die beteiligten Ausführungseinheiten hierarchisch untergeordnet sind und nicht ihrerseits selbst verteilt auszuführende Prozesse in das System einbringen [vdA99, ACMM07]. Bei einer *dezentralen Koordination* geschehen die genannten Aufgaben jeweils durch die Ausführungseinheiten, welche aktuell an der Bearbeitung des Prozesses beteiligt sind. Dies setzt insbesondere das Vorhandensein eines gemeinsamen Kommunikationsprotokolls voraus. In diesem Fall sind die beteiligten Ausführungseinheiten in der Regel gleichberechtigt und können ihrerseits eigene Prozesse initiieren und verteilt ausführen lassen [BKK+04]. *Hybride Varianten* erlauben schließlich die Wahl eines temporären Koordinators für individuelle Prozessinstanzen, wobei oft der Initiator des Prozesses auch die Rahmenbedingungen für dessen verteilte Ausführung festlegt und Aufgaben im Bereich der Synchronisation übernehmen kann [ZKL09b, ZKL09a].

In Abhängigkeit des definierten Kontrollflusses und der identifizierten Ziele einer verteilten Prozessausführung lassen sich zudem verschiedene Varianten der Verteilung ableiten, welche entweder eine *sequentielle* oder eine *parallele Ausführung* von Partitionen durch verschiedene Ausführungseinheiten zur Folge haben. Werden parallele Partitionen verteilt ausgeführt, kommt dabei nicht nur der Synchronisation des Kontrollflusses an einer etwaigen Zu-

Klassifizierung	Eigenschaft	Beschreibung
Granularität der Verteilung	Gesamtprozess	entspricht zentraler Ausführung
	Teilprozesse	Partitionen mit unterschiedlicher Granularität
	einzelne Aktivitäten	voll verteilte Ausführung
Abstraktionsebene der Verteilung	Prozessmodell	gleiche Verteilung für alle Instanzen
	Prozessinstanz	individuelle Verteilung für Instanzen
Zielort der Verteilung	organisationsintern	i.d.R. geringere Heterogenität
	organisationsübergreifend	i.d.R. höhere Heterogenität
Infrastrukturen	stationär	ausschließlich stationäre Ausführungseinheiten
	mobil	ausschließlich mobile Ausführungseinheiten
	hybrid	Integration stationärer und mobiler Ausführungseinheiten
Koordination	zentralisiert	Übergeordnete zentrale Instanz
	dezentralisiert	Gleichberechtigte Parteien ohne zentrale Kontrolle
	hybrid	Wahl eines temporären Koordinators
Ausführung von Partitionen	sequentiell	keine parallel verteilt ausgeführten Partitionen
	parallel	parallel verteilt ausgeführte Partitionen
Synchronisationszeitpunkte	unmittelbar	Austausch aktueller Zustandsdaten nach jeder Aktivität
	verzögert	Möglichkeit zur Offline-Bearbeitung von Prozessabschnitten
Zeitpunkt der Verteilung	Modellierung	Fixierung der Verteilung im Prozessmodell
	Deployment	Deployment einzelner Partitionen
	Instantiierung	Individuelle, aber statische Verteilung für jede Prozessinstanz
	Laufzeit	Dynamische Verteilung während der Ausführung
Durchführung der Verteilung	manuell	vollständig durch Entwickler/Benutzer durchgeführt
	unterstützt	teilweise automatisiert
	automatisch	vollständig automatisiert

Tabelle 4.1: Klassifizierung verteilt ausgeführter Prozesse

sammenführung paralleler Pfade, sondern auch der Synchronisation gemeinsam genutzter Daten eine besondere Bedeutung zu [CR04]. In Hinblick auf die Synchronisation von Daten kann dabei eine *unmittelbare Synchronisation* vor bzw. nach jeder Ausführung einer Aktivität auf den parallelen Pfaden durchgeführt werden, was in einem hohem Kommunikationsaufwand und – bei einer vorübergehenden Unerreichbarkeit von Ausführungssystemen – in einer Verzögerung der Prozessausführung resultieren kann [WWWD96, Sch01, CR04]. Alternativ kann der *Synchronisationszeitpunkt verzögert* werden, zum Beispiel um auch die Integration mobiler Ausführungsumgebungen mit einer eingeschränkten Konnektivität zu erlauben [WWWD96, Sch01]. Die notwendige Synchronisation kann zudem Einfluss auf die Partitionierung des Prozesses haben, zum Beispiel wenn synchronisationsaufwendige Prozesspartitionen zu einer gemeinsamen Partition zusammengefasst werden sollen, um den Kommunikationsaufwand möglichst gering zu halten [Sch01].

Für die Flexibilisierung verteilter Prozessausführung spielt insbesondere der *Zeitpunkt* eine besondere Rolle, zu dem die genannten Parameter der Verteilung bestimmt werden. Dabei kann bei einer verfeinerten Betrachtung unterschieden werden, ob Parameter zur Entwicklungszeit des Prozesses (also bereits während der Modellierung), während des Deployments des Prozessmodells, vor der Instantiierung jeder Prozessinstanz oder für jede Prozessinstanz zur Laufzeit bestimmt werden. Werden Verteilungsparameter erst während der Laufzeit bestimmt, so spricht man auch von einer *dynamischen Verteilung*,

während in den anderen Fällen von einer *statischen Verteilung* gesprochen wird (vgl. [Sch01] und Abschnitt 4.1.3). Dabei kann – zumindest theoretisch – für jeden Parameter der Verteilung ein individueller Verteilungszeitpunkt existieren [Sch01]. Zum Beispiel kann eine Partitionierung des Prozesses zur Entwicklungszeit vorgenommen und die entsprechenden Ausführungseinheiten erst zur Laufzeit der Prozessinstanz zugeordnet werden. Liegt der Verteilungszeitpunkt vor der Laufzeit des Prozesses, entsteht zur Ausführungszeit keine Belastung des Netzwerks mit der Durchführung der Verteilung. Ein großer Nachteil ist jedoch, dass während der Ausführung nicht mehr flexibel auf aktuelle Ereignisse reagiert werden kann. Somit ist nur bei einer dynamischen Verteilung die Berücksichtigung des aktuellen Kontextes, der Ergebnisse vorhergehender Aktivitätsausführungen und die darauf basierenden Anforderungen anstehender Aktivitäten möglich. Die Möglichkeit, Veränderungen zur Laufzeit zu erlauben, ist besonders dann wichtig, wenn aktive Prozessinstanzen lange laufen oder wenn Veränderungen schnell durchgeführt werden sollen, etwa um durch eine rasche Anpassung Wettbewerbsvorteile realisieren zu können [WHKS98]. Eng mit dem Zeitpunkt der Verteilung und der Anpassungsfähigkeit der prozessorientierten Anwendung verknüpft ist daher auch die Fähigkeit zur *Umverteilung* als Möglichkeit zur Änderung von einmal bestimmten Verteilungskonfigurationen [Sch01].

Eine Verteilung der Prozessausführung kann schließlich vollständig *manuell*, technisch *unterstützt* oder vollständig *automatisch* durchgeführt werden. Eine technische Unterstützung der Verteilung kann dabei entweder durch die Automatisierung einzelner Schritte (z. B. der Partitionierung oder der Auswahl von Ausführungseinheiten) oder durch die Bereitstellung von Werkzeugen zur interaktiven Teilnahme des Benutzers am Entscheidungsprozess realisiert werden. In Hinblick auf die Flexibilisierung der verteilten Prozessausführung ist dabei sowohl zur Minimierung des Aufwands für den Benutzer als auch zur Beschleunigung eines Anpassungsvorgangs zumindest eine teilweise Automatisierung der Verteilung anzustreben (vgl. Abschnitt 3.10).

Tabelle 4.1 zeigt eine Zusammenfassung der identifizierten Eigenschaften verteilt ausgeführter Prozesse. Zusammenfassend lässt sich festhalten, dass die Unterstützung möglichst vielfältiger Eigenschaften und Kombinationen an Eigenschaften hinsichtlich der Flexibilität dynamisch verteilter Prozesse einen wichtigen Qualitätsaspekt darstellt. In seiner Arbeit zur verteilten Workflow-Bearbeitung bezeichnet SCHAMBURGER [Sch01] die Festlegung der Partitionierung und Allokation einer zu verteilenden Anwendung auf ein verteiltes System entsprechend als *Verteilungskonfiguration*. Die Menge der möglichen Verteilungskonfigurationen begründet dabei den sogenannten *Konfigurationsraum*, welcher durch die individuellen Rahmenbedingungen des Anwendungsbereichs eingeschränkt werden kann. Der resultierende *zulässige Konfigurationsraum* kann dabei zum Beispiel durch Regeln beschrieben werden, welche die gewünschten Eigenschaften der Verteilung unter bestimmten Situationen angeben. Ergänzend nennt WUTKE [Wut10] den durch ein bestimmtes *Verteilungsmodell* realisierten Konfigurationsraum das *Verteilungsspektrum*. Dabei

stellt die potentielle Nutzung eines möglichst breiten Spektrums an Verteilungskonfigurationen einen wichtigen Flexibilitätsaspekt dar, welcher mit der in Abschnitt 3.2 identifizierten allgemeinen Dimension des *Anpassungsspielraums* korrespondiert. Im Folgenden werden klassische Verteilungsmodelle vorgestellt und in Hinblick auf die hier identifizierten Eigenschaften und ihr Verteilungsspektrum diskutiert.

4.3.3 Klassische Verteilungsmodelle

Ein erster Schritt zur Betrachtung der aktuellen Flexibilität verteilter prozessorientierter Anwendungen stellt die Identifikation von relevanten Verteilungsmodellen und Kooperationsformen dar, welche der Ausführung verteilter Prozesse zugrunde liegen können. Ein *Verteilungsmodell* bezeichnet dabei die Art und Weise, wie die Verantwortlichkeit für die Prozessausführung an verschiedene Ausführungseinheiten zugewiesen wird [BD99]. Mit der Klassifikation verteilter Prozesse nach VAN DER AALST [vdA99, vdA00] und einer alternativen Darstellung nach SCHULZ und ORLOWSKA [SO04] sollen im Folgenden zwei unterschiedliche, aber einander nicht ausschließende Klassifikationen für Verteilungsmodelle aufgezeigt werden.

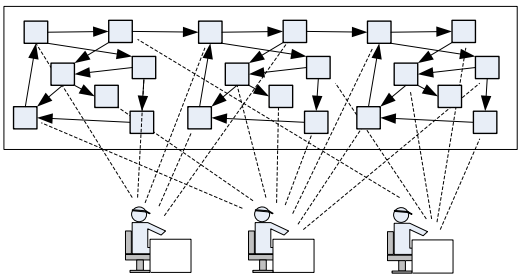
Klassifikation nach VAN DER AALST

Nach VAN DER AALST werden klassisch fünf verschiedene Arten der Verteilung von Prozessen unterschieden [vdA99, vdA00] (vgl. Abbildung 4.10):

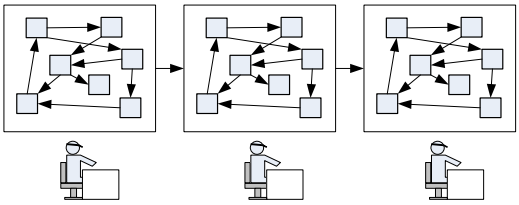
- **Verteilung durch gemeinsame Benutzung von Ressourcen (*Capacity Sharing*):** Bei dieser Vorgehensweise werden einzelne Aktivitäten eines Prozesses durch externe Ressourcen ausgeführt, wobei die Kontrolle über den Prozess jedoch nicht verteilt wird und einer einzigen (zentralen) Ausführungseinheit fest zugeordnet ist (vgl. Abbildung 4.10a). Die Zuordnung der Aktivitäten zu Ressourcen ist jedoch flexibel, solange die Ressourcen von der zentralen Ausführungsumgebung zugegriffen werden können. Diese Art der Verteilung entspricht der in Abschnitt 4.1.2 definierten ressourcenbasierten Verteilung und ist daher nicht Betrachtungsgegenstand dieser Arbeit.

- **Verteilung durch verkettete Ausführung (*Chained Execution*):** Hierbei wird der Prozess in aufeinander folgende Teilprozesse zerlegt, welche durch verschiedene Ausführungseinheiten in der festgelegten Reihenfolge ausgeführt werden (vgl. Abbildung 4.10b). Die Verteilung des Prozessmodells kann zur Entwicklungs- oder zur Laufzeit erfolgen. Zur Übergabe des Kontrollflusses sind festgelegte Schnittstellen erforderlich.

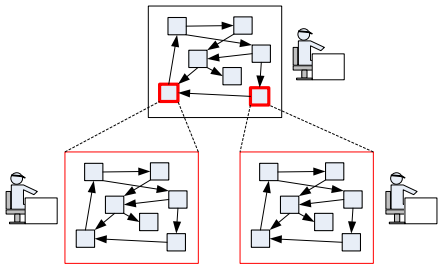
- **Verteilung durch Untervergabe von Aufgaben (*Subcontracting*):** Nach diesem Verteilungsmodell werden einzelne Subprozesse des Gesamtprozesses an externe Ausführungseinheiten ausgelagert. Der Aufbau des Gesamtsystems ist hierbei hierarchisch, d. h. eine



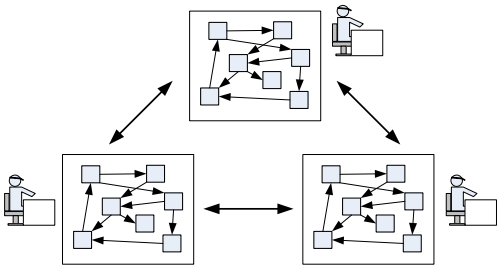
(a) Verteilung durch gemeinsame Benutzung von Ressourcen (*Capacity Sharing*)



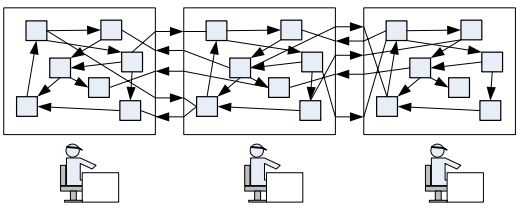
(b) Verteilung durch verkettete Ausführung (*Chained Execution*)



(c) Verteilung durch Untervergabe von Aufgaben (*Subcontracting*)



(d) Verteilung durch Übergabe von Prozessinstanzen (*Case Transfer*)



(e) Verteilung durch lose Kopplung (*Loosely Coupled*)

Abbildung 4.10: Arten der organisationsübergreifenden Verteilung von Prozessen nach VAN DER AALST (vdA99, vdA00)

Ausführungseinheit hat die übergeordnete Kontrolle über den initiierenden Prozess (vgl. Abbildung 4.10c). Anstelle des ausgelagerten Subprozesses wird eine abstrakte Aktivität (z. B. eine *Blockaktivität*, vgl. Abschnitt 2.4.3) als Platzhalter für die extern zu erbringende Leistung eingefügt.

- **Verteilung durch Übergabe von Prozessinstanzen (*Case Transfer*):** Bei dieser Form der Verteilung wird einer Ausführungsumgebung die Ausführung einer bestimmten Prozessinstanz zugewiesen, z. B. zur Lastverteilung eines hohen Prozessaufkommens auf verschiedene Ausführungseinheiten (vgl. Abbildung 4.10d). Dazu wird im Vorfeld das der Ausführung zugrunde liegende Prozessmodell repliziert und an alle potentiellen Ausführungseinheiten verteilt. Unter bestimmten Umständen ist auch ein Wechsel der Ausführungseinheit zur Laufzeit der Prozessinstanz möglich. Die erweiterte Variante des *Extended Case Transfer* erlaubt zudem zusätzlich geringfügige inhaltliche Änderungen an der Ausführung einer Prozessinstanz, indem unterschiedlichen Ausführungseinheiten zur Entwicklungszeit eine jeweils leicht abgewandelte Version des Prozessmodells zugewiesen wird.

- **Verteilung durch lose Kopplung (*Loosely Coupled*):** Bei dieser Vorgehensweise erfolgt eine beliebige Zerteilung des Gesamtprozesses in Verantwortungsbereiche einzelner Ausführungseinheiten. Die resultierenden Subprozesse des übergeordneten gemeinsamen Prozesses (*globaler Prozess* [vdA00]) werden hierbei auf verschiedenen Systemen ausgeführt, wobei die Ausführung der Subprozesse lokal gekapselt ist (*lokale Prozesse* [vdA00]). Der Austausch von Daten und der Kontrollfluss ist durch ein gemeinsames Kommunikationsprotokoll vorgegeben (vgl. Abbildung 4.10e). Die Umsetzung der lokalen Prozesse kann hierbei plattformabhängig bzw. organisationsintern erfolgen, solange das Kommunikationsprotokoll über festgelegte Schnittstellen eingehalten wird.

Während *Capacity Sharing* unter zentraler Kontrolle einer Prozess-Engine ausgeführt wird, ist die Ausführung des Prozesses in den anderen Fällen verteilt zentralisiert oder gänzlich dezentral organisiert (vgl. [vdA99]). Es handelt sich daher um verteilte Prozessausführungen im Sinne dieser Arbeit. Mit Ausnahme der ersten Verteilungsvariante ist zudem jeweils eine *Partitionierung* des Prozesses erforderlich. Van der Aalst unterscheidet in Hinblick darauf, ob *Modell-* oder *Instanzebene* eines Prozesses betroffen ist, zwei Dimensionen der Partitionierung (vgl. Abbildung 4.11):

Eine *vertikale Partitionierung* der Prozessausführung auf der Ebene von Prozessinstanzen umfasst die Aufteilung der einzelnen Instanzen eines Prozessmodells auf verschiedene Ausführungseinheiten, ohne das Prozessmodell als Schablone für deren Ausführung dabei (physikalisch) zu zerteilen. Die Auswahl von geeigneten Ausführungseinheiten sowie der Zeitpunkt, an dem eine Prozessinstanz an ein anderes System übertragen werden soll, kann

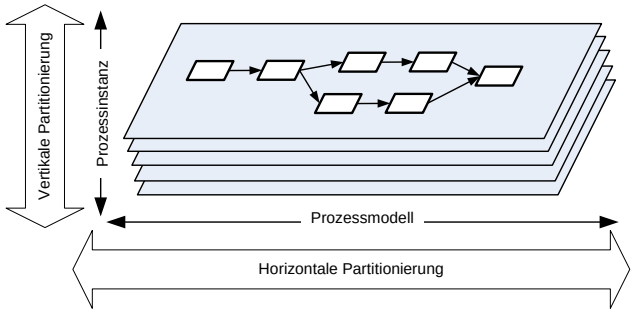


Abbildung 4.11: Horizontale und vertikale Partitionierung (vdA99)

durch benutzerdefinierte Rahmenbedingungen (*Transfer Policies*) festgelegt werden. Diese Art der Partitionierung hat nach VAN DER AALST den Vorteil, dass eine Instanz zu jedem beliebigen Zeitpunkt gänzlich durch eine bestimmte Ausführungseinheit verwaltet wird, was den Aufwand für Koordination zwischen den Teilnehmern minimiert und die Verantwortlichkeit für die Ausführung klar determiniert. Die Notwendigkeit der Replikation des Prozessmodells und die begrenzte Parallelisierbarkeit der Ausführung einer einzelnen Instanz stehen diesen Vorteilen jedoch entgegen [vdA99].

Basierend auf der Dimension von Prozessmodellen wird bei einer *horizontalen Partitionierung* ein einzelnes Prozessmodell auf verschiedene Ausführungseinheiten aufgeteilt, was eine explizite Zuordnung der zugehörigen Prozessinstanzen unnötig macht, da sie automatisch aus der vorangegangenen Verteilung resultiert [vdA99]. Bei dieser Art der verteilten Prozessausführung ist die oben genannte Parallelverarbeitung einzelner Instanzen möglich, wenn das Prozessmodell entsprechend aufgeteilt wurde. Zudem wird jede an der Ausführung teilnehmende Partei jeweils nur mit dem für sie explizit relevanten Teil der Prozessbeschreibung konfrontiert. Der Preis hierfür ist jedoch ein höherer Aufwand für die Koordination zwischen den beteiligten Ausführungseinheiten, z. B. in Hinblick auf Ausnahmebehandlung und Kontrolle von Nebenläufigkeit [vdA99].

Nach dieser Klassifizierung ist sowohl die *Verteilung durch Untervergabe* als auch die *lose gekoppelte Verteilung* durch eine horizontale Partitionierung und somit durch eine physikalische Aufteilung des Prozessmodells gekennzeichnet. Die einzelnen Prozessteilnehmer haben somit keine Kenntnis über den Gesamtprozess. Die *Verteilung durch Übergabe von Prozessinstanzen* verwendet eine vertikale Partitionierung und die *verkette Ausführung* stellt nach VAN DER AALST eine hybride Partitionierung dar. Dabei spricht VAN DER AALST insbesondere der *Verteilung durch Übergabe von Prozessinstanzen* und der *lose gekoppelten Verteilung* eine besonders hohe Flexibilität zu [vdA99]. Die Flexibilität bei einer *lose gekoppelten Verteilung* entsteht dabei vor allem durch eine hohe Verteilungstransparenz und die Möglichkeit, dass jede prozessausführende Partei die Implementierung der für den Gesamtprozess zu realisierenden Funktion autonom und unabhängig von den anderen Parteien realisieren kann. Diese Flexibilität entsteht also durch Abstraktion

von inhaltlichen und technischen Details der Prozessausführung, während die ursprüngliche Prozessbeschreibung für die (technische) Verteilung angepasst wird. Im Fall der vertikalen Partitionierung durch *Übergabe von Prozessinstanzen* wird die Verteilung außerhalb der Prozessbeschreibung realisiert. Die Flexibilität folgt hier also aus einer klaren Separation der Verteilungsstrategie und der inhaltlichen Beschreibung.

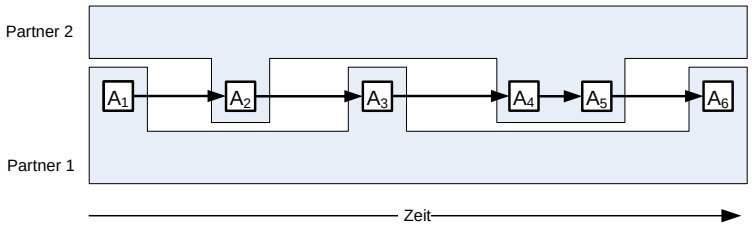
Klassifikation nach SCHULZ und ORLOWSKA

Eine teilweise ähnliche Unterscheidung über die Art der Verzahnung organisationsübergreifender Prozesse wird in der Arbeit von SCHULZ und ORLOWSKA getroffen [SO04], wobei nur die Ebene der horizontalen Partitionierung eines Prozesses betrachtet wird. Ergänzend zu den Ausführungen von VAN DER AALST wird hierbei jedoch auch auf die speziellen Rahmenbedingungen zur Realisierung dieser Verteilungsformen und insbesondere auf die hierfür notwendigen Anpassungen der Prozessbeschreibung eingegangen. SCHULZ und ORLOWSKA unterscheiden verteilt ausgeführte Prozesse dabei hinsichtlich ihrer Hierarchisierung. Abbildung 4.12a zeigt einen organisationsübergreifend auszuführenden Vorgang, der entweder durch einen (dezentral) *verteilten Prozess* oder durch die (zentrale) *Auslagerung von Prozessteilen* realisiert werden kann (vgl. [SO04]):

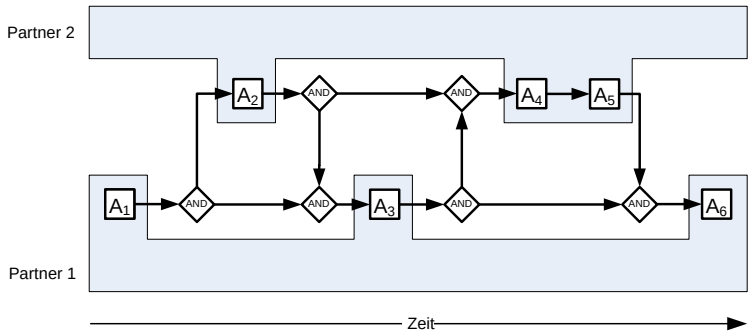
- **Verteilter Prozess (*Distributed Workflow*):** Diesem Modell liegt eine allen Ausführungseinheiten gemeinsame Prozessbeschreibung zugrunde, von der die einzelnen Teilnehmer jeweils festgelegte Aktivitäten eigenverantwortlich erfüllen. Die Teilergebnisse der Ausführungseinheiten werden an einer entsprechenden Position im Kontrollfluss über die Integration von zusätzlichen Kontrollflusskonstrukten (*AND-Split* und *AND-Join*) synchronisiert (vgl. Abbildung 4.12b).

- **Ausgelagerter Prozess (*Outsourced Workflow*)**: Bei einem ausgelagerten Prozess wird die Verarbeitung einer entfernt ausgeführten Aktivität, z. B. in Form eines Subprozesses lediglich angestoßen und erfolgt dann relativ autark durch die beauftragte Ausführungseinheit. Gegebenenfalls wird am Ende der Ausführung ein Ergebnis zurückgeliefert. Die Realisierung einer solchen Verteilung erfolgt durch abstrakte Aktivitäten (z. B. *Blockaktivitäten*, vgl. Abschnitt 2.4.3), welche durch die individuelle Leistung der jeweiligen Outsourcing-Partner ausgefüllt werden (vgl. Abbildung 4.12c).

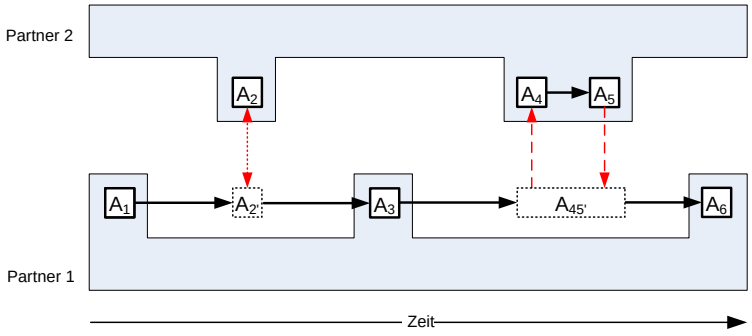
Das Verteilungsmodell der *ausgelagerten Prozesse* entspricht dabei der von VAN DER AALST genannten *Verteilung durch Untervergabe von Aufgaben* (vgl. Abbildung 4.10c). Die *verteilten Prozesse* hingegen realisieren eine andere Art von Verteilung, bei welcher der Gesamtprozess in mehrere kommunizierende Teilprozesse zerlegt wird. Wie den Abbildungen 4.12b und 4.12c zu entnehmen ist, kann die ursprünglich geplante Verteilung des funktionalen Prozesses (vgl.



(a) Verteilt auszuführender Prozess: Logische Zuordnung der durchzuführenden Aktivitäten A_1 bis A_6 zu Partner 1 und Partner 2



(b) Verteilter Prozess: Realisierung der Verteilung durch Einführung zusätzlicher Verzweigungs- und Synchronisationskonstrukte



(c) Ausgelagerter Prozess: Realisierung durch Einführung zusätzlicher (abstrakter) Aktivitäten

Abbildung 4.12: Arten der organisationsübergreifenden Verteilung von Prozessen nach SCHULZ und ORLOWSKA (SO04) (Beispiel)

Abbildung 4.12a) jedoch in beiden Fällen nur durch die zusätzliche Anreicherung der Prozessbeschreibung mit nicht-fachlichen Kontrollflusskonstrukten realisiert werden. Der Zeitpunkt, die Granularität und ggf. der Zielort der Verteilung wird somit in der Prozessbeschreibung festgeschrieben. Die teilnehmenden Ausführungseinheiten müssen zudem über geeignete Schnittstellen verfügen, welche vor dem Beginn der Prozessausführung festgelegt und implementiert werden müssen. Eine Anpassung dieser Verteilungsparameter zur Laufzeit der Prozessinstanzen ist somit nicht möglich, was hinsichtlich der Flexibilität dieser (horizontalen) Partionierungsansätze und deren Eignung für dynamische Umgebungen zu bedenken ist.

4.4 Anforderungsanalyse

Prozessorientierte Anwendungen, welche auf Basis aktueller Umgebungsbedingungen verteilt ausgeführt werden sollen, unterliegen klassischen und neuen Anforderungen, welche einerseits aus der Verteilung des Prozesses resultieren und andererseits von der anzustrebenden Flexibilität zur Ermöglichung von Anpassungen zur Laufzeit des Prozesses geprägt sind. In diesem Abschnitt werden beide Arten von Anforderungen genauer beschrieben. Die Anforderungen werden dabei aus drei unterschiedlichen Quellen zusammengefasst: Zum einen werden Anforderungen anhand der vorausgegangenen Literaturrecherche zur Flexibilisierung von prozessorientierten Anwendungen im Allgemeinen abgeleitet. Desweiteren werden die Aspekte der entsprechend durchgeführten Klassifikationen verteilt ausgeführter Prozesse im Speziellen berücksichtigt. Schließlich werden die bei der Betrachtung der Fallbeispiele aus verschiedenen Anwendungsgebieten identifizierten (praktischen) Herausforderungen integriert.

4.4.1 Allgemeine Anforderungen verteilt ausgeführter Prozesse

Eine verteilte Ausführung von prozessorientierten Anwendungen unterliegt auch ohne besondere Ansprüche an ein flexibles Verhalten gewissen Herausforderungen und Voraussetzungen, welche diese von einer zentralen Ausführung von Prozessen unterscheidet. Basierend auf der vorangegangenen Literaturrecherche und den vorgestellten Fallbeispielen werden diese Anforderungen im Folgenden beschrieben und am Ende der Erläuterungen in einer Tabelle (vgl. Tabelle 4.2) zusammengefasst. Zur Unterscheidung zu den speziellen Anforderungen dynamisch verteilter Prozesse (vgl. Abschnitt 4.4.2) sind die allgemeinen Voraussetzungen und Anforderungen verteilt ausgeführter Prozesse mit dem Buchstaben **A** gekennzeichnet:

- A_1 **Interoperabilität:** Für die verteilte Ausführung eines Prozesses wird eine gemeinsame Basis benötigt, welche die Syntax und Semantik des auszuführenden Prozesses mit seinen Aktivitäten und seinem Kontrollfluss festlegt [Hol95, JNDV99, Wes07]. Eine Möglichkeit hierfür ist die Einigung auf eine gemeinsame Prozessbeschreibungssprache [PC07], was durch zahlreiche Standardisierungsinitiativen unterstützt wird (vgl. Abschnitt 5.2.4). Des Weiteren werden festgelegte Schnittstellen zum Austausch von Nachrichten für die Koordination der verteilten Prozessausführung benötigt [vdAvH02, Wes07]. In Bezug auf die verteilte Ausführung eines Prozesses umfasst dies also geeignete Schnittstellen zur Verteilung des Prozesses, zur Übergabe des Kontrollflusses, zum Austausch von Prozessdaten und zur Bereitstellung und Sammlung von Daten über die Prozessausführung selbst.

- A_2 **Autonomie beteiligter Parteien:** Nehmen unterschiedliche Organisationen oder autonome Organisationseinheiten an der verteilten

Ausführung eines Prozesses teil, so sind geeignete Maßnahmen zu treffen, um die aus der Autonomie der beteiligten Parteien resultierenden individuellen Rahmenbedingungen für die Kooperation und den Zugang zu Informationen und Funktionalitäten zu berücksichtigen [KWA99a, CCSD04]. Dies betrifft zum einen den potentiellen Ausschluss hierarchisch übergeordneter Komponenten und somit die Wahl von dezentralen Architekturformen [YY07, WW08], und zum anderen die Beschränkung des Zugriffs auf Ressourcen, Funktionalitäten und Informationen einer internen Prozessausführung [SO04, Sch04a, Sch05], insbesondere zur Überwachung der Prozesse [CCSD04]. Teilweise können diese Anforderungen durch die Verhandlung und Spezifikation von Verträgen in Form von *Policies* oder *Service Level Agreements (SLAs)* berücksichtigt werden (z. B. [HLGG00, LDK04, WKK+10]).

A_3 **Verteilungsstrategie und -durchführung:** Basierend auf den Zielen der Verteilung sowie der organisatorischen und technischen Rahmenbedingungen wird eine geeignete Strategie benötigt, um den fachlich festgelegten Prozess zu partitionieren, die für eine Zuordnung geeigneten Ausführungseinheiten zu bestimmen und die entsprechend resultierenden Prozesspartitionen den gewählten Ausführungseinheiten zuzuweisen [Sch01, KL06]. Dies setzt voraus, dass Ausführungseinheiten bekannt sind bzw. aufgefunden werden können [Kun08] und ein funktionaler oder nicht-funktionaler Zusammenhang zwischen der auszuführenden Prozesspartition und den Ausführungseinheiten existiert, welcher in geeigneter Weise ausgedrückt werden kann, um eine manuelle oder automatische Durchführung der Verteilung vorzunehmen [Wut10].

A_4 **Korrektheit der Ausführung:** Das funktionale Ergebnis einer verteilten Ausführung eines Prozesses muss stets einem Ergebnis entsprechen, welches auch durch eine entsprechende zentrale Ausführung desselben Prozesses erreicht werden könnte. Im Speziellen bedeutet dies, dass nicht zwei Parteien (gleichzeitig) an der selben Aufgabe des Prozesses arbeiten dürfen oder deren Bearbeitung verschiedener Aufgaben keine Effekte verursachen dürfen, welche bei einer zentral verwalteten Ausführung nicht aufgetreten wären [AGK+95]. Diese Forderung umfasst vor allem den kontrollierten Zugriff auf gemeinsame Daten, d. h. die Weitergabe von aktuellen Variablenwerten im Fall einer sequentiellen Ausführung, deren Replikation und Synchronisation im Fall einer parallelen Ausführung von Aktivitäten [AGK+95, WW97, MWW+98, RRD03] sowie die Berücksichtigung und Behandlung von Inkonsistenzen, welche durch die nebenläufige Bearbeitung von Daten in parallel ausgeführten Prozesspfaden entstehen können [Sch01, CR04]. Durch die verteilte Ausführung eines Prozesses dürfen zudem keine organisatorischen oder technischen Vorgaben verletzt werden [Sch01], z. B. die vorgegebene Zuordnung von Ressourcen zu Aktivitäten oder die spezifizierte Art der Aktivitätsdurchführung (z. B. manuell oder automatisch).

A_5 **Fehlererkennung und -behandlung:** Um eine korrekte Ausführung des Prozesses zu gewährleisten, ist auch bei der verteilten Ausführung eine angemessene Erkennung und Behandlung von Fehlern erforderlich. Fachliche Fehler können dabei in der Regel – wie auch bei einer zentralen Ausführung – durch die explizite Beschreibung von Fehlerzuständen und entsprechenden semantischen Fehlerbehandlungsmaßnahmen adressiert werden (vgl. Abschnitt 2.4.5). Bei technischen Fehlern kann allgemein zwischen lokalen und globalen Auswirkungen eines Fehlers unterschieden werden [VGBA99, CR04, CYCX05, MM06]. Fehler und Fehlerbehandlungsmaßnahmen auf lokaler Ebene unterliegen dabei der Verantwortung des lokalen Systems. Beispiele hierfür sind die Auflösung von lokalen Zugriffskonflikten durch Verwendung der Transaktionskontrolle einer lokalen Datenbank oder das Ersetzen eines nicht verfügbaren Prozessteilnehmers durch einen anderen Mitarbeiter der betreffenden Organisation. Fehler auf globaler Ebene oder lokale Probleme, welche nicht durch lokale Fehlerbehandlungsmaßnahmen gelöst werden können, machen eine Fehlerbehandlung auf globaler Ebene erforderlich. Hierbei steht in Hinblick auf die Transaktionssicherheit im Vordergrund, dass entweder alle Aufgaben in der spezifizierten Reihenfolge ausgeführt werden oder die Prozessausführung keinerlei Effekte in dem betrachteten (verteilten) System verursacht [CYCX05, MM06, KWL09]. Um den Verlust bereits geleisteter Arbeitsschritte und das ineffiziente Sperren von Ressourcen über mehrere Ausführungseinheiten hinweg zu vermeiden, bieten sich vor allem *Forward-Recovery*-Mechanismen und optimistische Verfahren mit Kompensationsstrategien für die Fehlerbehebung bei der verteilten Ausführung von Prozessen an [CR04, MM06] (vgl. Abschnitt 2.4.5).

A_6 **Überwachung und Nachvollziehbarkeit der Prozessausführung:** Neben den zur Ausführung der fachlichen Prozessinhalte benötigten Funktionalitäten werden auch bei der verteilten Ausführung eines Prozesses administrative Funktionalitäten zur Überwachung und Analyse der durchgeführten Prozessinstanzen benötigt [HLGG00, CCSD04, BKK+04, WKK+10]. Da bei der organisationsübergreifenden Ausführung einer entfernt ablaufenden Prozesspartition in der Regel der direkte Einflussbereich des Prozessinitiators verlassen wird, bleiben jedoch auch die benötigten Informationen (wie z. B. der aktuelle Zustand bzw. Fortschritt der Prozessausführung oder die Ausführungsdauer der einzelnen Aktivitäten) verborgen [JNDV99, KWA99a, RSS06b, vRZ07]. Jeder Beteiligte kennt in diesem Fall also nur einen geringen Ausschnitt des gesamten Prozesses, so das es schwierig ist, einen Überblick über den gesamten Ablauf herzustellen [All05]. Über eine normale Analyse von Prozessdaten hinaus sind jedoch gerade bei einer verteilten Prozessausführung solche Daten von großer Relevanz, zum Beispiel als Grundlage für die Aufdeckung von Fehlern und Vertragsbrüchen, zur Abrechnung von Lei-

stungen zwischen den teilnehmenden Organisationen [HLGG00] oder für die Zuordnung von Verantwortlichkeiten im rechtlichen Sinne [WPH07]. Letztendlich spielt hierbei auch der Aufbau von Vertrauensbeziehungen eine große Rolle [CCSD04]. Es müssen somit geeignete Mechanismen zur Verfügung gestellt werden, um die für die Kooperation zwischen Organisationen wichtigen Zusammenhänge (d. h. ggf. auch auf feingranularer Ebene [HLGA01]) zu offenbaren, während auf der anderen Seite die Autonomie der beteiligten Parteien gewahrt bleibt [CCSD04, WPH07].

A_7 **Persistenz von prozessrelevanten Daten:** In der Regel ist ein persistenter Speicher erforderlich, um die zur Prozessausführung benötigten Daten zu speichern. Dies umfasst vor allem die relevanten Prozessmodell- und -instanzdaten, welche bis zum Zeitpunkt der Ausführung (bzw. einer Fortführung der Ausführung) bereitgestellt werden müssen. Instanzdaten müssen bis zur Beendigung der Prozessausführung gespeichert werden, um zur Behandlung von Fehlerfällen das Rücksetzen oder die Kompensation von Prozessschritten zu erlauben (vgl. Abschnitt 2.4.5). Im Fall einer Offline-Bearbeitung müssen die Ergebnisse der (Teil-)Prozessausführung in jedem Fall festgehalten werden, bis eine Synchronisation mit anderen Parteien der Prozessausführung (z. B. einem zentralen Server) möglich ist [AGK+95]. Gegebenenfalls müssen relevante Daten sogar nach der Ausführung des Prozesses weiterhin (lokal) aufgehoben werden, etwa um eine spätere Nachvollziehbarkeit des Prozesses zu erlauben (z. B. zu Abrechnungszwecken) oder um eine Evaluation über die Ausführung mehrerer Prozessinstanzen durchzuführen (z. B. zur Anfertigung von Statistiken).

A_8 **Vermeidung hohen Koordinationsaufwands:** Koordination umfasst im Allgemeinen die Maßnahmen, welche mehrere miteinander in Verbindung stehende Akteure zur Erreichung eines gemeinsamen Ziels durchführen, ein einzelner Teilnehmer zur Erreichung derselben Ziele jedoch nicht durchgeführt hätte [MT88]. Durch die Notwendigkeit der Verteilung des Prozesses und dem Austausch von Informationen zu Kontroll- und Datenfluss entsteht somit bei der verteilten Ausführung eines Prozesses zwangsläufig ein gewisser Koordinationsaufwand [JSH+01]. Dieser kann sich zum einen durch einen zusätzlichen Nachrichtenaustausch äußern, wodurch Recheneinheiten und Netzwerke zusätzlich belastet werden und ggf. auch monetäre Kosten anfallen können. Zum anderen kann sich die Prozessausführung um den für die Koordination erforderlichen Zeitaufwand verzögern. Um wahrnehmbare Verzögerungen bei der verteilten Prozessausführung zu vermeiden und diese möglichst kostengünstig durchzuführen, ist eine Minimierung des Koordinationsaufwands anzustreben [BD00, RB07]. Dies gilt insbesondere im Fall einer angestrebten Offline-Bearbeitung von Prozesspartitionen [Sch01].

A_9 **Sicherheit:** Bei einer zentralen Prozessausführung sind die erforderlichen Sicherheitsmaßnahmen zumeist auf die Authentifizierung und Autorisierung von (technischen und menschlichen) Ressourcen und menschlichen Systemadministratoren sowie auf Maßnahmen zur Sicherstellung von Vertraulichkeit und Integrität der zwischen der Ausführungseinheit und den Ressourcen zu übertragenen Daten beschränkt [CR97, WfM98b]. Im Rahmen solcher Maßnahmen wird in der Regel auch die Nachvollziehbarkeit der Verantwortlichkeit bzw. die Delegation von Verantwortlichkeiten für die Ausführung von Aktivitäten unterstützt [CRW98, WfM98b, JSH+01]. Bei der Nutzung einer ereignisbasierten Architektur kommen zudem die Erlaubnis und die Authentifizierung für das Erzeugen bzw. Verarbeiten von Ereignissen hinzu, welche die Prozessausführung beeinflussen können [CR97, JSH+01]. Im Kontext verteilt ausgeführter Prozesse und insbesondere bei organisationsübergreifenden Kooperationen muss diese Authentifizierung und Autorisierung entsprechend auf die autonomen Komponenten des verteilten Systems (d. h. auf die Organisationen mit ihren Ausführungsumgebungen) ausgedehnt werden [MM08]. Geschieht die Verteilung zur Laufzeit, so muss das unter Umständen von allen Teilnehmern verwendete Sicherheitskonzept die Delegation von Prozesspartitionen an andere Ausführungseinheiten unterstützen [CR97]. So muss einerseits – wie auch bei der zentralen Ausführung – sichergestellt werden können, dass keine Funktionalitäten einer Ausführungsumgebung durch böswillige Benutzer aufgerufen werden können und keine Ergebnisse von unbekannten und/oder böswilligen Quellen entgegengenommen werden können [CR97, KPF01, MM08]. Insbesondere ist der Schutz von Ausführungseinheiten vor nichtautorisierten oder sogar maliziösen Prozessen (z. B. Endlos-Prozessen) erforderlich. Auf der anderen Seite stellen bei einer Verteilung der Prozessausführung auch der Prozess selbst und seine Ausführungsdaten zu schützende Objekte dar, deren Vertraulichkeit und Integrität bei der Verteilung der Prozessbeschreibung auf mehrere Teilnehmer bzw. bei der Durchführung von Überwachungsmaßnahmen gewährleistet werden müssen [ACM01, MM08]. In beiden Fällen muss insbesondere die Autonomie der beteiligten Parteien berücksichtigt werden, welche ggf. in feingranularer Art und Weise über die Vergabe von Funktionalitäten und Informationen entscheiden wollen [SO04].

Tabelle 4.2 zeigt die beschriebenen Gruppen von Anforderungen mit ihren jeweiligen Teilanforderungen und -voraussetzungen. Dabei wurden nur diejenigen Anforderungen berücksichtigt, welche über die Anforderungen einer nicht verteilten Ausführung von Prozessen mittels eines einzelnen zentral gesteuerten Prozessmanagementsystems hinausgehen. Die Eigenschaften und Anforderungen individueller Systeme zur Ausführung prozessorientierter Anwendungen gelten hier natürlich entsprechend (vgl. Kapitel 2). Die speziellen Anforderungen, welche aus der Flexibilisierung verteilt ausgeführter Prozes-

Gruppe	Anforderung	Referenzen
A_1	Interoperabilität $A_{1.1}$ Syntax und Semantik des auszuführenden Prozesses $A_{1.2}$ Verteilung von Prozesspartitionen $A_{1.3}$ Übergabe des Kontrollflusses $A_{1.4}$ Austausch von Prozessdaten $A_{1.5}$ Bereitstellung und Sammlung von Daten über die Prozessausführung	(Hol95, JNDV99, vdAvH02, PC07, Wes07)
A_2	Autonomie beteiligter Parteien $A_{2.1}$ Dezentralität $A_{2.2}$ Beschränkung des Zugriffs auf Ressourcen, Funktionalitäten und Informationen	(KWA99a, SO04, CCSD04, Sch04a, Sch05, YY07, WW08)
A_3	Verteilungsstrategie und -durchführung $A_{3.1}$ Spezifikation von Rahmenbedingungen für die (verteilte) Prozessausführung $A_{3.2}$ Spezifikation von Zusammenhängen zw. Prozesspartitionen und Ausführungseinheiten $A_{3.3}$ Existenz und Bekanntheit bzw. Möglichkeit zum Auffinden von Ausführungseinheiten	(Sch01, JSH+01, KL06, Kun08, Wut10)
A_4	Korrektheit der Ausführung $A_{4.1}$ Einhaltung der spezifizierten Ausführungsreihenfolge und -bedingungen $A_{4.2}$ Kontrollierter Zugriff auf gemeinsame Daten $A_{4.3}$ Berücksichtigung organisatorischer oder technischer Vorgaben	(AGK+95, WW97, MWW+98, RRD03, Sch01, CR04)
A_5	Fehlererkennung und -behandlung $A_{5.1}$ Lokale Fehlererkennung und -behandlung $A_{5.2}$ Globale Fehlererkennung und -behandlung	(CR97, VGBA99, CR04, CYCX05, MM06, KWL09)
A_6	Überwachung und Nachvollziehbarkeit der Prozessausführung $A_{6.1}$ Bereitstellung und Sammlung von Daten über die Prozessausführung $A_{6.2}$ Zuordnung von Verantwortlichkeiten	(KWA99a, HLGA01, CCSD04, WPH07, vRZ07, WKK+10)
A_7	Persistenz von prozessrelevanten Daten $A_{7.1}$ Prozessmodelle $A_{7.2}$ Prozessinstanzdaten $A_{7.3}$ Prozesshistorien	(AGK+95)
A_8	Vermeidung hohen Koordinationsaufwands $A_{8.1}$ Minimierung des Nachrichtenaustausches $A_{8.2}$ Minimierung von Verzögerungen der (verteilten) Prozessausführung	(BD00, JSH+01, Sch01, RB07)
A_9	Sicherheit $A_{9.1}$ Autorisierung und Authentisierung von Ausführungseinheiten $A_{9.2}$ Verschlüsselung der Kommunikation zwischen Ausführungseinheiten $A_{9.3}$ Vertraulichkeit und Integrität der Prozessbeschreibung $A_{9.4}$ Delegation der Ausführung von Prozesspartitionen $A_{9.5}$ Verantwortlichkeit für die Ausführung von Prozesspartitionen	(CR97, WfM98b, ACM01, JSH+01, KPF01, SO04, MM08)

Tabelle 4.2: Allgemeine Anforderungen verteilt ausgeführter Prozesse

se resultieren, werden im folgenden Abschnitt identifiziert und im Detail beschrieben.

4.4.2 Anforderungen dynamisch verteilter Prozesse

Wie in den betrachteten Fallbeispielen (vgl. Abschnitt 4.2) gezeigt wurde, können verteilt ausgeführte Prozesse je nach Anwendungsgebiet und Zielsetzung der Verteilung einer großen Dynamik unterliegen. Die Anpassung der verteilten Prozessausführung als Reaktion auf relevante Umgebungsänderungen unterliegt dabei großen Herausforderungen. Zum einen muss Anpassungsbedarf anhand einer individuellen Beobachtung relevanter Kontextinformationen abgeleitet werden. Da dies bei einer verteilten Prozessausführung unter Umständen mehrere (potentiell) an der Pro-

zessausführung teilnehmende Systeme umfasst, gewinnt die zuvor als allgemeine Anforderung beschriebene Notwendigkeit zur Überwachung der Prozessausführung (Anforderung A_6) stark an Bedeutung. Zum anderen muss die Verteilungsstrategie (Anforderung A_3) geeignet sein, um spontane Anpassungen der Verteilungskonfiguration zu erlauben. Hinsichtlich der flexiblen Integration von mobilen Endgeräten sind schließlich die Offline-Bearbeitung von Prozesspartitionen und der Umgang mit Heterogenität zu berücksichtigen. Die folgende Zusammenstellung von Anforderungen zur Realisierung dynamisch verteilter Prozesse greift diese Besonderheiten auf und beschreibt die einzelnen Aspekte im Detail. Zur Unterscheidung zu den allgemeinen Anforderungen verteilt ausgeführter Prozesse (vgl. Abschnitt 4.4.1) sind die speziellen Anforderungen dynamisch verteilter Prozesse mit dem Buchstaben **D** gekennzeichnet:

D_1 **Umgang mit Unvorhersehbarkeit:** Die Betrachtung der Fallbeispiele (vgl. Abschnitte 4.2.1 bis 4.2.6) und der allgemeinen Eigenschaften verteilt ausgeführter Prozesse (vgl. Abschnitt 4.3.2) zeigt, dass die Verteilung eines Prozesses uneingeschränkt über seinen ganzen Lebenszyklus und unabhängig von einem konkreten Verteilungszeitpunkt möglich sein sollte. Über die Notwendigkeit sowie über die Art und Weise einer Verteilung kann dabei unter Umständen erst entschieden werden, wenn ein Prozessmodell bereits instantiiert wurde und die betreffende Prozessinstanz schon (zentral) ausgeführt wird. In diesem Fall muss die Verteilung des Prozesses auf Basis der laufenden Prozessinstanz geschehen. Aufgrund der Unvorhersehbarkeit muss hierbei davon ausgegangen werden, dass die Prozessbeschreibung vorab keine speziellen Informationen bezüglich der Verteilung enthält.

D_2 **Maximales Verteilungsspektrum:** Abhängig von den möglichen unterschiedlichen Zielen der Verteilung (vgl. Abschnitt 4.3.1) müssen verschiedene Verteilungskonfigurationen unterstützt werden, so dass ein möglichst großes Verteilungsspektrum zur Verfügung steht. Basierend auf der Identifikation wichtiger Eigenschaften verteilt ausgeführter Prozesse (vgl. Abschnitt 4.3.2) umfasst dies insbesondere die dynamische Festlegung der *Granularität* und die Bestimmung und Zuordnung der entsprechenden *Zielorte* der Verteilung (vgl. [Sch01]). Es sollte somit die Granularität der Verteilung (einzelne Aktivität oder ganzer Prozess) zu jedem Zeitpunkt frei wählbar sein. Es sollten zudem alle möglichen Bindungszeitpunkte berücksichtigt werden können, also zum Beispiel sowohl die längerfristige Reservierung als auch der spontane Austausch von Ausführungseinheiten (vgl. Abschnitt 2.7.3).

D_3 **Hohe Individualität:** Für einzelne Prozessinstanzen sollte eine jeweils unterschiedliche Verteilung möglich sein, um eine individuelle Anpassung der Prozessausführung anhand aktueller Rahmenbedingungen zu erlauben. Hierzu gehören, in Abhängigkeit der Zielsetzung der Vertei-

lung, die Eingabedaten der jeweiligen Prozessinstanz, die Zustände von Ressourcen und Ausführungseinheiten sowie die speziellen Interessen und Vorgaben des Prozessinitiators und der an der Prozessausführung beteiligten menschlichen Akteure (vgl. Abschnitte 4.2.1 bis 4.2.6 und Abschnitt 4.3.2).

D_4 **Zeitnahe Bereitstellung und Erhebung von Informationen:** Um Anpassungsbedarf auf der Basis von aktuellen Kontextdaten (vgl. Abschnitt 3.5.2) abzuleiten und für eine angemessene Anpassung der Prozessausführung nutzbar zu machen, ist eine möglichst zeitnahe Erhebung von Informationen erforderlich [CR97, MRD08] (vgl. Abschnitte 3.2, 3.5 und 3.7). Dabei ist wesentlich, dass die jeweils individuell relevanten Informationen [CCSD04] zu Prozessmodellen, Prozessinstanzen, Ausführungseinheiten, Ressourcen sowie dem sonstigen externen Kontext im Anwendungszusammenhang für eine Verarbeitung zugänglich gemacht werden können (vgl. im Speziellen Abschnitt 3.5.2). Hierfür werden verschiedene Strategien zur kontinuierlichen Überwachung ebenso benötigt wie Möglichkeiten zur spontanen Abfrage einzelner Details im konkreten Bedarfsfall. Dies entspricht dem (ereignisbasierten) *Push-Modell* bzw. dem *Pull-Modell* zum reaktiven Bezug von Monitoring-Daten [HLGG00].

D_5 **Proaktivität:** Im Rahmen der Flexibilitätsbetrachtung soziotechnischer Systeme in Abschnitt 3.2 wurde die Verhaltensdimension der Flexibilität durch die Möglichkeiten eines Systems zu reaktivem und proaktivem Verhalten bestimmt. Um proaktive Anpassungen auf vorhersehbare Umgebungsänderungen in der nahen Zukunft zu ermöglichen, sind geeignete Mittel zur Integration von Prognosen zukünftiger Entwicklungen erforderlich (vgl. Abschnitt 3.5.6).

D_6 **Steuerung der Prozessausführung:** Um auf eine laufende Prozessinstanz in Hinblick auf die durchzuführenden Änderungen Einfluss nehmen zu können, sind geeignete Maßnahmen zur Steuerung der Prozessausführung erforderlich. Im Kontext verteilt ausgeführter Prozesse ist es dabei unter Umständen notwendig, dass diese Steuerungsfunktionalitäten auch anderen Parteien bereitgestellt werden, welche an der Prozessausführung beteiligt sind, insbesondere, wenn Prozesspartitionen im Auftrag einer solchen Partei entfernt ausgeführt werden. Ein Beispiel hierfür ist die Ausführung von Prozessen im Rahmen des *Process-as-a-Service*-Modells (vgl. Abschnitt 4.2.6).

D_7 **Dezentralität:** Es sollte die Möglichkeit zu einer dezentralen Koordination der verteilten Prozessausführung gegeben sein, sofern die Dezentralität Zielsetzung der Verteilung ist (vgl. Abschnitt 4.3.1), eine Rahmenbedingung für die Kooperation zwischen autonomen Organisationen darstellt oder durch die Beschränkungen realer technischer Infrastrukturen entsteht (vgl. auch Anforderung $A_{2.1}$). Beispiele sind durch die or-

ganisationsübergreifenden Szenarien im E-Government (vgl. Abschnitt 4.2.2) bzw. die Kooperation von Ausführungseinheiten in mobilen Adhoc-Netzwerken (vgl. Abschnitt 4.2.4) gegeben. Weitere Gründe sind in typischen nicht-funktionalen Aspekten wie Skalierbarkeit oder Ausfallsicherheit zu finden (vgl. Abschnitt 4.2.5).

D_8 **Unterstützung mobiler Infrastrukturen:** Ist ein Zielort der Verteilung nicht durch eine fixe Infrastruktur und eine dauerhafte Konnektivität gekennzeichnet, so sind die damit verbundenen Besonderheiten für die Prozessausführung zu berücksichtigen. Dies umfasst insbesondere die eingeschränkten Ressourcen mobiler Geräte in Hinblick auf Speicher, Rechenleistung und Energie sowie deren Heterogenität in Bezug auf Netzwerkverbindungen, Diensttechnologien und Benutzungsschnittstellen [PC07, Kun08]. Als wichtigstes Merkmal steht dabei im Vordergrund, dass Prozesspartitionen offline bzw. in einem lokal begrenzten Netzwerk ausgeführt werden können (vgl. Abschnitte 4.2.3 und 4.2.4). Durch die Integration mobiler Infrastrukturen sollte es zudem nicht zu Einschränkungen für stationäre Komponenten kommen [Sch01].

D_9 **Anpassung an heterogene Infrastrukturen:** Im Fall einer dynamischen Verteilung ist eine besondere Berücksichtigung von heterogenen Ausführungsumgebungen erforderlich, da im Vorfeld unter Umständen nicht bekannt ist, welche Ausführungseinheit welche Partition des Prozesses ausführen und somit für den Aufruf von Ressourcen, die Übergabe und Entgegennahme von Daten und die Präsentation von Benutzungsschnittstellen verantwortlich sein wird. Es ist daher eine spontane Anpassung an die vorherrschenden Bedingungen im Rahmen der vorgegebenen Interoperabilität erforderlich (vgl. Abschnitte 4.2.1 bis 4.2.4).

D_{10} **Berücksichtigung benutzerdefinierter Vorgaben:** Zur Entwicklungszeit festgelegte Verteilungsabsichten und zur Laufzeit notwendige Verteilungen sollten miteinander kombinierbar sein. Hierzu ist die Definition von benutzerdefinierten Vorgaben erforderlich. Unter einem Benutzer kann dabei sowohl der Modellierer des fachlichen Prozessmodells als auch der Initiator einer oder mehrerer Prozessinstanzen verstanden werden. Im Fall einer dienstorientierten Architektur ist in der Regel der Dienstnutzer der Initiator des Prozesses. Zudem können auch die direkt an der Ausführung eines Prozesses beteiligten Personen (*Prozessteilnehmer*) kurzfristig Vorgaben für die Verteilung des Prozesses machen. Ein Beispiel ist die gewünschte Übertragung der von einer bestimmten Person auszuführenden Prozesspartition an ein mobiles Gerät. Benutzerdefinierte Vorgaben können dabei alle funktionalen und nicht-funktionalen Ziele der Prozessausführung umfassen (vgl. Abschnitt 4.3.1).

D_{11} **Nicht-funktionale Aspekte:** Die notwendigen Maßnahmen zur Flexibilisierung der verteilten Prozessausführung sollten möglichst wenig Aufwand verursachen (vgl. Abschnitt 3.1) und die Ausführung eines Prozes-

ses hinsichtlich nicht-funktionaler Kriterien nicht negativ beeinflussen. Dies bedeutet insbesondere, dass ein zentral ausgeführter Prozess nicht durch die bloße Möglichkeit einer (potentiellen) Verteilung negativ beeinträchtigt werden darf. Der zur Anpassung durchzuführende Koordinationsaufwand sollte möglichst wenig negative Auswirkungen auf das Kommunikationsnetzwerk sowie auf die Auslastung und Verfügbarkeit von Ressourcen und Ausführungseinheiten haben [CR97]. Die Skalierbarkeit hinsichtlich Umfang und Komplexität der Prozessbeschreibung, die Anzahl der gleichzeitig laufenden Prozessinstanzen sowie die Größe und Anzahl der zu übertragenden Datenobjekte sollte nicht negativ beeinflusst werden [CR97, Sch04a, Sch05]. Die zuvor genannten Anforderungen gelten insbesondere, wenn das Ziel der Anpassung in einer Optimierung von (denselben) nicht-funktionalen Aspekten besteht (vgl. Abschnitt 4.2.5).

D_{12} **Trennung fachlicher und technischer Logik:** Die Betrachtung klassischer Verteilungsmodelle zeigt, dass eine statische Integration von Verteilungslogik in das Prozessmodell die Flexibilität der Verteilung begrenzt (vgl. Abschnitt 4.3.3). Um zudem die Handhabbarkeit und Wartbarkeit von Prozessmodellen nicht zu beeinträchtigen, sollte nach Möglichkeit eine strikte Trennung von fachlicher und technischer Logik erfolgen [DGH03]. Dies trifft auf Maßnahmen zur Verteilung, aber auch auf Maßnahmen zur Überwachung, zur Steuerung und zur Integration von technischen Anpassungen der Prozessausführung zu [MRD08].

D_{13} **Fortwährende Anpassbarkeit der Organisationsstruktur:** Die Verteilung der Prozessausführung sollte den Umgang mit Änderungen der Organisationsstruktur unterstützen bzw. die Anpassbarkeit der Organisationsstruktur, wie die Erweiterung oder Reduktion von Ressourcen oder Rollen bzw. deren Zuordnung zu anderen oder neuen Organisationen bzw. Organisationseinheiten nicht beeinflussen [CR97]. Im Speziellen bedeutet dies, dass das Verteilungsmodell und die unterliegende Softwarearchitektur stets die Integration neuer und bisher unbekannter Dienste und Ausführungseinheiten zulassen muss und den Teilnehmern erlauben soll, aktuelle Partnerschaften zu verlassen und sich neuen Partnerschaften anzuschließen zu können [Sch04a, Sch05]. Dies entspricht der Flexibilität klassischer prozessorientierter Anwendungen (vgl. Abschnitte 2.7.3 und 3.3.2) und insbesondere der durch die Nutzung dienstorientierter Architekturen erzielten Flexibilität (vgl. Abschnitt 3.4).

D_{14} **Fachliche Evolution von Prozessmodellen:** Die Art und Weise der Modellierung von fachlichen Prozessmodellen und die Prozessmodelle selbst sollten nach Möglichkeit nicht verändert werden, um die Wiederverwendung, Wartung und Erweiterbarkeit von bestehenden Prozessmodellen zu erlauben bzw. nicht zu erschweren [MWL08]. Die durch den Lebenszyklus allgemeiner prozessorientierter Anwendungen bereitgestellte Fähigkeit zur beliebigen (inhaltlichen) Anpassung eines Prozesses im

Gruppe	Anforderung	Referenz
$\mathbf{D_1}$	Umgang mit Unvorhersehbarkeit $D_{1.1}$ Unabhängigkeit von feststehenden Verteilungszeitpunkten $D_{1.2}$ Spontane Verteilung von zentral ausgeführten Prozessen $D_{1.3}$ Partitionierung zur Laufzeit	Abschnitte 4.2.1 bis 4.2.6, Abschnitt 4.3.2
$\mathbf{D_2}$	Maximales Verteilungsspektrum $D_{2.1}$ Beliebige Granularität $D_{2.2}$ Dynamische Bestimmung der Zielorte der Verteilung $D_{2.3}$ Variable Bindungszeitpunkte	Abschnitte 2.7.3, 4.2.1 bis 4.2.6, 4.3.1 und 4.3.2
$\mathbf{D_3}$	Hohe Individualität $D_{3.1}$ Unterschiedliche Verteilung für einzelne Prozessinstanzen	Abschnitte 4.2.1 bis 4.2.6, Abschnitt 4.3.2
$\mathbf{D_4}$	Zeitnahe Bereitstellung und Erhebung von Informationen $D_{4.1}$ Kontinuierliche Überwachung relevanter Informationen (*Push-Modell*) $D_{4.2}$ Individuelle Abfrage einzelner Details im Bedarfsfall (*Pull-Modell*)	Abschnitte 3.2, 3.5 und 3.7; (HLGG00, CCSD04)
$\mathbf{D_5}$	Proaktives Verhalten $D_{5.1}$ Integration von Vorhersagen zukünftiger Entwicklungen	Abschnitte 3.2 und 3.5
$\mathbf{D_6}$	Steuerung der Prozessausführung $D_{6.1}$ Bereitstellung von Management-Funktionalitäten	Abschnitt 4.2.6
$\mathbf{D_7}$	Dezentralität $D_{7.1}$ (Potentielle) dezentrale Koordination der verteilten Ausführung	Abschnitte 4.2.2, 4.2.4, 4.2.5 und 4.3.1
$\mathbf{D_8}$	Unterstützung mobiler Infrastrukturen $D_{8.1}$ Eingeschränkte Ressourcen $D_{8.2}$ Offline-Bearbeitung von Prozesspartitionen $D_{8.3}$ Keine Beeinträchtigung stationärer Komponenten	Abschnitte 4.2.3 und 4.2.4; (Sch01, PC07, Kun08)
$\mathbf{D_9}$	Anpassung an heterogene Infrastrukturen $D_{9.1}$ Aufruf von Ressourcen $D_{9.2}$ Ein- und Ausgabedaten $D_{9.3}$ Benutzungsschnittstellen	Abschnitte 4.2.1 bis 4.2.4
$\mathbf{D_{10}}$	Berücksichtigung benutzerdefinierter Vorgaben $D_{10.1}$ Vorgaben des Prozessmodellierers $D_{10.2}$ Vorgaben des Prozessinitiators bzw. Dienstnutzers $D_{10.3}$ Vorgaben von Prozessteilnehmern	Abschnitt 4.3.1
$\mathbf{D_{11}}$	Nicht-funktionale Aspekte $D_{11.1}$ Keine Beeinflussung zentral ausgeführter Prozesse $D_{11.2}$ Minimierung des zur Anpassung durchzuführende Koordinationsaufwands $D_{11.3}$ Keine negativen Auswirkungen auf Skalierbarkeit von Prozessen und Systemen	Abschnitte 3.1 und 4.2.5; (CR97, BD00, Sch04a, Sch05)
$\mathbf{D_{12}}$	Trennung fachlicher und technischer Logik $D_{12.1}$ Verteilungsinformationen $D_{12.2}$ Maßnahmen zur Überwachung $D_{12.3}$ Technische Anpassungen	Abschnitt 4.3.3; (DGH03, MWL08, MRD08)
$\mathbf{D_{13}}$	Fortwährende Anpassbarkeit der Organisationsstruktur $D_{13.1}$ Integration neuer und bisher unbekannter Dienste und Ausführungseinheiten $D_{13.2}$ Reduktion bzw. Austreten bisher bekannter Dienste und Ausführungseinheiten	Abschnitte 2.7.3, 3.3.2 und 3.4; (CR97, Sch04a, Sch05)
$\mathbf{D_{14}}$	Erhalt der fachlichen Evolution von Prozessmodellen $D_{14.1}$ Erweiterbarkeit von bestehenden Prozessmodellen	Abschnitt 2.8
$\mathbf{D_{15}}$	Erhalt der fachlichen Anpassbarkeit laufender Prozessinstanzen $D_{15.1}$ Inhaltliche Anpassbarkeit auf Basis bestehender Konzepte	Abschnitt 3.3.1; (RB07)
$\mathbf{D_{16}}$	Automatisierbarkeit des gesamten Anpassungsvorgangs $D_{16.1}$ Erhebung und Verarbeitung von Informationen $D_{16.2}$ Ableitung von Anpassungsbedarf $D_{16.3}$ Durchführung der Anpassung (Verteilung)	Abschnitte 3.6, 3.8 und 3.9

Tabelle 4.3: Anforderungen dynamisch verteilt ausgeführter Prozesse

Sinne einer Evolution des Prozessmodells (vgl. Abschnitte 2.8 und 3.3.1) sollte nicht beeinträchtigt werden [CR97].

D_{15} **Fachliche Anpassbarkeit laufender Prozessinstanzen:** Nach Möglichkeit sollte auch eine inhaltliche Anpassung dynamisch verteilt ausgeführter Prozessinstanzen auf Basis bestehender Konzepte nicht ausgeschlossen werden [RB07] (vgl. Abschnitt 3.3.1).

D_{16} **Automatisierbarkeit des Anpassungsvorgangs:** Wie in den Abschnitten 3.6, 3.8 und 3.9 gezeigt wurde, sollten die für eine Flexibilisierung erforderlichen Teilvorgänge der Erhebung und Verarbeitung von aktuellen Informationen, die Ableitung von Anpassungsbedarf sowie die Durchführung der Anpassungen nach Möglichkeit automatisierbar sein, um den (Zeit-)Aufwand für die Anpassung möglichst gering zu halten.

Tabelle 4.3 fasst die genannten Anforderungen zusammen, welche speziell die Flexibilisierung prozessorientierter Systeme adressieren, um eine dynamisch verteilte Ausführung von Prozessen zu unterstützen. Dabei sind die Anforderungen in Gruppen mit ihren jeweiligen Teilanforderungen und -voraussetzungen gegliedert. Auf Basis dieser Anforderungen kann im Folgenden ein idealtypisches Lebenszyklusmodell abgeleitet werden, welches die Entwicklung, Ausführung und Anpassung prozessorientierter Anwendungen auch in Bezug auf eine dynamische Verteilung der Prozessausführung unterstützt. Die Erweiterung des zur fachlichen Evolution von Prozessmodellen zugrunde gelegten Lebenszyklusmodells ist Inhalt des nächsten Abschnitts.

4.4.3 Erweiterung des Prozesslebenszyklus

Aus der Betrachtung der oben genannten Fallstudien, der Ziele und Eigenschaften verteilt ausgeführter Prozesse sowie der Flexibilität klassischer Verteilungsmodelle lässt sich ableiten, welche Auswirkungen eine dynamische Verteilung auf den Lebenszyklus prozessorientierter Anwendungen haben kann. Abbildung 4.13 zeigt das um die gesammelten Erkenntnisse erweiterte Lebenszyklusmodell im Vergleich zum Lebenszyklus klassischer nichtverteilter Prozesse in Abbildung 2.25. Während das Lebenszyklusmodell klassischer (Geschäfts-)Prozesse auf der technischen Ebene durch relativ stark voneinander isolierte sequenzielle Phasen gekennzeichnet ist, verlangt die Möglichkeit einer für verschiedene Prozessinstanzen individuell dynamisch verteilten Ausführung vor allem stärkere Interdependenzen zwischen den einzelnen Lebenszyklusphasen zur Laufzeit des Prozesses, um hinsichtlich der Verteilung flexible Reaktionen auf unvorhergesehene Umgebungsänderungen zu ermöglichen [ZKL09b, ZKL09a].

Solange eine Verteilung zur Entwicklungszeit des Prozesses nicht explizit vorgesehen ist, sollte die fachliche Ebene von der *Identifikation und Analyse des Anwendungsfalls* über die Erstellung eines strategischen bzw. operationalen Prozessmodells (*fachliche Modellierung*) bis hin zur entsprechenden

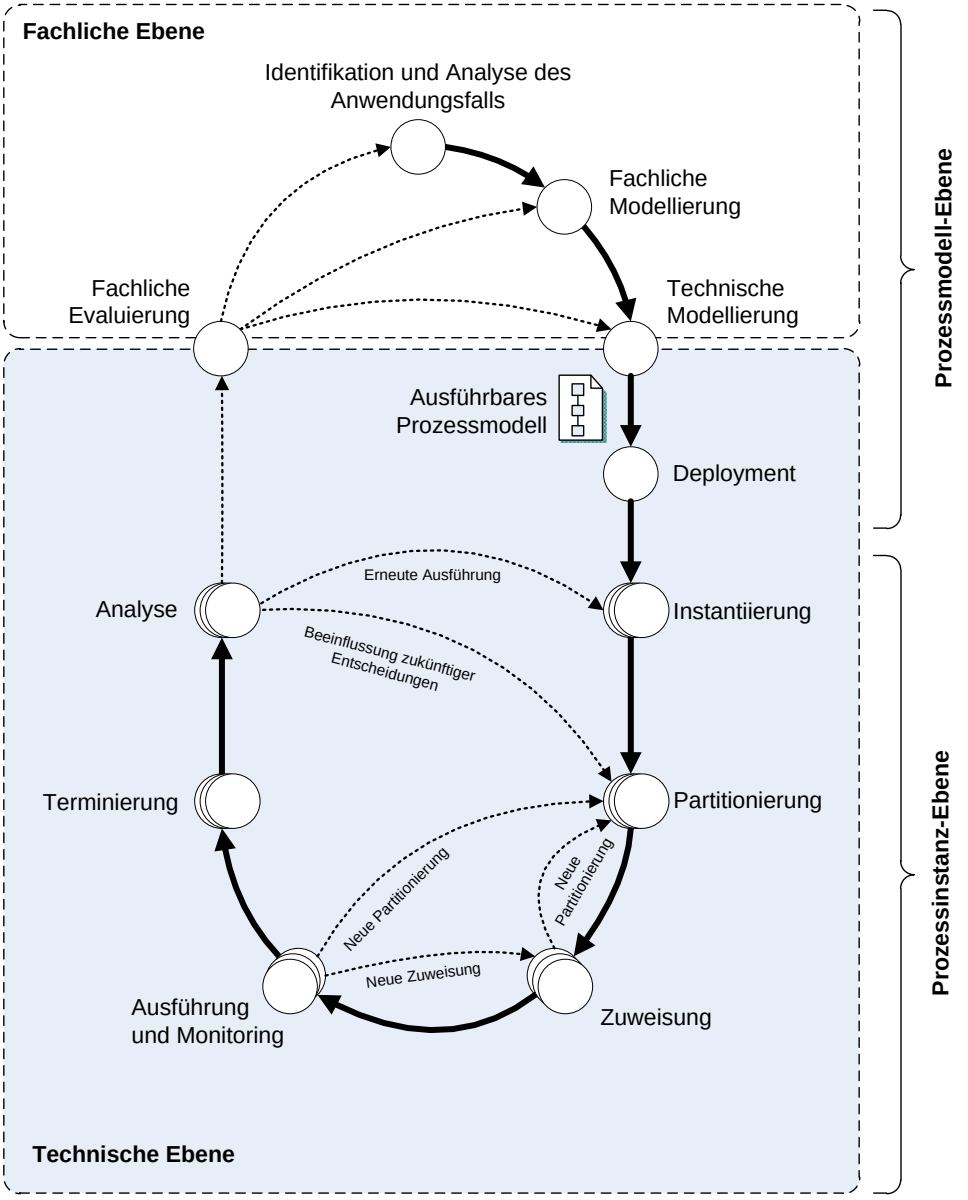


Abbildung 4.13: Lebenszyklusmodell dynamisch verteilt ausgeführter Prozesse (ZKL09b, ZKL09a)

Übersetzung in ein ausführbares technisches Prozessmodell (*technische Modellierung*) von einer Verteilung in technischer Hinsicht möglichst unberührt bleiben. Dies erlaubt insbesondere eine weitergehende Verwendung bekannter und etablierter Werkzeuge zur Entwicklung prozessorientierter Anwendungen auf den verschiedenen Modellierungsebenen (vgl. Abschnitt 2.3) sowie die Nutzung regulärer bzw. standardisierter Prozessbeschreibungssprachen als wichtige Basis für die Interoperabilität zwischen mehreren Ausführungseinheiten zur Laufzeit (vgl. Anforderung A_1). Zudem können bereits bestehende Prozessmodelle oder allgemeine Vorlagen zur Erstellung von neuen Prozessmodellen weiterverwendet werden. Sollten bereits zur Entwicklungszeit fachlich motivierte Rahmenbedingungen für die Verteilung der Prozessausführung be-

stehen, so sollten diese jedoch als Vorgaben des Prozessmodellierers bei der Ausführung des Prozesses geeignet berücksichtigt werden können (vgl. Anforderung $D_{10.1}$). Sofern die verwendete Prozessbeschreibungssprache die Definition von Verteilungsinformationen unterstützt (wie zum Beispiel die Sprachen *XPDL* oder *BPMN*, vgl. Abschnitt 5.2.4), so muss zudem in Betracht gezogen werden, dass diese auch in (bestehenden) Prozessmodellen enthalten sein können.

Das Ergebnis der Modellierung sollte in jedem Fall ein *ausführbares Prozessmodell* sein, welches in einer bestimmten (lokalen oder entfernten) Ausführungsumgebung mit einer für die gewählte Prozessbeschreibungssprache geeigneten Prozess-Engine installiert werden kann (*Deployment*). Der Ablauf sollte hierbei möglichst weitestgehend dem Deployment eines zentral auszuführenden Prozesses entsprechen, um auch die Anpassung von ursprünglich zentral auszuführenden Prozessen zu unterstützen (vgl. Anforderung D_1).

Durch die explizite *Instantiierung* des Prozessmodells bzw. den Aufruf des entsprechenden Dienstes durch einen Dienstnutzer sollte eine neue Prozessinstanz entstehen, welche zunächst zentral durch die gewählte Ausführungseinheit verwaltet werden kann. Um die Verteilung auf der Ebene der Prozessinstanzen individuell zu beeinflussen (vgl. Anforderung D_3), sollten an dieser Stelle jedoch Vorgaben und Rahmenbedingungen des Prozessinitiators berücksichtigt werden können. Diese sollten dabei explizit durch den Prozessinitiator vorgegeben oder durch individuelle Vereinbarungen der Dienstnutzung abgeleitet werden können. Besteht zum Beispiel eine individuelle Dienstgütevereinbarung über die Ausführungsdauer eines (komplexen) Dienstes, so ist – falls notwendig – eine Verteilung der Prozessausführung hinsichtlich dieses Kriteriums vorzunehmen (vgl. Anforderung $D_{10.2}$).

Soll der Prozess zentral ausgeführt werden, so sollte der Lebenszyklus der Prozessinstanz direkt mit der Phase *Ausführung und Monitoring* auf der lokalen Prozess-Engine fortgesetzt (vgl. Anforderung $D_{11.1}$) und somit das klassische Lebenszyklusmodell nicht-verteilt ausgeführter Prozesse durchlaufen werden können (vgl. Abbildung 2.25). Andernfalls sollte eine Verteilung des Prozesses geprüft werden können. Der Vorgang der Verteilung setzt sich aus den Phasen der *Partitionierung* des Prozesses (also der Bestimmung der Granularität der Verteilung) und der *Zuweisung* der resultierenden Prozesspartitionen an ausgewählte Ausführungseinheiten zusammen (vgl. Abschnitt 4.3.2 und Anforderung D_2). Bei beiden Schritten sind – falls vorhanden – die zuvor spezifizierten Rahmenbedingungen für die Verteilung zu berücksichtigen. Zudem kann aufgrund der Dynamik der Systemumgebung die Partitionierung des Prozesses auch von der Verfügbarkeit oder der Leistungsfähigkeit geeigneter Ausführungseinheiten beeinflusst werden (vgl. z. B. Anforderung D_8). In diesem Fall ist gegebenenfalls eine neue Partitionierung des Prozesses in Abhängigkeit der bereits getroffenen oder der angestrebten Zuweisung erforderlich. Ein anderer Aspekt hierbei ist die Realisierung einer frühen Bindung von Ausführungseinheiten (vgl. Anforderung $D_{2.3}$), zum Beispiel um die erforderlichen Kapazitäten der gewählten Teilnehmer zu reservie-

ren und so später die Ausführungsgeschwindigkeit zu erhöhen. Falls sich der Ausführungskontext jedoch in der Zwischenzeit ändert, müssen die Reservierungen ggf. zurückgenommen und die Folge der Ausführungseinheiten neu berechnet werden. Der Lebenszyklus wird dazu (für die verbleibenden Teilschritte des Prozesses) in die Phase *Partitionierung* oder *Zuweisung* zurückgesetzt [ZKL09b, ZKL09a].

Die Phase *Ausführung und Monitoring* umfasst zum einen die zentrale Navigation eines (Teil-)Prozesses sowie die Ausführung seiner Aktivitäten. Hierbei werden Softwareanwendungen bzw. Dienste von der für die lokale Ausführung verantwortlichen Prozess-Engine aufgerufen oder dem Benutzer entsprechende Schnittstellen zur Interaktion mit dem Prozess angezeigt und dessen Eingaben entgegengenommen (vgl. Anforderung D_9). Dabei kann eine Verteilung des Prozesses durch die dynamische organisatorische Zuordnung von Ressourcen zu Ausführungseinheiten oder durch die Prozessausführung selbst notwendig werden, zum Beispiel aufgrund von aktuellen Eingabedaten (vgl. Abschnitt 4.2.2). Zum anderen sollte die Prozessausführung in dieser Phase parallel durch geeignete Monitoring-Maßnahmen überwacht werden (vgl. Anforderungen A_6 und D_4). Kommt es zu Abweichungen des erwarteten Verhaltens, so sollte in Abhängigkeit des aufgetretenen Problems ein Rücksprung zur Phase der *Zuweisung* oder der *Partitionierung* möglich sein, um entweder eine Kompensation bereits ausgeführter Prozessschritte einzuleiten, die Ausführung einer Aktivität oder einer Prozesspartition (ggf. durch eine andere Ausführungseinheit) zu wiederholen oder für Folgeaktivitäten eine neue Verteilung zu bestimmen (vgl. Anforderungen A_4 und A_5).

Die letzte hier betrachtete Phase des beschriebenen Lebenszyklus erlaubt eine (technische) *Analyse* der ausgeführten Aktivitäten einer terminierten Prozessinstanz, z. B. um einen Soll-Ist-Vergleich durchzuführen oder Informationen über Prozessteilnehmer bzw. Ausführungseinheiten zu verdichten. Die dabei gewonnenen Erkenntnisse können einerseits wertvolle Erfahrungswerte für die Ausführung zukünftiger Prozessinstanzen darstellen, z. B. um die Verteilung des Prozesses in Hinblick auf festgelegte Kriterien zu optimieren oder unzuverlässige Ausführungseinheiten zukünftig von der Ausführung von Prozessen auszuschließen. Andererseits kann die Analyse einzelner Prozessinstanzen analog zu einer nicht-verteilten Prozessausführung Anhaltspunkte für eine strukturelle oder sogar fachliche Anpassung des Prozessmodells liefern und somit in die *Evaluierung* des Prozesses auf fachlicher Ebene einfließen (vgl. Anforderung A_6).

4.5 Zusammenfassung

Die verteilte Ausführung von Prozessen stellt eine besondere Art der Verteilung prozessorientierter Anwendungen dar, bei welcher mehrere Ausführungseinheiten (in der Regel in Form von *Prozess-Engines*) für die Bearbeitung des Kontrollflusses und den Aufruf von Ressourcen zur Ausführung des Prozesses verantwortlich sind. Die Motivation für eine solche verteilte

Ausführung ist oftmals durch die Abbildung von in der Realität vorherrschenden Organisationsstrukturen gegeben, wie zum Beispiel durch die technische Unterstützung von unternehmens- bzw. organisationsübergreifenden Kooperationen. Aber auch die zunehmende Integration mobiler Geräte, die Relevanz des Ausführungskontextes oder die Optimierung nicht-funktionaler Kriterien machen eine Verteilung der Prozessausführung über mehrere Systeme vorteilhaft oder notwendig. In den genannten Anwendungsdomänen lässt sich jedoch ebenfalls eine sehr hohe Dynamik beobachten, welche eine flexible Anpassung der Verteilungskonfiguration insbesondere zur Laufzeit des Prozesses notwendig macht. Nach einer Erarbeitung der speziellen Anforderungen solcher *dynamisch verteilter Prozesse* wurde daher in diesem Abschnitt das Lebenszyklusmodell prozessorientierter Anwendungen auf der technischen Ebene um die Details einer dynamischen Verteilung zur Laufzeit einzelner Prozessinstanzen und die entsprechenden Interdependenzen zwischen den Lebenszyklusphasen erweitert.

Betrachtet man die Verteilung als Anpassungsgegenstand, so kann basierend auf der vorangegangenen Untersuchung der Grundlagen prozessorientierter Anwendungen (vgl. Kapitel 2) und ihrer Flexibilisierung (vgl. Kapitel 3) als Zwischenergebnis dieses Kapitels der Arbeit abgeleitet werden, dass für eine dynamisch verteilte Prozessausführung zwei wesentliche Teilaspekte im Vordergrund stehen:

- **Flexibilisierung der Verteilung:** Um eine Anpassung der Verteilungskonfiguration in Bezug auf Verteilungszeitpunkt, Verteilungsgranularität und Zielort der Verteilung individuell für verschiedene Prozessinstanzen vorzunehmen, wird ein Verteilungsansatz benötigt, welcher die dynamische Partitionierung einer Prozessinstanz und die Zuweisung der resultierenden Prozesspartitionen zu Ausführungseinheiten während der Laufzeit des Prozesses unterstützt.

- **Interoperabilität und Verteilungstransparenz:** Während die Verteilung der Prozessausführung für die Anwendungsebene weitestgehend transparent gehalten werden kann, ist die zeitnahe Ableitung von Anpassungsbedarf stark von der Bereitstellung, Erhebung und Verarbeitung relevanter Informationen sowie deren Austausch zwischen den potentiell an der Prozessausführung beteiligten Systemen abhängig. Eine Anpassung der Prozessausführung in Bezug auf die Verteilung ist zudem nur möglich, falls entsprechende Steuerungsfunktionalitäten zu deren Durchführung zur Verfügung stehen. Um solche fachlichen und technischen Belange der Prozessausführung angemessen zu trennen, wird eine geeignete *Middleware-Unterstützung* benötigt, welche ein ausreichendes Maß an Interoperabilität zwischen den teilnehmenden Systemen bereitstellt und die Komplexität des Anpassungsvorgangs vor Anwendungsentwicklern (hier insbesondere Prozessmodellierern) und Benutzern (hier Prozessteilnehmern) verbirgt.

Hinsichtlich der Definition von Flexibilität (vgl. Abschnitt 3.1) sollte der Anpassungsaufwand an Prozessen bzw. an ausführenden Systemen nach Möglichkeit minimiert werden und der Anpassungsvorgang selbst weitestgehend automatisierbar sein. Zudem müssen die sich aus dem Kontext dynamisch verteilt ausgeführter Prozesse ergebenen Rahmenbedingungen wie die potentielle Mobilität von Ausführungseinheiten, die Autonomie teilnehmender Organisationen, die Umsetzung benutzerdefinierter Vorgaben sowie die Gewährleistung der Korrektheit, der Sicherheit und der Benutzerfreundlichkeit prozessorientierter Anwendungen in einem Konzept zur Flexibilisierung verteilter Prozessausführung angemessen berücksichtigt werden. Im folgenden Kapitel wird der Stand der Forschung verteilt ausgeführter Prozesse in Hinblick auf die hier erarbeiteten Anforderungen genauer untersucht.

5 Stand der Forschung und verwandte Arbeiten

In den vergangenen Jahren ist bereits eine Vielzahl von Arbeiten über verteilt ausgeführte Prozesse im Allgemeinen und deren Flexibilisierung im Speziellen hervorgegangen, deren Ergebnisse für diese Arbeit berücksichtigt und mit den hier gesammelten Anforderungen dynamisch verteilter Prozesse abgeglichen werden sollen. In diesem Kapitel wird daher der für eine Anpassung verteilt ausgeführter Prozesse relevante Stand der Forschung untersucht und eine Abgrenzung der Inhalte dieser Arbeit zu verwandten Arbeiten vorgenommen. In Abschnitt 5.1 wird zunächst die Vorgehensweise und die Auswahl der untersuchten Ansätze motiviert. Basierend auf den Ergebnissen von Kapitel 3 und der grundlegenden vorangegangenen Analyse dynamisch verteilt ausgeführter Prozesse in Kapitel 4 sind diese in Ansätze und Möglichkeiten zur Interoperabilisierung des Prozessmanagements (vgl. Abschnitt 5.2), verschiedene Möglichkeiten zur Verteilung von Prozessen (vgl. Abschnitte 5.3 bis 5.6), zur Überwachung und Steuerung verteilt ausgeführter Prozesse (vgl. Abschnitt 5.7) sowie zur Integration und Darstellung geeigneter Benutzungsschnittstellen (vgl. Abschnitt 5.8) gegliedert. Das Kapitel schließt mit einer speziell auf die Anforderungen dynamisch verteilt ausgeführter Prozesse ausgerichteten Flexibilitätsbetrachtung und einer Zusammenfassung der erarbeiteten Erkenntnisse.

5.1 Vorgehensweise und Auswahl der betrachteten Ansätze

Wie in Kapitel 3 gezeigt wurde, ist zur Ableitung geeigneter Anpassungen zunächst eine möglichst zeitnahe Erhebung von Informationen über die Prozessausführung, über beteiligte Ressourcen und Ausführungssysteme sowie über deren relevanten Kontext notwendig. In Bezug auf verteilt ausgeführte Prozesse setzt dies ein Mindestmaß an *Interoperabilität* der beteiligten Hard- und Softwaresysteme sowie ein gemeinsames Verständnis über die Prozessausführung voraus. Als grundlegendes Modell für die Interoperabilisierung des Prozessmanagements wird dazu das Referenzmodell der *Workflow Management Coalition* (*WfMC*) sowie die darauf aufbauenden Ansätze *Wf-XML* und *CrossFlow* herangezogen, welche konsekutiv weiter entwickelt wurden und daher bis heute eine große Relevanz besitzen (vgl. z. B. [LDK04, Wes07, Gad10]). Da bei einer verteilten Prozessausführung auch die Prozesse selbst ausgetauscht werden, kommt zudem der Standardisierung aktueller Prozessbeschreibungssprachen in diesem Kontext eine große Bedeutung zu. Da die Ansätze zur Interoperabilisierung des Prozessmanagements sowohl aus dem Gesichtspunkt der Verteilung als auch der Überwachung und Steuerung verteilt ausgeführter Prozesse grundlegend sind und die Prozessbeschrei-

bungssprachen *XPDL*, *WS-BPEL* und *BPMN* für das weitere Verständnis dieser Arbeit benötigt werden, werden diese im Vorfeld der folgenden Ansätze in Abschnitt 5.2 kurz erläutert.

Für eine fortwährende Durchführbarkeit von Anpassungen in Bezug auf die Verteilung der Prozessausführung ist nun zunächst eine geeignete Grundstruktur in Hinblick auf das Verteilungsmodell und die zugrunde liegende Architektur von Ausführungsumgebungen erforderlich. Dabei steht nach dem Begriffsverständnis von Flexibilität im Vordergrund, dass die Verteilung eines Prozesses einen möglichst geringen Aufwand verursacht und nach Möglichkeit nicht zukünftige Anpassungen an der Ausführung des Prozesses erschwert (vgl. Abschnitt 3.1). Nach ihren Eigenschaften in Bezug auf diese Flexibilität der Verteilung werden die zu dieser Arbeit verwandten Ansätze in vier Gruppen klassifiziert:

- als Dienst-Choreographien modellierte verteilte Prozessausführungen (vgl. Abschnitt 5.3),
- als horizontale Partitionierung durch die physische Aufteilung des Prozessmodells verteilte Prozessausführungen (vgl. Abschnitt 5.4),
- als vertikale Partitionierung durch Migration von Prozessinstanzen verteilte Prozessausführungen (vgl. Abschnitt 5.5), und
- als Vollverteilung mit minimalen Prozesspartitionen verteilte Prozessausführungen (vgl. Abschnitt 5.6).

Zu jeder Gruppe werden wesentliche Ansätze genannt und in Hinblick auf ihre Flexibilität untersucht. Dabei wird angestrebt, möglichst zu jeder Gruppe den Stand der Forschung zu sowohl stationären als auch mobilen oder hybriden Infrastrukturen aufzugreifen. Auffällig hierbei ist, dass in den genannten Bereichen viele ältere Ansätze existieren, welche vor dem Hintergrund des klassischen Workflow-Managements und der Anwenderintegration entstanden sind, und eine ganze Reihe neuerer Lösungen vorgeschlagen werden, welche den Kontext moderner dienstorientierter Architekturen und der entsprechenden Anwendungsintegration adressieren. Um beide konvergierende Perspektiven des Prozessmanagements integrieren und deren konstruktive Ideen für eine Flexibilisierung verteilter Prozessausführung nutzbar machen zu können, sollen im Folgenden die Ergebnisse beider Arten von Ansätzen aufgegriffen werden. Da viele Ansätze zudem aufeinander aufbauen, wird die Darstellung dabei zumeist in chronologischer Reihenfolge vorgenommen.

Die meisten Verteilungsansätze konzentrieren sich ausschließlich auf den Vorgang der Verteilung und lassen die Erhebung von Informationen zur Feststellung von Anpassungsbedarf sowie die Frage, wie auf entfernten Ausführungseinheiten außerhalb des Einflussbereichs des Prozessinitiators Anpassungen durchgeführt werden sollen, unberücksichtigt. Im Folgenden werden daher zusätzlich Ansätze betrachtet, welche speziell die Überwachung

entfernt ausgeführter Prozesse bzw. Prozesspartitionen und deren etwaige Anpassungen adressieren. Diese werden nach ihrer Architektur klassifiziert und die für die verteilte Prozessausführung relevanten Varianten, welche Einblick in die Interna entfernt ausgeführter Prozesspartitionen erlauben, genauer betrachtet (vgl. Abschnitt 5.7). Gleiches gilt für die Repräsentation von Benutzerschnittstellen im Kontext dynamisch ausgeführter Prozesse, welche aufgrund der bisher fehlenden Dynamik in diesem Kontext nur begrenzt betrachtet wurden. Es werden daher zunächst abstrakte Varianten der Integration von Aufgabenbeschreibungen klassifiziert und hinsichtlich ihrer Flexibilität bewertet. Um eine Basis für die angestrebte Lösung zu erarbeiten, werden im Anschluss sowohl Ansätze aus dem Bereich des Prozessmanagements als auch allgemeine und speziell auf mobile Endgeräte ausgerichtete Ansätze aus dem Gebiet der Mensch-Maschine-Interaktion betrachtet (vgl. Abschnitt 5.8).

Die Ergebnisse der Untersuchung werden in einer integrierten Flexibilitätsbetrachtung zusammengefasst, bei der die Stärken und Schwächen der dargestellten Varianten von Lösungsansätzen herausgearbeitet werden. Die Ergebnisse dienen somit als Ausgangspunkt der nachfolgenden Konzeption zur Flexibilisierung der verteilten Prozessausführung in Kapitel 6.

5.2 Interoperabilität für verteilt ausgeführte Prozesse

Jeder verteilten Prozessausführung muss ein gewisser Grad an Interoperabilität zugrunde liegen, damit relevante Informationen zwischen prozessausführenden Systemen ausgetauscht, interpretiert und sinngemäß verarbeitet werden können. Die Interoperabilität kann dabei sowohl in der Entwicklungsphase prozessorientierter Anwendungen (d. h. deren Modellierung) als auch zu deren Laufzeit (d. h. deren Ausführungsphase) benötigt werden [vdAvH02]. In der *Entwicklungsphase* kann eine gemeinschaftliche Entwicklung bzw. Abbildung von Prozessen oder ein Austausch von fertigen Prozessmodellen notwendig sein. Hierfür ist zunächst ein gemeinsames Verständnis über die Syntax und die Semantik von Prozessen erforderlich. Ein Austausch von Prozessmodellen kann bei einer heterogenen Systeminfrastruktur und unterschiedlichen Prozessbeschreibungssprachen z. B. über ein gemeinsames Austauschformat geregelt werden, in das proprietär spezifizierte Prozessbeschreibungen übersetzt werden können. In der *Ausführungsphase* kann der Austausch von Laufzeitinformationen unterstützt werden. Hierbei steht die Kommunikation zwischen Prozessmanagementsystemen zur Ausführung, Überwachung und die Steuerung von laufenden Prozessinstanzen im Vordergrund.

Im Folgenden werden für beide Phasen grundlegende Konzepte und Ansätze zur Begründung von *Interoperabilität* beschrieben und insbesondere in Hinblick auf ihre Bedeutung und Eignung zur verteilten Prozessausführung sowie auf ihre damit verbundene Flexibilität untersucht.

5.2.1 Referenzmodell der WfMC

Die *Workflow Management Coalition* (*WfMC*) fasst die oben genannten Maßnahmen zur Interoperabilität von Prozessmanagementsystemen in einem integrierten Referenzmodell zusammen, welches bis heute als abstrakte Darstellung der Prozessautomatisierung eine große Relevanz besitzt (vgl. z. B. [Wes07, Gad10]). Die WfMC beschreibt dazu als Basis ein gemeinsames Glossar an Begrifflichkeiten [WfM99a], welche für die Interoperabilität zwischen verschiedenen Prozessmanagementsystemen von Bedeutung sind. Abbildung 5.1 zeigt die wesentlichen Entitäten des Glossars und ihre Beziehungen untereinander, welche auch weitestgehend dieser Arbeit zugrunde liegen (vgl. Kapitel 2). Dabei wird ein Vorgang der Realwelt (*Business Process*) in Form eines Prozessmodells (*Process Definition*) spezifiziert, welches aus Teilprozessen (*Sub-Processes*) und Aktivitäten (*Activities*) zusammengesetzt sein kann. Aktivitäten können manuell verwaltet (*Manual Activities*) oder automatisiert gesteuert werden (*Automated Activities*). Die automatisiert gesteuerten Aspekte eines Prozesses werden durch ein Prozessmanagementsystem (*Workflow Management System*) verwaltet. Hierbei werden aus dem Prozessmodell individuelle Prozessinstanzen (*Process Instances*) abgeleitet, welche ausführbare Repräsentationen der automatisiert zu steuernden Aktivitäten (*Activity Instances*) enthalten. Diese können entweder einem menschlichen Prozessteilnehmer zugewiesen (*Work Items*) oder durch eine Computeranwendung ausgeführt werden (*Invoked Applications*).

Das auf diesem Verständnis aufbauende Referenzmodell (*Workflow Reference Model*) [Hol95] spezifiziert fünf Schnittstellen, die zusammen ein idealtypisches Rahmenwerk für den Austausch von Prozessinformationen darstellen, um die Ausführung von Prozessen zwischen heterogenen Ausführungsumgebungen zu ermöglichen (vgl. Abbildung 5.2). Im Zentrum des Referenzmodells steht eine Ausführungsumgebung für die Interpretation und Bearbeitung von Prozessen (*Workflow Enactment Service*), welche aus mehreren *Workflow-Engines* bestehen kann. Zudem existieren Schnittstellen zur Kommunikation mit anderen Komponenten des Prozessmanagementsystems, also zu Werkzeugen für die Prozessmodellierung oder zu Simulations- und Auswertungskomponenten [Hol95](vgl. Abschnitt 2.7):

- **Process Definition (Interface 1):** Die erste Schnittstelle im Referenzmodell definiert ein allgemeingültiges Format zur Prozessbeschreibung. Proprietäre Prozessmodelle können in eine XML-basierte Sprache übersetzt und zum Austausch mit anderen Prozesspartnern verwendet werden. Die Schnittstelle enthält hierzu sowohl ein Metamodell, welches abgrenzbare elementare Konstrukte eines Prozesses definiert, als auch ein konkretes XML-Schema zum Austausch von Prozessdefinitionen.

- **Workflow Client Application (Interface 2):** Diese Schnittstelle steht zur Verfügung, um Prozesse und Aktivitäten interaktiv durch mensch-

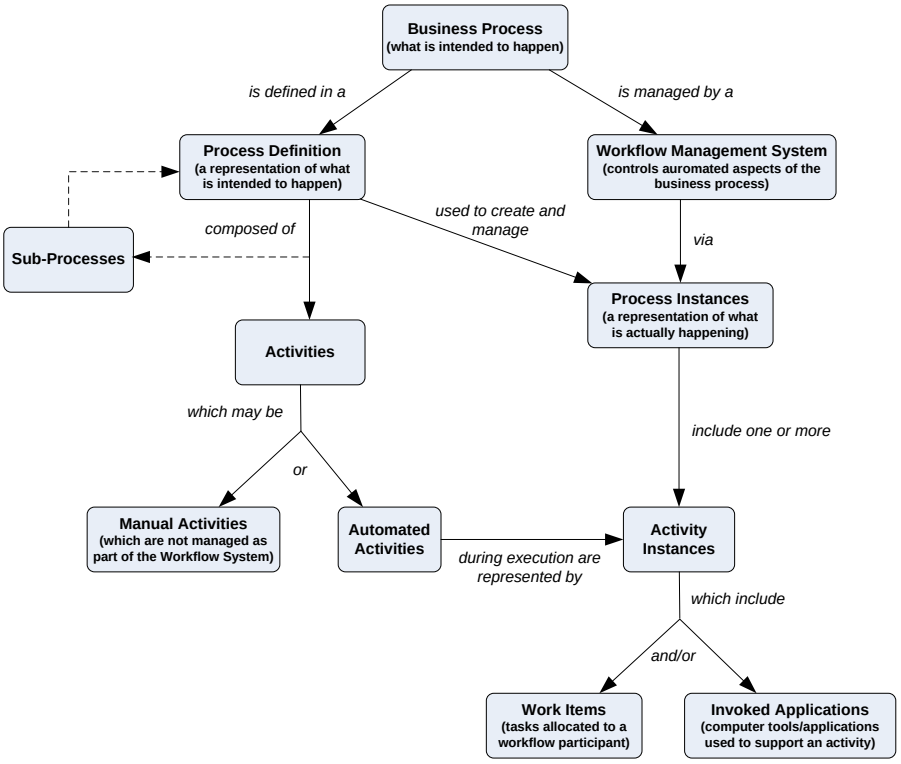


Abbildung 5.1: Grundlegende Konzepte und Terminologie als Grundlage der Interoperabilität des Prozessmanagements (WfM99a)

liche Benutzer zu initiieren oder Aufgaben mit manuellem Anteil ausführen zu lassen.

- **Invoked Applications (Interface 3):** Analog zur interaktiven Ausführung erlaubt die dritte Schnittstelle eine automatisierte Ausführung von Aktivitäten durch einen direkten Programmaufruf.
- **Workflow Interoperability (Interface 4):** Mit Hilfe dieser Schnittstelle können Prozesse und Zustände transformiert werden, um Abläufe auf anderen Workflow-Systemen zu ermöglichen. Zum Beispiel sind Abbildungen und Synchronisationsmaßnahmen nötig, um Subprozesse auf einem heterogenen Workflow-Management-System auszuführen.
- **Administration and Monitoring Tools (Interface 5):** Hinter dieser Schnittstelle verbergen sich Verwaltungs- und Überwachungsfunktionen zum Management von Benutzern, Ressourcen und Prozessen.

Das Referenzmodell der WfMC dient in erster Linie zur Interoperabilisierung von (organisationsinternen) Systemkomponenten unterschiedlicher Hersteller, wobei jedoch insbesondere die erste und vierte Schnittstelle auch eine organisationsübergreifende Zusammenarbeit adressieren (vgl. [vdAvH02, Gad10]). Für die erste Schnittstelle schlägt die WfMC das Austauschformat *WfMC Process Definition Language* (*WPDL*) und darauf aufbauend die

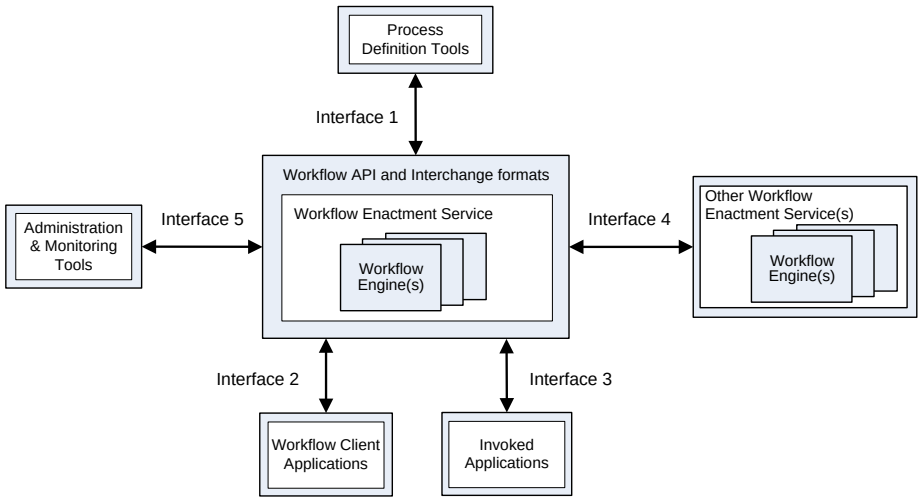


Abbildung 5.2: Referenzmodell der WfMC (Hol95)

XML-basierte Repräsentation *XML Process Definition Language* (*XPDL*) vor [NM02, WfM08, Sha08], welche in Abschnitt 5.2.4 betrachtet wird. Alternative Formate zum Austausch von Prozessen sind das *Process Interchange Format* (*PIF*) [LGJ+98], die *Process Specification Language* (*PSL*) sowie deren gemeinsame, gleichnamige Weiterentwicklung [SGT+04]. Auch die Verwendung von *Petrinetzen* [Rei10] hat in diesem Bereich eine gewisse Verbreitung gefunden, zum Beispiel durch die Anwendung der XML-basierten *Petri Net Markup Language* (*PNML*) auf den Austausch von Geschäftsprozessen [BCvH+03, WK03, Kin04].

Durch die Relevanz verteilt ausgeführter Prozesse wurde das ursprüngliche Referenzmodell inzwischen teilweise weiterentwickelt. Über die *WfMC Interoperability Abstract Specification* [WfM99b], welche Schnittstellen zur Auswahl, Instantiierung und Ausführung von Prozessmodellen auf entfernten Prozess-Engines beschreibt, und das entsprechende internetbasierte *Simple Workflow Access Protocol* (*SWAP*) [Swe99] wird die vierte Schnittstelle unter dem Standard *Wf-XML* [SPG04] zur Überwachung und Steuerung verteilter Prozessinstanzen und deren Verknüpfung zu einem durchgängigen Gesamtprozess fortgeführt [Gad10]. Wf-XML wird daher im folgenden Abschnitt 5.2.2 genauer untersucht.

Im Kontext der vierten und fünften Schnittstelle gewinnt dabei auch der Austausch von aktuellen oder historischen Daten über die Ausführung von Prozessen an Bedeutung. Um Daten dieser Art gezielt weitergeben, interpretieren und weiterverarbeiten zu können, muss zunächst ein gemeinsamer Konsens darüber bestehen, welche Daten im Rahmen der Prozessausführung von den beteiligten Systemen gesammelt und in welcher Form diese zu Analysezwecken bereitgestellt werden können. Die *Audit Data Specification* der WfMC [WfM98a] definiert hierzu einen entsprechenden Vorschlag, welcher auch in der Arbeit des Autors ZUR MUEHLEN [zMR00, zM04] in einem erweiterten Referenz-Metamodell für Audit-Trail-Daten aufgegriffen wird.

Obwohl das explizit für den Kontext der Prozessautomatisierung mit Anwenderintegration entwickelte Rahmenwerk der WfMC in seiner ursprünglichen Form nur den statischen Austausch von Prozessmodellen zur Entwicklungszeit vorsieht, werden hiermit wichtige Grundlagen und Anforderungen für verteilt ausgeführte Prozesse im Allgemeinen adressiert. Das Vorhandensein eines gemeinsamen Verständnisses über Prozessmodelle, Prozessinstanzen, Ausführungssysteme, Daten, Anwendungen und menschliche Prozessteilnehmer auf einer abstrakten, technologie- und organisationsunabhängigen Ebene stellt eine wichtige Grundvoraussetzung für die Verteilung, die verteilte Bearbeitung, die Überwachung und die Anpassung von prozessorientierten Anwendungen dar. Die Bereitstellung von geeigneten Schnittstellen zum Aufruf von Ressourcen, zum Abruf von Informationen und zur Steuerung des Prozessmanagementsystems stellt eine weitere wichtige Anforderung dar. Wie in Abschnitt 3.4 gezeigt wurde, führt die Umsetzung von offenen Schnittstellen im Rahmen von dienstorientierten Architekturen zu einem großen Flexibilisierungspotential, während der in einer geeigneten Form vereinheitlichte Austausch von Daten über die Ausführung verteilter Prozesse eine wichtige Voraussetzung für das Erkennen von Anpassungsbedarf sein kann (vgl. Abschnitte 3.5, 3.7 und 3.8). Das sowohl in der Wissenschaft als auch in der Praxis anerkannte Modell der WfMC bildet daher einen wichtigen Ausgangspunkt dieser Arbeit.

5.2.2 Wf-XML

Um die Integration von verschiedenartigen Prozess-Engines zur Laufzeit zu unterstützen, schlägt die WfMC mit *Wf-XML* [SPG04] ein XML-basiertes Protokoll zur Ausführung und Überwachung von Prozessen auf entfernten Ausführungseinheiten vor. Das vorgeschlagene Protokoll basiert auf dem *Asynchronous Service Access Protocol* (*ASAP*) von OASIS [OAS04, OAS05], welches eine einfache Erweiterung zu SOAP darstellt, um auf Anwendungsebene asynchrone Web Services zu steuern und ihre Ausführung zu beobachten. Ein Beobachter (*Observer*) kann dabei ausgehend von einem Dienstverzeichnis (*Service Registry*) über eine Liste verfügbarer Diensttypen auf Funktionen zur Verwaltung der einzelnen angebotenen Dienste zugreifen. Die Steuerungs- bzw. Überwachungsmöglichkeiten umfassen die Erstellung eines Dienstes, dessen Konfiguration, das Starten und Stoppen von Dienstinstanzen sowie den Bezug von Informationen zu Fehlern, dem Status und dem Ende der Ausführung sowie den Ergebnissen des Dienstes. Wf-XML erweitert die durch ASAP vorgeschlagenen Schnittstellen dabei für den speziellen Fall, dass es sich bei dem Dienst um eine komplexe, durch eine Prozess-Engine verwaltete Funktionalität handelt [SPG04].

Das aus diesen Überlegungen resultierende Modell eines durch eine Prozess-Engine ausgeführten asynchronen Dienstes im Sinne von ASAP ist in Abbildung 5.3 dargestellt. Dabei wird das Prozessmodell auf die dargestellte Dienstfabrik (*Factory*) und die Prozessinstanz auf die Dienstinstanz (*Instance*) ab-

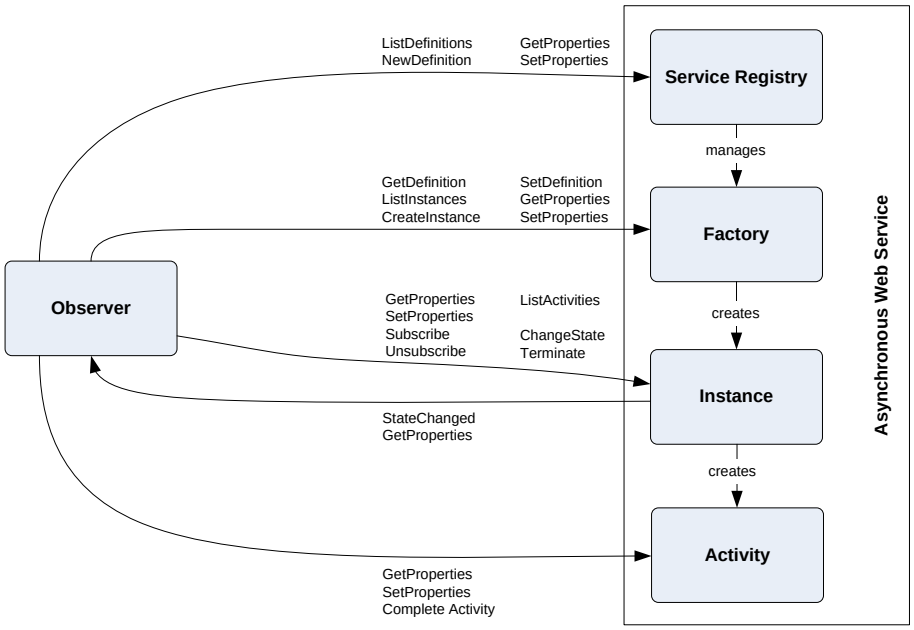


Abbildung 5.3: Wf-XML-Modell eines durch eine Prozess-Engine ausgeführten Dienstes (SPG04)

gebildet, welche in diesem Fall aus Aktivitäten (*Activities*) zusammengesetzt ist. Die durch Wf-XML definierten Erweiterungen erlauben darauf aufbauend zusätzliche Funktionalitäten, wie die Abfrage der durch die Prozess-Engine auszuführenden Aktivitäten, der Zuordnung von Ressourcen, des Inhaltes von Prozessvariablen oder sogar der Prozessmodelle von entfernt ausgeführten Subprozessen. Zudem wird durch die Bereitstellung entsprechender Schnittstellen das Einstellen, Ändern oder Entfernen von Prozessmodellen erlaubt [SPG04]. Wf-XML kann somit als wichtiger Schritt in Richtung der Modellierung eines Prozesses als *verwaltbare Ressource* im Sinne von Abschnitt 3.8.3 gewertet werden.

5.2.3 CrossFlow

Auf Basis des von der WfMC vorgeschlagenen Referenzmodells (vgl. Abschnitt 5.2.1) unterstützt das Projekt *CrossFlow* [LH99, GALH00, HLGG00, HLGA01] die Abbildung von organisationsübergreifenden Geschäftsbeziehungen durch die geeignete Verknüpfung von Anbietern und Konsumenten zur (prozessorientierten) Leistungserstellung. Dabei steht bei CrossFlow vor allem die Erweiterung der von der WfMC vorgeschlagenen Interoperabilisierung des Prozessmanagements durch die Aushandlung und Erstellung von Verträgen sowie die Durchführung und Überwachung der vereinbarten Leistungen in funktionaler und nicht-funktionaler Hinsicht im Mittelpunkt. Die betrachtete Einheit der organisationsübergreifenden Zusammenarbeit ist hier durch das Auslagern einer einzelnen Aktivität eines Prozesses an einen externen Kooperationspartner gegeben. Analog zu der dynamischen Integration von elektroni-

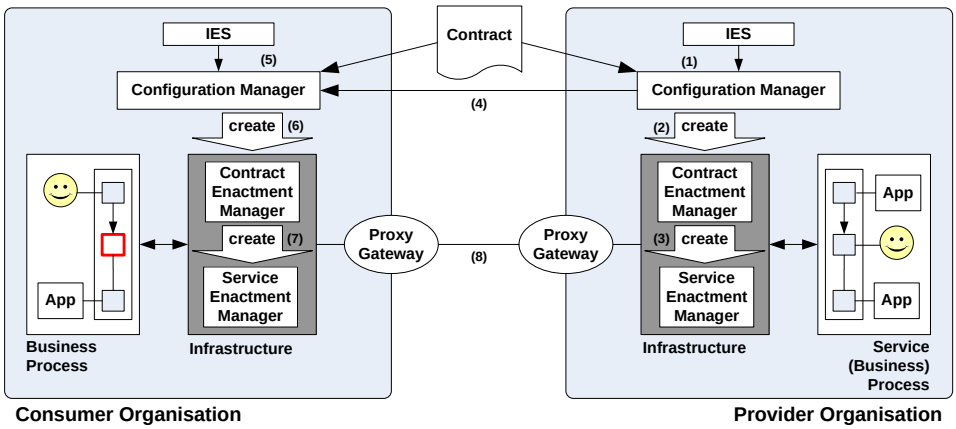


Abbildung 5.4: CrossFlow: Erstellung von Komponenten zur Verknüpfung von Prozessmanagementsystemen auf Basis von Verträgen (HLGG00)

schen Diensten (vgl. Abschnitt 3.4.2) wird hierbei folgende Vorgehensweise vorgeschlagen [LH99, HLGG00]:

- Die auslagernde Organisation sucht nach verfügbaren Dienstangeboten zur Erbringung der benötigten Leistung und erzielt mit der dienstanbietenden Organisation (ggf. nach einer Verhandlungsphase) Einigung über die inhaltlichen und technischen Voraussetzungen für eine Zusammenarbeit.
- Beide Organisationen konfigurieren ihre Systeme entsprechend der im Vertrag festgelegten Bedingungen.
- Die Aktivität wird ausgelagert und durch den Diensterbringer in einem internen Prozess ausgeführt. Diese Phase kann (falls Teil des Vertrags) auch die Überwachung der extern ausgeführten Leistung oder erweiterte Steuerungsfunktionalitäten beinhalten.
- Die Ausführung des Dienstes terminiert und die Geschäftsbeziehung wird beendet.

Um einen dynamischen und möglichst automatisierten Ablauf dieser organisationsübergreifenden Kooperation zu erlauben, werden auf Basis der vertraglichen Vereinbarung spezielle Schnittstellen in Form von *Proxy-Gateways* generiert, welche exakt die vorgesehene Kooperation zwischen den beteiligten Parteien erlauben (vgl. Abbildung 5.4) [LH99, HLGG00, HLGA01]. Diese umfasst die Verbindung der Prozessmanagementsysteme zur funktionalen Durchführung sowie zur Überwachung und Steuerung der festgelegten Leistung (inklusive einer etwaigen Transformation von Daten- und/oder Nachrichtenformaten). Die Einrichtung der spezialisierten Schnittstellen ist dabei insbesondere für wiederkehrende Geschäftsbeziehungen gleicher Art vorteilhaft, da hierbei die bereits generierten Komponenten wiederverwendet werden können [LH99].

Der Ansatz von CrossFlow trägt zur dynamischen Zusammenarbeit zwischen mehreren Prozessmanagementsystemen bei, verfolgt aber durch die Auslagerung einzelner Aktivitäten (vgl. *ressourcenbezogene Verteilung* in Abschnitt 4.1.2) im Vergleich zu der Problemstellung dieser Arbeit ein anderes Ziel. So wird bei CrossFlow nicht ein in seiner Struktur festgelegter Prozessabschnitt zur Ausführung an ein anderes Prozessmanagementsystem übergeben, sondern eine vorgegebene Funktionalität in einer ausgehandelten Qualität auf beliebige Weise durch einen externen Kooperationspartner erbracht. Aufgrund der Orientierung an realen Vorgängen zu Vertragsverhandlungen stellt die hier beschriebene Vorgehensweise dennoch ein wichtiges Rahmenwerk zur Begründung von dynamischen Kooperationen und einen wichtigen Ausgangspunkt für weiterführende Arbeiten (z. B. [KWA99a, KWA99b, LDK04], vgl. Abschnitte 5.7.1 und 5.7.4) dar.

5.2.4 Standardisierte Prozessbeschreibungssprachen

Die Bereitstellung, der Austausch und die Transformation von Prozessmodellen mit Hilfe eines gemeinsamen Austauschformats ist zwar in vielen Fällen automatisch möglich (vgl. [vdAvH02]), jedoch sind die damit verbundenen Abbildungsvorgänge aufgrund der Individualität der proprietären Prozessbeschreibungssprachen oft fehleranfällig und nicht verlustfrei möglich [RM06]. Auf Basis eines grundlegenden allgemeinen Verständnisses der Prozessautomatisierung stellt daher gegenwärtig die Nutzung einer gemeinsamen ausführbaren Prozessbeschreibungssprache eine wichtige Vorgehensweise zur weiteren Interoperabilisierung des Prozessmanagements dar, was insbesondere durch die zunehmenden Bemühungen zur Standardisierung zahlreicher Prozessbeschreibungssprachen (z. B. [OAS07, WfM08, OMG11]) unterstrichen wird. Im Folgenden werden exemplarisch drei weit verbreitete, standardisierte Prozessbeschreibungssprachen vorgestellt, welche den in Abschnitt 2 vorgestellten Eigenschaften von Prozessbeschreibungssprachen zur Definition ausführbarer prozessorientierter Anwendungen entsprechen.

XML Process Description Language (XPDL)

Als Realisierung der von der WfMC vorgeschlagenen ersten Schnittstelle des Referenzmodells (vgl. Abschnitt 5.2.1) setzt die *XML Process Description Language* (*XPDL*) die Idee eines Metaformats zum Austausch von Prozessmodellen zwischen verschiedenen Prozessmanagementsystemen um [NM02, NMS05, WfM08]. Neben dieser Funktion als Austauschformat hat sich XPDL jedoch in den vergangenen Jahren teilweise auch als direkt ausführbare Prozessbeschreibung durchgesetzt. Nach einer Aufstellung der WfMC wird die Sprache derzeit in über 70 kommerziellen oder freien Produkten zur Modellierung und/oder Ausführung von Prozessen unterstützt [WfM11].

Bei XPDL handelt es sich um eine XML-basierte Prozessbeschreibungssprache zur Definition überwiegend graphstrukturierter Prozessmodelle mit einem relativ hohen Abstraktionsgrad. So wurde in vorhergehenden Versionen der

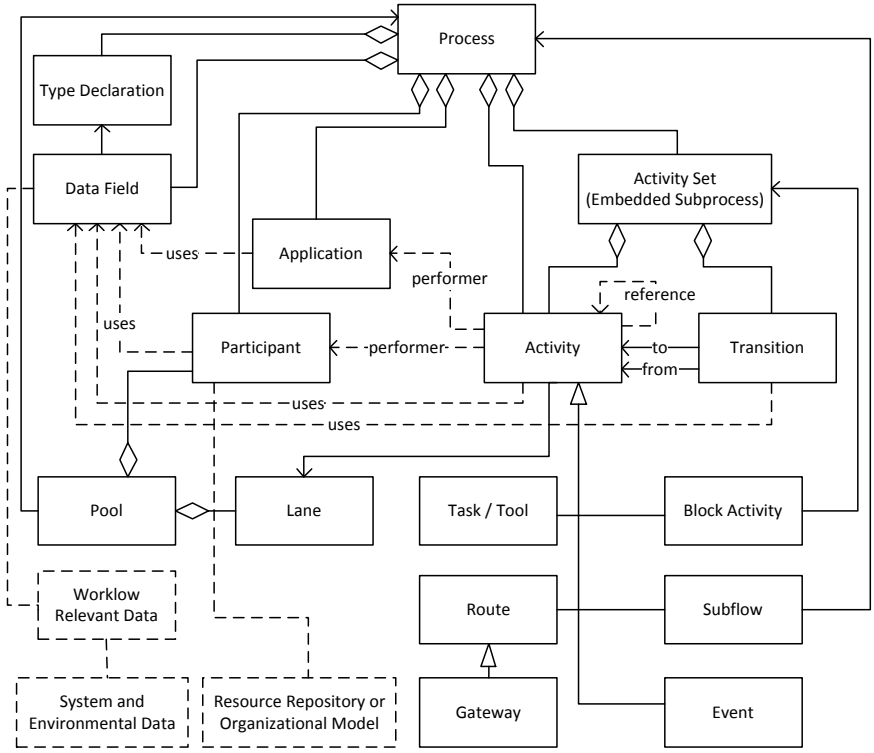


Abbildung 5.5: Metamodell zur Erstellung von XPDL-Prozessmodellen (WfM08)

Sprache stets von konkreten Technologien zur Integration von Anwendungsfunktionalitäten abstrahiert und nur zwischen manuell auszuführenden Aktivitäten und durch Computersysteme auszuführenden Aktivitäten unterschieden [NM02, NMS05]. Erst in einer neueren Version von XPDL [WfM08] wird der voranschreitenden Akzeptanz zur Repräsentation von Anwendungsfunktionalität durch Web Services Rechnung getragen und es werden spezielle Konstrukte zur Unterstützung von Web-Service-Schnittstellen zur Verfügung gestellt. XPDL unterstützt über das generische Konstrukt des *Application Types* neben Web Services jedoch auch noch eine Reihe anderer Technologien (wie z. B. EJB-Komponenten oder gewöhnliche Java-Klassen) und kann somit immer noch als relativ technologieunabhängige Prozessbeschreibungssprache angesehen werden.

Neben den meisten der bereits in Abschnitt 2.4 beschriebenen Kontrollflusselementen setzt XPDL vor allem die Definition und Zuordnung von Prozessbeteiligten zu Aktivitäten um. Abbildung 5.5 zeigt eine Übersicht des Metamodells zur Spezifikation von XPDL-Prozessen. Hierbei kann ein Prozess aus beliebigen atomaren oder komplexen Aktivitäten (*Activities*) und ihrer Verknüpfung durch Transitionen (*Transitions*) zusammengesetzt sein. Bei der Ausführung einer Aktivität durch eine Softwarefunktionalität (*Application*) oder einen menschlichen Prozessteilnehmer (*Participant*) können Datenfelder (*Data Fields*) eines bestimmten Datentyps (*Type Declaration*) bearbeitet werden. Durch die Zuordnung von Aktivitäten zu einer Bahn (*Lane*) können

menschliche Prozessteilnehmer aus einem *Pool* nach der entsprechenden Deklaration in einem Organisationsmodell (*Resource Repository or Organizational Model*) zur Ausführung der Aktivität zugewiesen werden (vgl. auch Abschnitt 2.6) [WfM08]. Die Zuordnung zu Pools erlaubt dabei prinzipiell auch die Spezifikation der Verteilung des Prozesses an die für die Ausführung verantwortlichen Systeme.

Um Kompatibilität zu einer geeigneten graphischen Darstellung zu erreichen orientiert sich XPDL zur Zeit stark an der Entwicklung der graphischen Prozessbeschreibungssprache *Business Process Model and Notation* (*BPMN*) (s.u.). Des Weiteren wird eine Unterstützung des *Business Activity Monitorings* (*BAM*) und der Simulation von Prozessausführungen angestrebt [Sha08].

Web Service Business Process Execution Language (WS-BPEL)

Die *Web Service Business Process Execution Language* (*WS-BPEL*) ist eine XML-basierte Sprache zur prozessorientierten Komposition von Web Services [And03, OAS07]. Sie wurde im Jahr 2007 in der Version 2.0 [OAS07] von der *Organization for the Advancement of Structured Information Standards* (*OASIS*) zu einem offiziellen Standard erklärt. Der durch WS-BPEL beschriebene Prozess wird dabei nach dessen Deployment ebenfalls über einen Web Service bereitgestellt. Er kann daher durch Aufruf der entsprechenden Web-Service-Schnittstelle instantiiert und somit auch als Bestandteil anderer WS-BPEL-Prozesse verwendet werden.

In Hinblick auf eine verteilte Ausführung von WS-BPEL-Prozessen können bei der Modellierung zwei verschiedene Ausprägungen unterschieden werden [OAS07]:

- **Modellierung direkt ausführbarer Geschäftsprozesse:** Tatsächlich ausführbare Geschäftsprozesse stellen organisationsinterne Vorgänge dar. Sie enthalten eine konkrete Implementierung, die nach außen hin sichtbar und veränderbar ist [OAS07]. Der Prozess wird dabei immer aus der Perspektive einer der involvierten Parteien koordiniert. Dies entspricht der Sichtweise der *Orchestrierung* von Web Services (vgl. Abschnitt 3.4.4).

- **Modellierung abstrakter Prozesse:** Hierbei handelt es sich um die Definition sogenannter Geschäftsprotokolle, die für eine Interaktion mit anderen Organisationen zur Verfügung stehen. Die Koordination ist daher von gemeinschaftlicher Natur, in der jede involvierte Partei beschreibt, welche Rolle sie innerhalb des Prozesses spielt. Abstrakte Prozesse bilden lediglich Schnittstellen zu konkreten ausführbaren Prozessen und verbergen so ihr internes Verhalten [OAS07]. Dies entspricht im Wesentlichen der Sichtweise der *Choreographie* von Web Services (vgl. Abschnitt 3.4.5).

WS-BPEL ist eine überwiegend blockstrukturierte Sprache, so dass im Wesentlichen für alle in Abschnitt 2.4 beschriebenen Kontrollflussstrukturen ex-

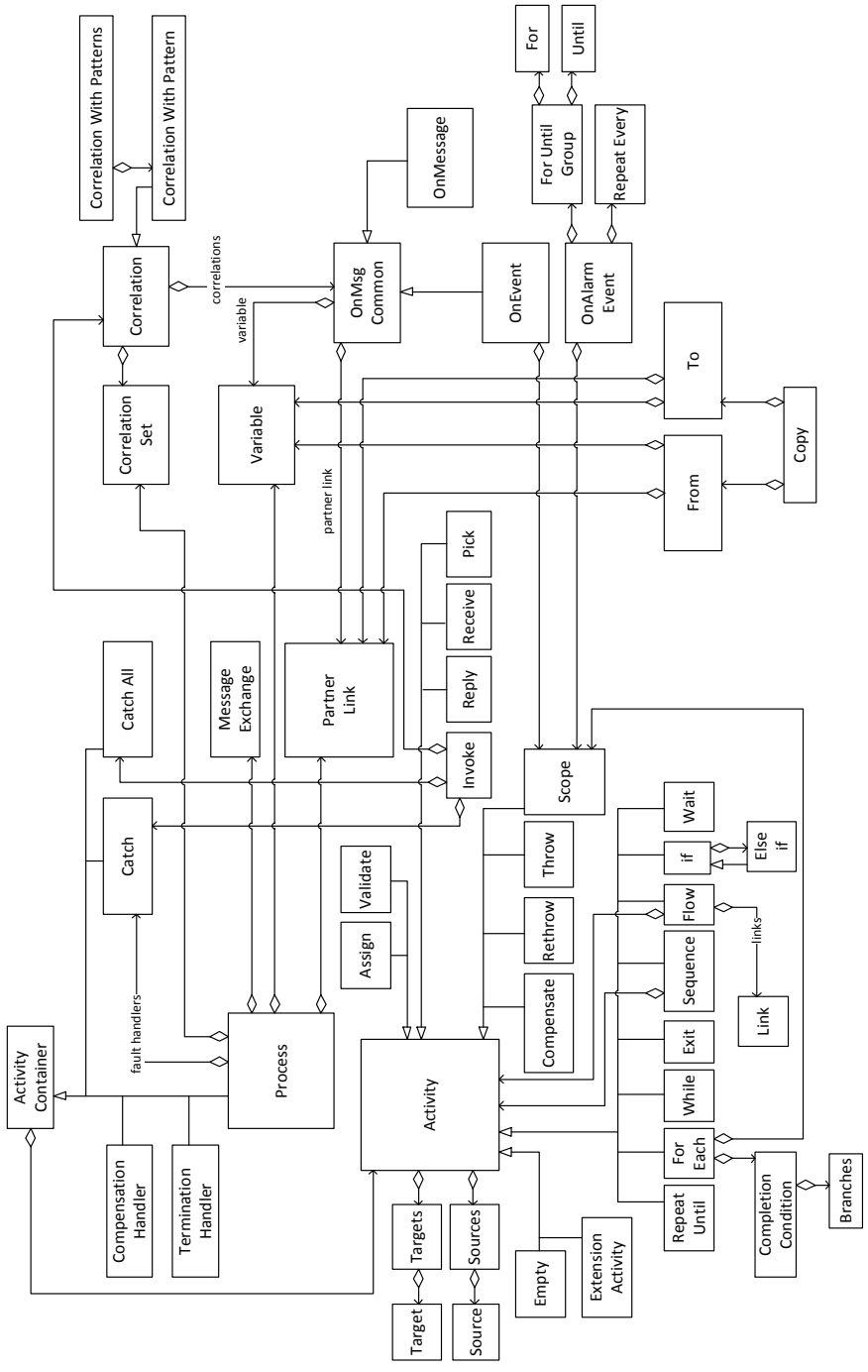


Abbildung 5.6: Meta-Modell zur Erstellung von WS-BPEL-Prozessmodellen (Version 2.0) (OAS07), Darstellung nach (Dub07)

plizite Konstrukte bereitgestellt werden (vgl. Abbildung 5.6). Die atomaren Aktivitäten sind jedoch nicht wie bei XPDL generisch verwendbar, sondern sind stark auf die Komposition von Web Services ausgerichtet. So stehen insbesondere spezielle Konstrukte zum Aufruf von Web Services (*invoke*), zum Warten auf einen eingehenden Aufruf (*receive*) und zur Beantwortung von Aufrufen (*reply*) zur Verfügung. Die Blockstrukturierung bietet zudem umfassende Möglichkeiten zur Definition von Gültigkeitsräumen, zum Beispiel für die lokale Zuordnung von Variablen, Fehlerbehandlungsmaßnahmen und Ereignissen. Da WS-BPEL die Anwendungsintegration fokussiert, wird die direkte Interaktion mit menschlichen Prozessteilnehmern in der Basisspezifikation nicht berücksichtigt. Der Ausschluss von interaktiven oder manuellen Aktivitäten stellt bei der Abbildung von Realweltvorgängen jedoch ein großes Defizit dar. Die Erweiterungen *WS-Human-Task* [AAD+07a] und *WS-BPEL4People* [AAD+07b] stellen daher zusätzliche Möglichkeiten zur (abstrakten) Beschreibung von benutzergesteuerten Aufgaben, die Definition von Rollenbeziehungen und die Umsetzung von Interaktionsmustern zur Verfügung (vgl. auch Abschnitt 5.8.2).

Business Process Model and Notation (BPMN)

Ziel der aktuellen Version der Prozessbeschreibungssprache *Business Process Model and Notation* (*BPMN*) [OMG11] ist es, den Bruch zwischen der strategischen-operativen Modellierung auf fachlicher Ebene und der ausführbaren Beschreibung eines Prozesses auf der technischen Ebene weitestgehend zu vermeiden (vgl. Abschnitt 2.3). Dazu wird durch BPMN in der Version 2.0 zur Unterstützung verschiedener menschlicher Experten eine graphische Modellierungssprache spezifiziert, welche direkt in ein entsprechendes maschinenlesbares XML-Format übertragen werden kann. Die aus diesen beiden Komponenten bestehende Prozessbeschreibungssprache BPMN 2.0 mit expliziter Ausführungssemantik wurde im Jahr 2011 durch die *Object Management Group* (*OMG*) als offizieller Standard verabschiedet.

Sowohl die graphische als auch die textuelle Repräsentation von BPMN sind graphstruktiert, wobei jedoch auch komplexe Aktivitäten in Form von Blockstrukturen und ereignisbasierten Gültigkeitsbereichen ausgedrückt zugeordnet werden können. Neben der Modellierung von Prozessen mit den in Abschnitt 2.4 beschriebenen Kontrollflussstrukturen unterstützt BPMN außerdem die Entwicklung von Kollaborations- und Choreographiemodellen. Während des Entwicklungsvorgangs (vgl. [FRH10]) kann dabei ein besonderer Schwerpunkt auf die Zuordnung von Prozessaktivitäten zu Rollen und Benutzern sowie deren Zuordnung zu automatischen Ausführungseinheiten (Prozess-Engines) gelegt werden. Ähnlich wie bei XPDL wird dies durch die graphische Darstellung von Bahnen (*Lanes*) und übergeordneten Behältern (*Pools*) erreicht, welche ebenfalls in die XML-Repräsentation des Prozesses übernommen werden können [OMG11]. Die Modellierung von Verteilung kann

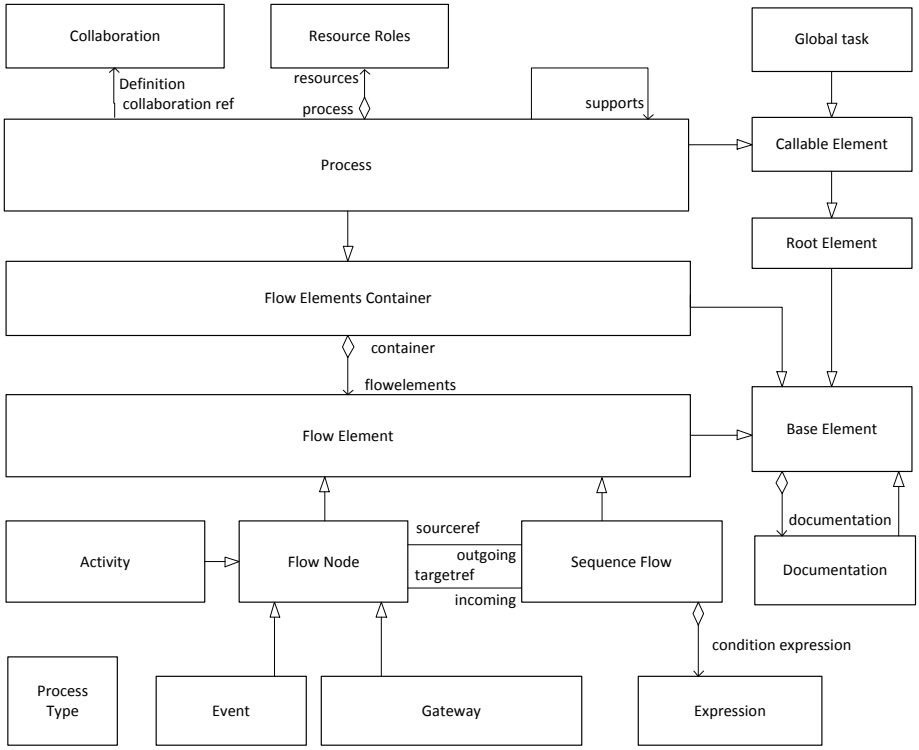


Abbildung 5.7: Meta-Modell zur Erstellung von BPMN-Prozessmodellen (Version 2.0)(OMG11) (Überblick)

daher mit BPMN bei Bedarf schon zur Entwicklungszeit eines Prozesses unterstützt werden.

Die aus der Transformation des graphischen Modells gewonnene XML-Repräsentation ist in den meisten Fällen noch nicht direkt ausführbar und muss um plattform- bzw. technologiabhängige Beschreibungen zur Integration von Anwendungen (z. B. Java-Klassen oder Web Services) und Benutzungsschnittstellen (z. B. HTML-Formulare) ergänzt werden [FRH10, RvL11]. Ebenso wie XPDL abstrahiert somit auch BPMN von konkreten Ausführungsdetails. Durch die Definition einer expliziten Ausführungssemantik kann BPMN jedoch eingeschränkt auch zur graphischen Repräsentation anderer Prozessbeschreibungssprachen (z. B. WS-BPEL) oder als Austauschformat zur Interoperabilisierung verschiedener Organisationen oder Organisationseinheiten eingesetzt werden. Abbildung 5.7 zeigt einen Überblick über das Meta-Modell von BPMN 2.0. Eine detailliertere Betrachtung erfolgt in Abschnitt 8.3.3.

Unter der Annahme, dass Prozesse standardisierter Prozessbeschreibungssprachen durch Prozessmanagementsysteme unterschiedlicher Organisationen gleichartig interpretiert werden, kann sowohl eine gemeinsame Modellierung als auch eine organisationsübergreifende Ausführung von prozes-

sorientierten Anwendungen erleichtert werden. Die Standardisierung von Prozessbeschreibungssprachen stellt daher eine wichtige Basis für eine dynamisch verteilte Prozessausführung dar, da somit zwischen kompatiblen Ausführungseinheiten prinzipiell auch zur Laufzeit eine Übertragung von direkt ausführbaren Prozessen bzw. Prozesspartitionen ermöglicht werden kann. Ein hoher Abstraktionsgrad zum Aufruf von Ressourcen (wie z. B. bei XPDL und BPMN) trägt zudem dazu bei, dass nach einer Verteilung die durch die Prozessbeschreibung vorgegebenen Ressourcen durch die lokale Ausführungseinheit neu gebunden werden können. Es kann damit zusammenfassend nicht nur die ausführungsbezogene Verteilung, sondern auch die ressourcenbezogene bzw. die organisationsbezogene Verteilung besser unterstützt werden.

5.3 Verteilung durch Choreographie

Ein aktueller Ansatz zur Verteilung der Prozessausführung ist die Beschreibung einer Choreographie (vgl. Abschnitt 3.4.5). Hierbei wird auf Basis einer dienstorientierten Architektur der auszuführende Prozess in Form eines abstrakten Kommunikationsprotokolls vorgegeben. Die an der Prozessausführung (potentiell) teilnehmenden Parteien realisieren hierzu jeweils entsprechende Dienstschnittstellen, hinter denen die Implementierung der für den Gesamtprozess zu erbringenden Leistung lokal gekapselt ist. Die Kontrollflusslogik der lokalen Prozesse und die Auswahl und Einbindung von lokalen oder entfernten Ressourcen für die Ausführung des Prozesses bleibt somit den teilnehmenden Parteien der Choreographie selbst überlassen. In diesem Abschnitt wird die Flexibilität von durch Choreographie verteilt ausgeführten Prozessen diskutiert. Dazu werden im Folgenden sowohl Choreographien mit statisch festgelegten Teilnehmern als auch Möglichkeiten zur dynamischen Auswahl von Ausführungseinheiten betrachtet.

5.3.1 Statische Festlegung von Ausführungseinheiten

Eine einfache Art und Weise zur technischen Umsetzung einer verteilten Prozessausführung ist das explizite Festschreiben aller an einer Choreographie beteiligten Parteien und die entsprechende Integration der konkreten Dienstendpunkte der Partner (z. B. die URL) in die jeweiligen Dienstaufrufe der privaten Orchestrierungen (vgl. Abbildung 5.8). Dabei wird die Übergabe des Kontrollflusses an einen anderen Prozessteilnehmer als weitere Aktivität eingebunden, welche zum Inhalt hat, eine festgelegte Schnittstelle des jeweiligen Partners aufzurufen und die zur Verteilung der Prozessausführung relevanten Daten zu übergeben. Der globale, durch die Choreographie definierte Prozess ist hierbei relativ fest determiniert und eine Anpassung der teilnehmenden Ausführungseinheiten muss entweder durch jeden lokalen Prozessteilnehmer selbst verwaltet werden (was unter Umständen die Konsistenz des übergeordneten Prozesses gefährden kann) oder erfordert den Eingriff ei-

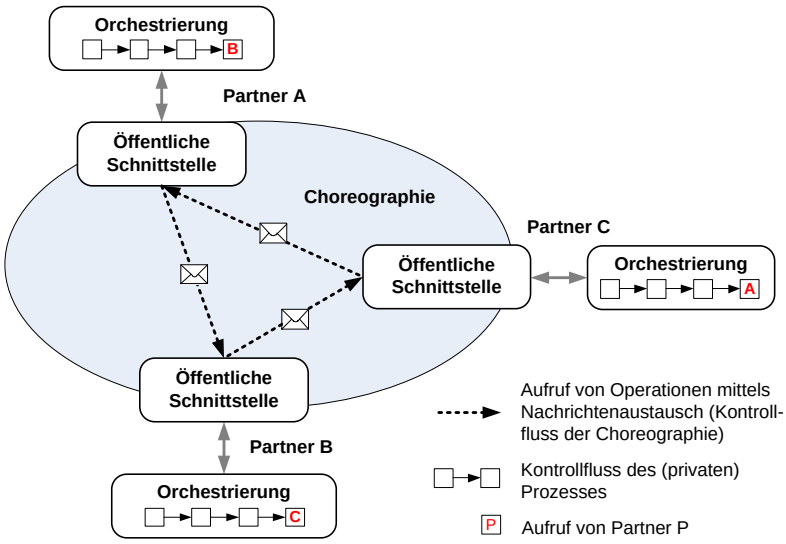
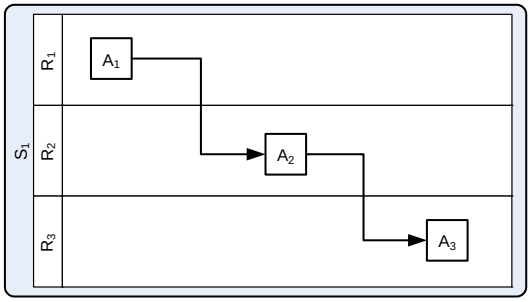


Abbildung 5.8: Statische Choreographie (vgl. Abbildung 3.8)

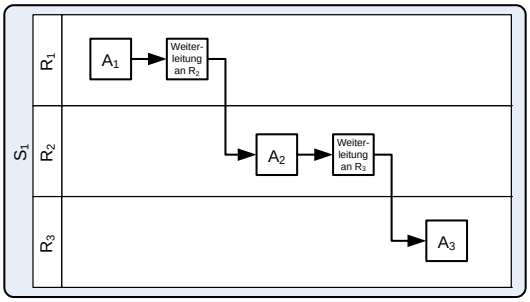
nes zentralen Koordinators (was wiederum die Autonomie der teilnehmenden Parteien in Bezug auf ihre internen Prozesse beeinträchtigt).

Vergleicht man diesen Ansatz der statischen Zuweisung von Prozessteilnehmern mit der in Kapitel 2 vorgestellten Modellierung von fachlicher Ausführungslogik und der über die Abbildung einer Organisationsstruktur lose gekoppelten Zuweisung von Ressourcen, so wird deutlich, dass an dieser Stelle technische Aspekte in die Prozessbeschreibung eingewoben werden, welche im Fall zentral ausgeführter Prozesse eigentlich durch das Prozessmanagementsystem verwaltet und somit gekapselt werden. Der in Abbildung 5.9a dargestellte fachliche Prozess würde daher bei Vorhandensein einer zentralen Kontrollinstanz in den seltensten Fällen wie in Abbildung 5.9b modelliert werden, da dies augenscheinlich die zuvor bestehende Flexibilität der Zuordnung von Aufgaben an konkrete Ressourcen auflösen würde. In der Modellierung von statischen Choreographien ist dieser Schritt jedoch gegenwärtig eine empfohlene Vorgehensweise, um eine einfache dezentrale Ausführung des Prozesses zu erreichen (vgl. Abbildung 5.9c) [FRH10]. Wiederverwendbarkeit und Anpassung des fachlichen Prozessmodells sowie die spontane Änderung der Verteilungskonfiguration werden dadurch erschwert oder sogar gänzlich verhindert. Die Art der Flexibilität, die durch eine solche Umsetzung erreicht wird, bezieht sich daher ausschließlich auf die neu erreichten Freiheitsgrade für die teilnehmenden Parteien einer Choreographie, welche die technische Umsetzung der von ihnen bereitzustellenden Funktionalität unter den vorgegebenen semantischen und syntaktischen Rahmenbedingungen frei gestalten können.

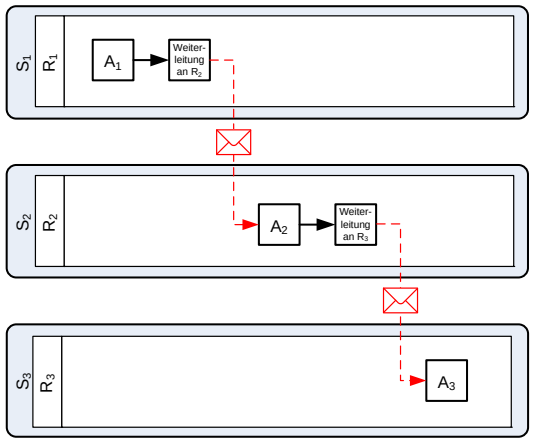
Eine statische Zuordnung von Ausführungseinheiten trägt daher offensichtlich nicht zu einer flexiblen verteilten Ausführung für Prozesse in dynamischen Umgebungen bei. Ein großer Vorteil von Choreographien besteht jedoch darin, dass die lokalen Orchestrierungen der teilnehmenden Parteien einer Choreographie wieder über eine Dienstschnittstelle aufgerufen wer-



(a) Zentral ausgeführter Prozess mit fachlichen Aktivitäten A_1, A_2 und A_3



(b) Organisationsinterne, dezentrale Ausführung mit Aktivitäten zur Kontrollflusssteuerung



(c) Organisationsübergreifende, dezentrale Ausführung mit Aktivitäten zur Kontrollflusssteuerung

Abbildung 5.9: Explizite Modellierung des Kontrollflusses über mehrere Ressourcen und Systeme (Darstellung nach (FRH10))

den können. Es können daher zur dynamischen Auswahl von geeigneten Ausführungseinheiten prinzipiell alle bestehenden Verfahren zum Auffinden und zur Einbindung von (atomaren) Diensten verwendet werden (vgl. auch Abschnitt 3.4.3). Im folgenden Abschnitt wird die Anwendung solcher Konzepte auf die Auswahl von Teilnehmern einer Choreographie zusammenfassend betrachtet.

5.3.2 Dynamische Auswahl von Ausführungseinheiten

Im Bereich der dienstorientierten Architekturen existieren vielfältige Konzepte zur dynamischen Auswahl von Diensten und zu der flexiblen Einbindung von Diensten in Dienstkompositionen. Nach ALONSO et al. [ACKM04] können im Allgemeinen vier verschiedene Arten der Bindung von Diensten in prozessorientierten Anwendungen unterschieden werden (vereinfacht auch in [Wes07]):

- **Statische Bindung:** Der Endpunkt eines Dienstes ist statisch in die Prozessbeschreibung integriert.
- **Dynamische Bindung durch explizite Referenz:** Hierbei wird der Endpunkt eines Dienstes als Wert einer bestimmten Prozessvariablen spezifiziert. Um einen Endpunkt dynamisch einer Prozessvariablen zuzuweisen, muss eine zusätzliche (technische) Aktivität spezifiziert werden, welche den konkreten Endpunkt während der Laufzeit des Prozesses bei einem Verzeichnisdienst abfragt und das Ergebnis der Abfrage als Variablenwert ablegt.
- **Dynamische Auswahl von Operationen:** Bei dieser Art der dynamischen Bindung wird nicht nur der konkrete Endpunkt des Dienstes, sondern auch eine funktional geeignete Operation dieses Dienstes ausgewählt. Die dynamische Auswahl kann wiederum als explizite Alternative im Kontrollfluss des Prozesses modelliert werden, oder durch eine abstrakte Aktivität beschreiben werden. Es ist hierbei also theoretisch auch eine inhaltliche Anpassung des Prozesses möglich.
- **Dynamische Bindung durch Nachschlagen (Look-Up)**: In diesem Fall erlaubt die zur Komposition verwendete Middleware eine Integration von komplexen Suchanfragen als Teil der Aktivitätsbeschreibung. Für jede Aktivität wird dann durch die Middleware ein entsprechend geeigneter Dienst gewählt, welcher die benötigte Funktionalität implementiert. Auch die Berücksichtigung nicht-funktionaler Aspekte ist möglich.

Sind bei verschiedenen potentiellen Kooperationspartnern die in der Choreographiebeschreibung spezifizierten Dienstschnittstellen vorhanden, so können die genannten Verfahren prinzipiell auch auf die Auswahl und Einbindung von Partnern einer Choreographie und somit auf die Auswahl von Ausführungseinheiten einer verteilten Prozessausführung angewendet werden. Zur Flexibilisierung der verteilten Prozessausführung in einer Choreographie von (Teil-)Prozessen beschreibt WESKE [Wes07] daher analog zu der *dynamischen Bindung durch explizite Referenz* die explizite Modellierung von Aktivitäten zur Auswahl von Ausführungseinheiten. Dabei kann die für die Verteilung der Prozessausführung verantwortliche Partei selbst eine Menge von möglichen Kooperationspartnern verwalten oder einen (unabhängigen) Dritten (z. B. einen Verzeichnisdienst oder aktiven Vermittler) mit der Auswahl eines geeigneten Partners beauftragen. Abbildung 5.10 zeigt ein Beispiel für

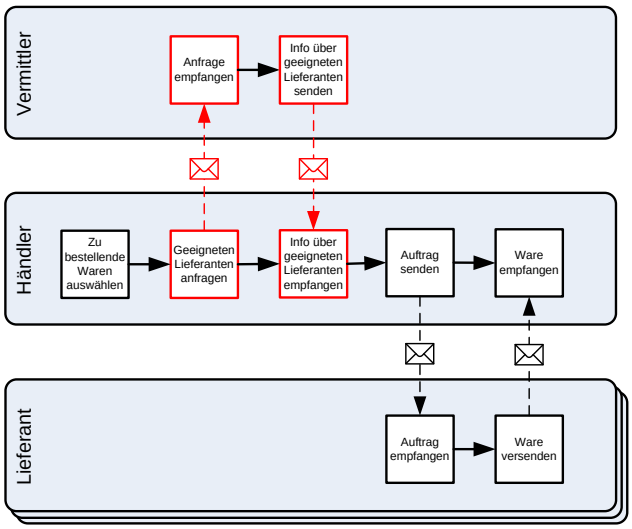


Abbildung 5.10: Auswahl von Kooperationspartnern zur Laufzeit des Prozesses (Wes07)

eine solche dynamische Auswahl innerhalb einer Choreographie, indem ein Vermittler auf Basis der zu bestellenden Waren eines Händlers zur Auswahl eines geeigneten Lieferanten herangezogen wird (vgl. [Wes07]).

Eine Weiterentwicklung dieser Vorgehensweise besteht in der Annotation der Prozessbeschreibung auf technischer Ebene und eine automatische Transformation des modellierten fachlichen Prozessmodells, so dass das fachliche Prozessmodell in seiner ursprünglichen Form bestehen und für eine Wiederverwendung oder Weiterentwicklung auf der fachlichen Ebene erhalten bleibt (vgl. z. B. [Wut10]). Die Vermengung von fachlicher und technischer Ausführungslogik bleibt damit auf die ausgeführten Prozessinstanzen begrenzt und ist somit aus der Sicht des Entwicklers minimiert. Aus der Sicht der Ausführung werden jedoch trotzdem neue statische Konstrukte in den Prozess integriert, welche die Flexibilität und unter Umständen auch die Ausführung des Prozesses stark restriktieren können. Zum einen kann die Granularität der Verteilung auch in diesem Fall zur Laufzeit nicht mehr geändert werden. Zum anderen muss nun auch noch die Kommunikation mit dem verwendeten Vermittlungs- bzw. Verzeichnisdienst in der technischen Prozessbeschreibung festgehalten und zur Laufzeit durchgeführt werden. Fällt zum Beispiel der in Abbildung 5.10 dargestellte Vermittler aus, so kann auch die fachliche Prozessausführung nicht weiter voranschreiten. Durch die Vermengung von fachlicher und technischer Ausführungslogik kann zudem ggf. nicht festgestellt werden, ob es sich hierbei um ein fachliches oder ein technisches Problem handelt, was die Fehlerbehebung weiter erschwert. Wie bereits in Kapitel 3 gezeigt wurde, ist auch die Modellierung aller erdenkbarer fachlicher und technischer Eventualitäten keine vielversprechende Vorgehensweise für eine weitere Flexibilisierung, da die daraus entstehende Komplexität die Wartbarkeit und Änderbarkeit der prozessorientierten Anwendung erschwert.

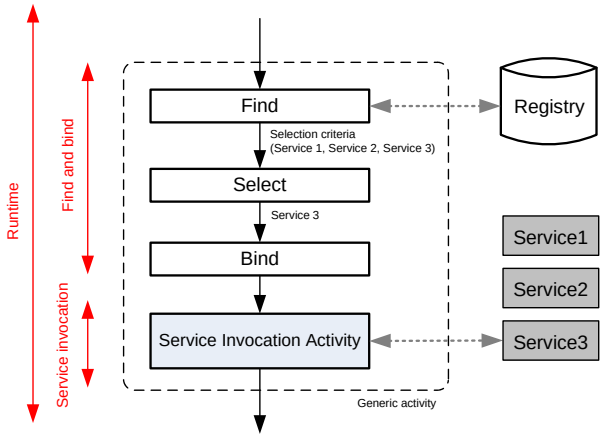


Abbildung 5.11: Dynamische Auswahl von Diensten durch zusätzliches „Find and Bind"-Element (nach (KB04b, KHC+05))

In Bezug auf die dynamische Auswahl einzelner elektronischer Dienste in zentral ausgeführten WS-BPEL-Prozessen beschreiben KARASTOYANOVA et al. [KB04b, KHC+05] eine Vorgehensweise zur Integration von komplexen Suchanfragen als Teil der Aktivitätsbeschreibung. Hierbei wird ein neues Sprachkonstrukt (das „Find and Bind"-Element) eingeführt, welches den eigentlichen Dienstaufruf gemeinsam mit einer vorgelagerten Verzeichnisabfrage innerhalb einer generischen Aktivität kapselt und so bei Bedarf ein dynamisches Binden auf Basis benutzerdefinierter Richtlinien erlaubt. Der Aufbau dieses Elements und die Vorgehensweise für die Auswahl eines konkreten Dienstes zur Laufzeit des Prozesses sind beispielhaft in Abbildung 5.11 dargestellt. Hierbei werden auf Basis der im Prozess beschriebenen Schnittstelle zunächst passende Dienstinstanzen durch einen Verzeichnisdienst gesucht. Im weiteren Verlauf wird einer der funktional gleichwertigen Dienste nach benutzerspezifizierten Kriterien ausgewählt und die URL der gewählten Dienstinstanz dem folgenden Aufruf zugrunde gelegt. Durch die Kapselung der Auswahl und des Aufrufs in einer Aktivität können Fehler beim Dienstaufruf somit besser behandelt werden, indem die „Find and Bind"-Phase ggf. noch weitere Male durchlaufen wird. Zudem können durch die explizite Formulierung von Regeln zur Dienstauswahl diese bei Bedarf sogar noch zur Laufzeit durch den Benutzer modifiziert werden [KB04b, KHC+05]. Obwohl auch bei diesem Ansatz die Auswahl von Ressourcen technisch in der fachlichen Prozessbeschreibung verankert ist, lässt der Ansatz mehr Spielraum für dynamische Anpassungen zur Laufzeit. Für die Umsetzung ist jedoch eine Anpassung der Prozessbeschreibungssprache (in diesem Fall WS-BPEL) und der ausführenden Prozess-Engine notwendig.

Eine dynamische Auswahl von Ausführungseinheiten ist für die Flexibilisierung verteilter Prozessausführungen sehr relevant. Die Vermengung von fachlicher und technischer Ausführungslogik ist jedoch vor dem Hintergrund der steigenden Komplexität der Prozessbeschreibungen problematisch. So wäre zum Beispiel eine umfangreiche, ggf. redundante Beschreibung von Ersatzteil-

nehmern zu integrieren, um bei Ausfällen eines Partners innerhalb der Choreographie handlungsfähig zu bleiben. Zudem steht auf der Ebene der Choreographie hierbei eine ressourcenbezogene Verteilung auf der Ebene einzelner Aktivitäten als Granulat der Verteilung im Vordergrund, was nicht der Zielsetzung dieser Arbeit entspricht. Im Folgenden werden daher Ansätze und Strategien zur (möglichst dynamischen) Partitionierung eines Prozesses und zum Umgang mit den entsprechenden Partitionen gesondert betrachtet.

5.4 Verteilung durch Partitionierung von Prozessen

Liegt ein ausführbares Prozessmodell in einer bestimmten Notation vor, die von den potentiell an der Prozessausführung teilnehmenden Parteien einheitlich interpretiert und durch deren technische Ausführungseinheiten unterstützt wird, so kann eine Partitionierung des Prozesses und eine Verteilung der Partitionen auf die entsprechend ausgewählten Ausführungseinheiten vorgenommen werden. Im Folgenden werden für die Vorgehensweise der Partitionierung relevante Ansätze vorgestellt und hinsichtlich ihrer Flexibilität untersucht. Dabei steht insbesondere die Verteilung individueller Prozessinstanzen zur Laufzeit und die Realisierung unterschiedlicher Verteilungskonfigurationen im Vordergrund.

5.4.1 Exotica/FMDC

Bereits in frühen Arbeiten befassen sich die Autoren ALONSO et al. [AGK+95, AGK+96] mit einer umfassenden Unterstützung zur Einbindung von mobilen Geräten zur Offline-Bearbeitung von Prozessabschnitten. Sie schlagen hierfür in ihrem Projekt *Exotica/FMDC* (*FlowMark for Mobile and Distributed Clients*) zwei unterschiedliche Lösungsansätze vor: Die erste Möglichkeit umfasst die vorausschauende Auswahl von aktuell durch einen Benutzer zu bearbeitenden Aktivitäten und das Herunterladen aller hierfür relevanten Informationen auf das mobile Gerät, so dass auf diesem eine Stapelverarbeitung von (logisch unzusammenhängenden) Aufgaben möglich ist (*Batch Mode*). Hierbei ist es nur möglich, bereits als ausführbar gekennzeichnete einzelne Aktivitäten (im Zustand *Ready*) auszuwählen, zum Beispiel für die Offline-Bearbeitung anstehender Aktivitäten paralleler Pfade oder unterschiedlicher Prozessinstanzen [AGK+95, AGK+96].

Die Autoren beschreiben für die Verteilung eine genaue Vorgehensweise auf Basis eines Zustandsmodells des Prozesses, seiner Aktivitäten und seiner Daten in Zusammenhang mit einem Kommunikationsprotokoll für die Interaktion zwischen einer leichtgewichtigen Benutzungsschnittstelle auf dem mobilen Gerät (*Workflow Client*) und einer zentralen Prozess-Engine auf einem stationären Server. Vor einer Offline-Bearbeitung wird hierbei zunächst eine Synchronisation durchgeführt, bei welcher die ausgewählten Aktivitäten zur Verhinderung einer Mehrfachbearbeitung für andere Benutzer gesperrt werden. Konflikte beim Anfordern von Sperren werden mittels Serialisierung

behandelt. Zudem werden alle mit den ausgewählten Aktivitäten assoziierten Daten zwecks späterer Synchronisation markiert. Nach dem Herunterladen der Daten kann optional überprüft werden, ob alle für die Ausführung benötigten Anwendungen auf dem Mobilgerät verfügbar sind. Während der Offline-Bearbeitung der ausgewählten Aktivitäten werden alle (Zwischen-)Ergebnisse auf dem mobilen Gerät gespeichert und bei erneuter Verbindung mit dem zentral verwalteten Prozess synchronisiert. Bearbeitet ein Benutzer die Aktivitäten nicht in einem gewissen Zeitraum oder kann er die erarbeiteten Ergebnisse nicht rechtzeitig synchronisieren (z. B. aufgrund fehlender Netzwerkverbindungen), so werden die gesperrten Aktivitäten serverseitig für andere Benutzer freigegeben [AGK+95, AGK+96].

Im Kontrollfluss nachfolgende Aktivitäten können durch den mobilen Benutzer bei diesem Ansatz nicht ohne erneute Verbindung mit der zentralen Prozess-Engine ausgeführt werden. Aus diesem Grund weisen die Autoren auf die Möglichkeit des eigenständigen Bearbeitens des Prozesskontrollflusses durch eine eigene leichtgewichtige Prozess-Engine auf dem mobilen Gerät hin. Da mit dem von ihnen vorgeschlagenen ersten Ansatz jedoch nur bereits aktivierte Aktivitäten gesperrt und entnommen werden können, ist eine vollständige Erweiterung des Ansatzes in diese Richtung nicht möglich. Zudem weisen die Autoren auf Schwierigkeiten bei der Zusammenarbeit mehrerer Prozessbeteiligter hin, da z. B. aufgrund von noch fehlenden Daten Wartezustände auftreten können, die eine Abkopplung eines Teilprozesses zur Offline-Bearbeitung verzögern oder sogar gänzlich verhindern könnten. Daher schlagen sie als hybride Lösung vor, nur die Offline-Bearbeitung von einzelnen in sich abgeschlossenen *Blockaktivitäten* (*Simple Block Activities*) zu erlauben [AGK+95, AGK+96].

In diesem Ansatz erfolgt eine sehr datenzentrische Sicht auf die verteilte Ausführung von Prozessen mit einer sehr eingeschränkten Flexibilität zur Verteilung der Prozessausführung. Die Granularität der Verteilung ist auf einzelne Aktivitäten beschränkt. Dabei werden jedoch die inhärenten Eigenschaften und Probleme der Offline-Bearbeitung durch mobile Geräte besonders berücksichtigt, welche gewisse Einschränkungen und ggf. Verzögerungen während der Prozessausführung rechtfertigen. Beide Teilansätze adressieren jedoch nicht das Problem der Heterogenität mobiler Geräte und der entsprechend heterogenen Benutzungsschnittstellen.

5.4.2 MENTOR

Im Rahmen des klassischen Workflow-Managements stellen die Autoren des Ansatzes MENTOR (*Middleware for Enterprise-wide Workflow Management*) [WWWD96, WW97, MWW+98] eine Vorgehensweise zur Partitionierung und Verteilung von unternehmensinternen Prozessen vor. Hierbei wird davon ausgegangen, dass ein Unternehmen mehrere Workflow-Server besitzt, denen auf der Basis eines Organistionsmodells jeweils eine Menge von menschlichen Prozessteilnehmern zugeordnet ist. Der Kontroll- und Datenfluss von verteilt aus-

zuführenden Prozessen wird hierbei durch *State-* und *Activity-Charts* ausgedrückt und, auf dieser Aufteilung basierend, zur Entwicklungszeit der prozessorientierten Anwendung in mehreren Schritten partitioniert. Dazu werden zunächst die Aktivitäten des Prozesses den gewünschten Prozessteilnehmern bzw. ihren Rollen und auf der Basis des Organisationsmodells bestimmten Partitionen zugeteilt. An den Zerlegungspunkten des Prozessmodells werden dann in einer zweiten Phase Zustandsübergänge eingeführt, welche einen Wechsel der verantwortlichen Ausführungseinheit angeben. An diesen Zustandsübergangen wird schließlich zur Laufzeit die komplette Prozessinstanz an die nächste Ausführungseinheit transferiert. Eine Umverteilung der auf diese Weise bestimmten Verteilung ist nicht vorgesehen, womit Anpassungen während der Ausführung des Prozesses nicht mehr möglich sind (vgl. hierzu auch [Sch01]).

Ein besonderer Fokus der Arbeit liegt in der Gewährleistung einer zur zentralen Prozessausführung äquivalenten verteilten Ausführung. Die Autoren des Ansatzes schlagen hierbei neben einer *strikten Synchronisation*, welche eine Kommunikation des Prozesszustandes mit seinen Variablen nach der Ausführung einer einzelnen Aktivität erfordert, eine *gelockerte Synchronisation* vor, welche durch die Einführung von *Synchronisationseinheiten* als Menge von Aktivitäten eine autonome Bearbeitung von Prozessabschnitten ermöglicht. Hierbei werden durch den Prozessmodellierer individuelle Synchronisationspunkte vorgegeben, welche während des Vorgangs der Partitionierung des Gesamtprozesses als expliziter Nachrichtenaustausch zwischen den resultierenden Prozessabschnitten umgesetzt werden. Die Synchronisation der Prozesszustände wird dabei im Wesentlichen durch einen TP-Monitor unterstützt, welcher im Rahmen der Middleware-Infrastruktur die Kommunikation zwischen den beteiligten Ausführungseinheiten über persistente Warteschlangen realisiert [WWWD96].

MENTOR besitzt hinsichtlich der Festlegung der Granularität einen deutlich größeren Anpassungsspielraum als der zuvor genannte Ansatz. Die Verteilung kann hierbei jedoch ausschließlich zur Entwicklungszeit vorgenommen werden. Daher ist die hier angestrebte individuelle Verteilung einzelner Prozessinstanzen zur Laufzeit nicht möglich.

5.4.3 Ansatz von VAN DER AALST

Bei seiner Klassifizierung möglicher Verteilungsmodelle (vgl. Abschnitt 4.3.3) spricht VAN DER AALST der *Verteilung durch lose Kopplung* ein hohes Maß an Flexibilität zu. Die Aufteilung eines globalen Prozesses in mehrere lokale Prozessabschnitte, deren Realisierung und Ausführung vollständig unter der Kontrolle einer autonomen Partei erfolgt, führt bei dem von VAN DER AALST präsentierten Ansatz [vdA99] zu einer festgelegten Kommunikationsstruktur zwischen den Prozessbeteiligten auf der Basis eines untereinander abgestimmten Interaktionsprotokolls. Insgesamt sind hierfür die folgenden drei Schritte erforderlich:

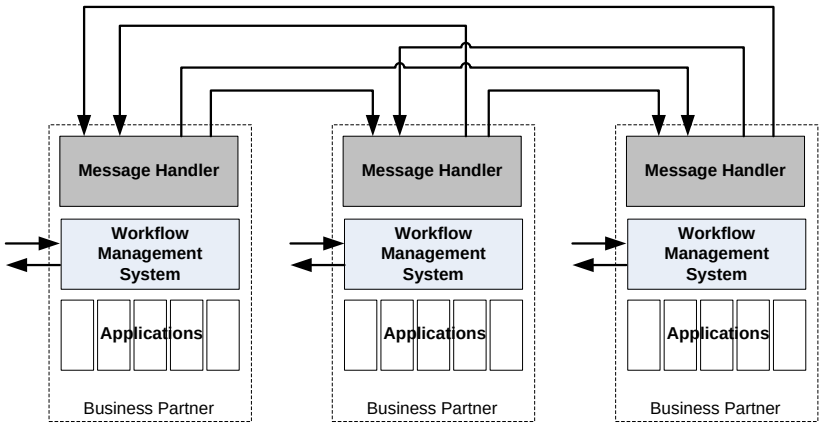


Abbildung 5.12: Architektur verteilten Prozessausführung durch lose Kopplung (vdA99)

1. Spezifikation eines Interaktionsprotokolls unter Beteiligung der teilnehmenden Parteien,

2. Entwicklung eines zum Interaktionsprotokoll konsistenten lokalen Prozesses,

3. Instantiierung und Ausführung des (Gesamt-)Prozesses.

Die ersten beide Schritte können dabei einen sehr großen Zeitraum einnehmen [vdA99] und sich auch gegenseitig bedingen. In der Regel ist hierbei ein großer Koordinationsaufwand zu erwarten. Zur Vereinfachung des Vorgangs schlägt Van der Aalst die Anfertigung und Nutzung von *Prozessvorlagen* vor, die in einem zentralen Prozess-Repository gehalten werden und bereits Anhaltspunkte für eine mögliche Partitionierung enthalten können. Die Ausführung des Prozesses ist dafür ohne weiteren Aufwand durch das Senden, Empfangen und Verarbeiten der zuvor spezifizierten Nachrichten zu realisieren. Es können dabei zudem fast ausschließlich bestehende Technologien eingesetzt werden. Abbildung 5.12 zeigt die von VAN DER AALST vorgeschlagene Architektur für eine verteilte Prozessausführung durch lose Kopplung. Eine einfache Komponente zur Verarbeitung der Koordinationsnachrichten (*Message Handler*) ist hier als Ergänzung bestehender Systeme ausreichend.

Die aus diesem Architekturansatz resultierende Flexibilität bezieht sich in erster Linie auf die resultierenden Freiheitsgrade für die an der Ausführung beteiligten Parteien. Durch die Kapselung der lokalen, privaten Prozessteile können diese unabhängig vom Gesamtprozess in ihrer Implementierung flexibel geändert werden. Die Aufteilung des globalen Gesamtprozesses auf die einzelnen Parteien geschieht jedoch statisch und eine Änderung der Verteilungskonfiguration ist in der Regel sehr aufwendig, da hierfür der gesamte Entwicklungsprozess neu durchlaufen werden muss. Zudem ist die Nachvollziehbarkeit des Gesamtprozesses durch die Verteilung und die individuelle Realisierung der Prozesspartitionen stark eingeschränkt.

5.4.4 MOBILE

Das Workflow-Management-System MOBILE erlaubt nach Erweiterungen von SCHAMBURGER [Sch01] und JABLONSKI et al. [JSH+01] die Verteilung der Prozessbearbeitung zur Unterstützung von Mitarbeitern an verschiedenen Arbeitslokationen. Dabei wird der Prozess in Partitionen aufgeteilt, welche auf verschiedenen Workflow-Servern möglichst in räumlicher Nähe zu den Bearbeitern installiert werden können. Ein wichtiges Teilziel hierbei ist die Integration *stark konnektierter* (d. h. über leistungsfähige stationäre Prozessmanagementsysteme angebundene Personen) und *schwach konnektierter* Prozessteilnehmer (d. h. über nur zeitweilig verbundene mobile Ausführungseinheiten angebundene Personen).

Die Bestimmung des Zielorts der Verteilung wird bei Schamburgers Ansatz durch den Benutzer selbst festgelegt. Dazu gibt es drei mögliche Vorgehensweisen [Sch01]:

- Zunächst ist durch eine Voreinstellung jeder Benutzer einer bestimmten Arbeitslokation zugeordnet.

- Diese Voreinstellung kann der Benutzer ändern, indem er vor Beginn der Prozessausführung Zeiträume spezifiziert, in denen er sich voraussichtlich an anderen Orten befindet.

- Befindet sich der Benutzer dennoch zur Laufzeit an einer nicht zuvor spezifizierten Arbeitslokation, so kann er manuell eine Umverteilung des Prozesses anfordern. Die Übertragung des Prozesses erfolgt in diesem Fall unter Aufsicht des Benutzers, so dass zum Beispiel auch ein schwach konnektiertes (mobiles) System zur Ausführung gewählt und durch den Benutzer ggf. aktiv verbunden werden kann.

Des Weiteren diskutiert SCHAMBURGER Spracherweiterungen für die Prozessbeschreibungssprache von MOBILE, damit ein Prozess selbst Einfluss auf seine Verteilung nehmen kann. Hierbei stehen die Definition von Partitionierungs- und Allokationsregeln im Vordergrund, um innerhalb der Prozessbeschreibung zu spezifizieren, unter welchen Umständen eine Partitionierung erlaubt ist und welche Benutzer(-gruppen) ggf. zuzuweisen sind [Sch01].

Die Verteilung bzw. Verwaltung der Prozessausführung geschieht bei der Erweiterung von MOBILE durch ein zentrales *primäres* (stark konnektiertes) Prozessmanagementsystem, welches mit anderen primären Prozessmanagementsystemen und sogenannten *sekundären* (schwach konnektierten) Prozessmanagementsystemen kommuniziert. Zur Berechnung einer geeigneten Verteilungskonfiguration wird ein Algorithmus vorgeschlagen, der zu Ausführungsbeginn des Prozesses eine statische Partitionierung festlegt, wobei einzelne Partitionen jedoch durch eine spätere *dynamische Umverteilung* an alternative Ausführungseinheiten transferiert werden können. Bei der

Übergabe des Kontrollflusses zwischen einzelnen Partitionen erfolgt dabei keine Migration der Instanzdaten des Prozesses, sondern die zuvor festgelegten Teilprozesse werden entfernt gestartet [BD99].

Bei der vorhergehenden Partitionierung des Prozesses liegt dabei ein besonderer Fokus auf der Integration schwach konnektierter Systeme. Zunächst wird eine Vorauswertung der Prozessdaten vorgenommen, um rechnerübergreifende Kommunikation möglichst zu vermeiden. Insbesondere sollen Kommunikationsprobleme und Verzögerungen zur Laufzeit des Prozesses von vornherein verhindert werden. Daher fordert SCHAMBURGER, dass die Synchronisation paralleler Prozesspfade nicht auf schwach konnektierten Systemen stattfinden darf. Zudem führt er das Konzept *autonomer Synchronisationshüllen* ein. Hierbei wird die Partitionierung des Prozesses so gestaltet, dass notwendige Synchronisationen möglichst innerhalb der Grenzen einer Ausführungsumgebung durchgeführt werden können [Sch01].

Im Gegensatz zu anderen Ansätzen aus dem Bereich der Anwenderintegration verfügt die erweiterte Variante von MOBILE über eine relativ hohe Flexibilität. Die (eingeschränkte) Integration diskonnektierter Ausführungseinheiten sowie die Möglichkeiten, Prozesspartitionen zur Laufzeit neu zuzuweisen, erlauben eine vielfältige Anpassung an dynamische und mobile Umgebungen. Es wird jedoch durch das zentrale Architekturkonzept nur eine zentralisierte Verteilung bzw. Ausführung der Prozessfragmente unterstützt. Das Konzept ist zudem stark auf die unternehmensinterne Anwenderintegration und die Verwaltung von zentralen Arbeitslisten ausgelegt. Der Grund hierfür liegt in der beschleunigten Ausführung des Prozesses durch die angenommene Konkurrenz von Benutzern um die Bearbeitung von Aufgaben. Dieser Ansatz ist im Fall von Prozessen mit einem (hohen) Anteil an automatisch ausgeführten Aktivitäten (wie im Fall von dienstbasierten Prozessen) nicht uneingeschränkt übertragbar, da Softwareanwendungen sich nicht „freiwillig“ für eine Aufgabe melden, sondern in der Regel (passiv) ausgewählt und zugewiesen werden. Eine weitere Ergänzung des Ansatzes um ein aktives Anbieten von Diensten (und menschlichen Benutzern) durch ein spezielles Verzeichnis oder einen Vermittler, der die aktuelle Auslastung von Ressourcen überwacht, ist jedoch nicht ausgeschlossen.

5.4.5 Ansatz von BARESI, MAURINO und MODAFFERI

Die Autoren BARESI, MAURINO und MODAFFERI adressieren in ihren Arbeiten die dezentrale Ausführung von WS-BPEL-Prozessen durch eine Föderation von heterogenen mobilen Geräten [BMM04, MM05b, MM05a, BMM06]. Aufbauend auf einer UML-Graphtransformation stellen die Autoren einen formalen Ansatz für die automatische Partitionierung von WS-BPEL-Prozessen vor. Hierbei erfolgt eine Auftrennung des WS-BPEL-Prozesses in eine Menge von aufeinander abgestimmter Teilprozesse, welche um die notwendigen Kommunikationsmechanismen ergänzt werden, um sowohl die Übergabe von Daten als auch den Kontrollfluss zwischen den ausführenden Parteien des Gesamt-

prozesses zu organisieren. Dabei wird insbesondere angestrebt, dass die an der Ausführung teilnehmenden Geräte möglichst unabhängig voneinander operieren können.

Die grundlegende Idee eines weitergehenden Ansatzes [BMM06] ist die Spezifikation eines Metamodells für WS-BPEL, welches explizit diejenigen Konstrukte der Prozessbeschreibungssprache beinhaltet, welche für die Partitionierung von Prozessen relevant sind. Das Metamodell bildet somit eine Abstraktionsebene für die Partitionierung. Dabei werden die tatsächlichen Sprachelemente von WS-BPEL klassifiziert (z. B. in atomare und strukturierte Aktivitäten) und über ihre Klassenzuordnung mit einer Menge von Partitionierungsregeln verknüpft, welche den Informationsaustausch zwischen den beteiligten Parteien und die Behandlung von Fehlern, Kompensation und Ereignissen adressieren. In Abhängigkeit der den einzelnen Aktivitäten (manuell) zugeordneten Ausführungseinheiten werden auf Basis dieser Partitionierungsregeln Prozessfragmente erzeugt, welche neben den fachlichen Aktivitäten auch neue Konstrukte für die Verteilung des Kontrollflusses, die Synchronisation von Prozessdaten und die Behandlung von Fehlern und Ereignissen enthalten. Die entstehenden Prozessfragmente sind dadurch weitestgehend auf standardkonformen WS-BPEL-Prozess-Engines ausführbar [BMM06].

Für die Durchführung der Delegation von Aufgaben werden konkret spezielle Aktivitäten eingeführt, welche den Kontrollfluss an eine andere Ausführungseinheit weitergeben (*Do Start*) und von einer anderen Ausführungseinheit entgegennehmen können (*Receive Start*). Dabei entspricht *Do Start* syntaktisch einer *Invoke*-Aktivität und *Receive Start* einer *Receive*-Aktivität in WS-BPEL, so dass durch eine entsprechende Transformation wieder standardkonforme Prozesse entstehen. Hinzu kommen die Aktivitäten *Remote Assign* und *Receive Assign* zur Synchronisation von Daten, welche zur Auswertungen von Bedingungen benötigt werden. Dieses erlaubt auch die teilweise Aufspaltung von komplexen Kontrollflussstrukturen, wie zum Beispiel von Schleifen. Durch die neue Aktivität *Reply End* wird schließlich der Kontrollfluss an den Prozessinitiator zurückgegeben. Die Nutzung globaler Variablen ist in diesem Ansatz nicht erlaubt [MM05a].

Die Transformation eines Prozesses kann sowohl zur Entwicklungs- als auch zur Laufzeit eines Prozesses stattfinden, wobei eine Partitionierung zur Laufzeit und die damit verbundenen Schwierigkeiten (wie zum Beispiel die Aktualisierung der laufenden Prozessinstanz oder das dynamische Deployment von Web-Service-Schnittstellen) nicht explizit beschrieben werden. Werden hierbei jedoch alle Partitionen zur gleichen Zeit bestimmt, so ist eine Umverteilung der einmal bestimmten Verteilungskonfiguration nicht mehr ohne weiteres möglich. Schreitet die Partitionierung jedoch schrittweise während der Ausführung voran, so könnte eine deutlich höhere Flexibilität des Ansatzes erzielt werden.

5.4.6 MobiWork und CiAN

Die teilweise gemeinsamen Autoren der Projekte *MobiWork* [HSH+06, SHH+07] und *CiAN* [SRF06, SRG08] adressieren die Ausführung von Prozessen über mehrere Geräte in mobilen Ad-Hoc-Netzwerken. Hierzu wird bei MobiWork zunächst eine prozessorientierte Anwendung in Teilprozesse (*SubPlans*) fragmentiert und diese auf Basis der Fähigkeiten mobiler Geräte und räumlich-zeitlicher Rahmenbedingungen geeigneten Ausführungseinheiten zugewiesen. Die initiale Partitionierung eines Prozesses und die entsprechende Allokation von Ressourcen findet dabei in einer der Ausführung vorgelagerten *Planungsphase* durch ein zentrales System statt, welches Kenntnis über die Teilnehmer seiner (lokalen) Gruppe besitzt und mit diesen über eine stabile Netzwerkverbindung verbunden ist (vgl. auch [MCA+06]). Dabei wird in einer vordefinierten Prozessbeschreibung spezifiziert, an welchem Ort und zu welcher Zeit die auszuführenden Aufgaben bearbeitet werden müssen. Im Rahmen der Planung können daher zum Beispiel nicht zwei Aufgaben derselben Ausführungseinheit zugewiesen werden, wenn sie gleichzeitig an verschiedenen Orten erledigt werden müssen. Genauso können nicht zwei aufeinander folgende Aktivitäten zwei unterschiedlichen Ausführungseinheiten zugeordnet werden, wenn diese räumlich zu weit voneinander entfernt sind, um die Ergebnisse der Ausführung rechtzeitig weiterzugeben. Die eingeschränkten Ressourcen mobiler Geräte werden dabei durch die Anwendung heuristischer Algorithmen zur Auswahl geeigneter Ausführungseinheiten berücksichtigt [HSH+06, SHH+07]. Nach Abschluss der Planungsphase kann sich die Gruppe auflösen und jeder Teilnehmer kann mit der Bearbeitung der ihm zugewiesenen Aufgaben beginnen. Bei Auftreten eines Fehlers während der (dezentralen) Ausführung kann zudem eine lokale *Neuplanung* stattfinden, wobei jedoch nur das jeweilige lokale Teilwissen über die Prozessausführung für eine Umverteilung des Teilprozesses zur Verfügung steht [HSH+06].

Ähnlich wie MobiWork ist auch das weiterführende Projekt der Autoren *CiAN* (*Collaboration in Ad-hoc Networks*) [SRF06, SRG08] auf die Unterstützung einer kooperativen Prozessausführung in mobilen Ad-hoc-Netzwerken ausgelegt. Im Gegensatz zu MobiWork wird bei CiAN jedoch nicht die verteilte Ausführung einer einzelnen Orchestrierung von (mobilen) Diensten, sondern die Umsetzung von Choreographien in mobilen Umgebungen unterstützt. Bei der Choreographie wird dabei die Art und Weise der Prozessbeteiligung durch das Bereitstellen von entsprechenden Schnittstellen und die Festlegung eines Kommunikationsprotokolls zur Entwicklungszeit vorgenommen (vgl. Abschnitt 3.4.5). Die Autoren des Ansatzes teilen die Konzepte von CiAN daher in zwei separate Phasen ein [SRG08]: Die erste Phase umfasst das Planen der Prozessausführung mit der Partitionierung, der Zuweisung der Prozessabschnitte und der Festlegung für die Choreographie erforderlichen Schnittstellen. Aufgrund der Ähnlichkeit der Zielsetzung wird hierfür teilweise auf die Realisierung von MobiWork verwiesen, so dass auch die Planungsphase von CiAN auf der Annahme einer temporär fixen Infrastruktur

mit zentralem Koordinator beruht. Die zweite Phase umfasst die Ausführung der durch die Choreographie festgelegten Prozessteile. Hierfür schlagen die Autoren ein Publish-Subscribe-basiertes Verfahren mit Gossiping zur dezentralen Übergabe des Kontrollflusses und zur Synchronisation von Prozessdaten vor [SRF06, SRG08].

Die durch die Choreographie festgelegte Kommunikation kann somit sowohl bei MobiWork als auch bei CiAN unter besonderer Berücksichtigung der eingeschränkten Kommunikationsmöglichkeiten in mobilen Ad-hoc-Netzen ausgeführt werden. Eine Anpassung der vorab festgelegten Zuweisung von Aktivitäten zu Prozesspartitionen und deren Zuordnung zu Ausführungseinheiten kann bei CiAN jedoch nicht, und bei MobiWork nur sehr eingeschränkt vorgenommen werden. Ein Vorteil dieser Ansätze ergibt sich daher nur für Prozesse, welche oft in gleicher Weise wiederholt ausgeführt werden. Für die individuelle Verteilung von einzelnen Prozessinstanzen ist der Aufwand, der mit der zentralen Vorausplanung der Ausführung verbunden ist, jedoch relativ hoch. Fehlende Möglichkeiten zur dynamischen Umverteilung schränken die Flexibilität dieser Ansätze zudem weiter ein.

5.4.7 Ansatz von KHALAF und LEYMANN

Das Ziel der Arbeit von KHALAF und LEYMANN [KL06, Kha08] ist es, die Zuweisung von Aktivitäten eines WS-BPEL-Prozesses zu prozessausführenden Systemen zu unterstützen. Hierzu schlagen sie vor, den Vorgang der Partitionierung eines Prozesses von der Modellierung der fachlichen Prozesslogik zu entkoppeln. Das originäre WS-BPEL-Prozessmodell wird hierzu zunächst in eine modifizierte Version von WS-BPEL mit der Bezeichnung *BPEL-D* übersetzt. Das BPEL-D-Prozessmodell wird dann zusammen mit den assoziierten WSDL-Dateien und einer Beschreibung, welche die Zuordnung von Aktivitäten zu Ausführungseinheiten spezifiziert, durch einen automatischen Transformationsansatz in individuelle, standardkonforme WS-BPEL-Prozesse mit jeweils einer eigenen WSDL-Datei zerlegt und eine globale Interaktionsbeschreibung für das Zusammenspiel der einzelnen Prozessteile erzeugt [KL06].

Für die Zwischenrepräsentation BPEL-D werden Datenabhängigkeiten durch explizite *Data Links* repräsentiert und Einschränkungen definiert, welche für eine spätere Verteilung des Prozesses von Bedeutung sind. Zum Beispiel dürfen die Aktivitäten „Receive“ und „Reply“ zur Umsetzung asynchroner Kommunikationsparadigmen nicht auf zwei verschiedene Ausführungseinheiten verteilt werden [KL06]. Auf der Basis einer (manuellen) Zuordnung von Ausführungseinheiten (z. B. durch Swimlane-Diagramme) werden die teilnehmenden Parteien sowie deren Schnittstellen zur Intraprozesskommunikation identifiziert und auf WSDL-Dateien abgebildet. Jeder individuelle Teilprozess wird dabei um entsprechende Partner-Links, Variablen und eine Korrelationsmenge ergänzt. Die Übergabe des Kontrollflusses wird über das Einfügen von zusätzlichen „Invoke“-“Receive“-Aktivitätspaaren erreicht, so dass die für die Ausführung von Folgeaktivitäten verantwortliche

Ausführungseinheit als Dienst durch die aktuelle Ausführungseinheit aufgerufen werden kann. Durch die „Receive"-Aktivität wird hierbei jeweils eine neue lokale Prozessinstanz des Teilprozesses erzeugt und durch die definierte Korrelationsmenge mit der Ausführung des globalen Gesamtprozesses verknüpft [KL06].

Der Ansatz von Khalaf und Leymann erreicht ein zur zentralen Ausführung semantisch äquivalentes Laufzeitverhalten verteilt ausgeführter WS-BPEL-Prozesse weitestgehend unter Nutzung bestehender Technologien. Die dienstorientierte Anbindung unterschiedlicher Prozessteilnehmer durch den im WS-BPEL-Prozess definierten Aufruf weiterer Teilprozesse kann weiter flexibilisiert werden, indem Ausführungseinheiten mit ihren Teilprozessen dynamisch durch die Verwendung eines Verzeichnisdienstes ausgewählt und erst zur Laufzeit eingebunden werden. Die Verteilungskonfiguration in Bezug auf die einzelnen Prozessfragmente wird bei diesem Ansatz jedoch zur Entwicklungszeit bestimmt und kann zur Laufzeit höchstens unter großem Aufwand angepasst werden, da zur Anpassung der gesamte Ablauf der Transformation des funktionalen Prozesses, des Deployments und der Bereitstellung der Schnittstellen bei den teilnehmenden Ausführungseinheiten erneut durchlaufen werden müsste. Für die Anpassung einzelner Prozessinstanzen eignet sich dieser Ansatz daher nur bedingt.

5.4.8 Ansatz von WUTKE, MARTIN und LEYMANN

Mit dem Ziel der dezentralen Ausführung von regulären WS-BPEL-Prozessen stellen die Autoren WUTKE, MARTIN und LEYMANN [MWL08, WML09a, WML09b, Wut10] einen Ansatz zur nicht-invasiven Anpassung von Prozessmodellen an ihre jeweilige Ausführungsumgebung vor, bei welchem sowohl die Orchestrierungslogik als auch die Schnittstellen der verwendeten Dienste erhalten bleiben können [Wut10]. Der Ansatz umfasst ein Verfahren zur (Vor-) Verarbeitung der Prozessmodelle, ein Verfahren zu deren automatischer Partitionierung sowie die Architektur eines verteilten Prozessmanagementsystems als Laufzeitumgebung für deren Ausführung auf der Basis von *Tuplespace*-Konzepten [Wut10]. Abbildung 5.13 zeigt einen Überblick über die Vorgehensweise, um auf der Basis eines regulären WS-BPEL-Prozesses eine dezentrale Ausführung zu ermöglichen [MWL08]:

1. Zunächst erfolgt die Modellierung eines regulären BPEL-Prozesses mit bestehenden Werkzeugen zur Prozessmodellierung. An dieser Stelle wird kein Wissen über eine mögliche Verteilung integriert, wodurch das ursprüngliche Prozessmodell erhalten bleibt und wiederverwendet werden kann.

2. Im Anschluss wird eine Prozess-Segmentierung durch Annotation des Prozessmodells mit Zuordnungsinformationen vorgenommen, um eine Zuweisung von Prozesspartitionen an Prozessteilnehmer zu erlauben. Die Dekomposition des Prozesses in einzelne funktionale Einheiten

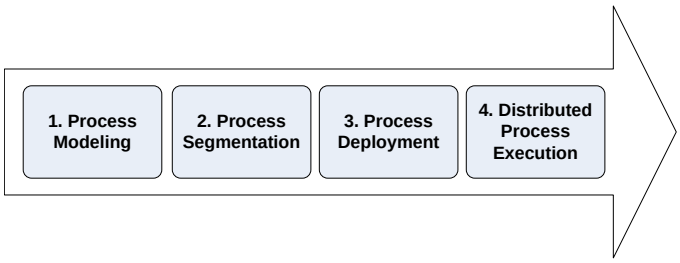


Abbildung 5.13: Vorgehensweise zur dezentralen Ausführung von WS-BPEL-Prozessen (MWL08)

ist (optional) automatisierbar und kann unter Verwendung von Kostenfunktionen (z. B. zur Berücksichtigung der Netzwerkdistanz, der Ausführungswahrscheinlichkeiten von Aktivitäten oder der zu erwartenden Datenvolumina) verschiedene Strategien zur Verteilung der Prozesslogik unterstützen. Resultat dieses Schritts ist ein Deployment-Deskriptor, welcher das Prozessmodell auf die vorliegende technische Infrastruktur abbildet.

3. Es erfolgt eine Transformation in ein dezentralisiert ausführbares Modell (*Executable Workflow Network*, kurz *EWFN*), welches nativ auf der vorgeschlagenen Tuplespace-Infrastruktur ausführbar ist. Hierzu ist die explizite Beschreibung der Interaktionen erforderlich, welche zwischen den vorgesehenen Prozessteilnehmern erfolgen müssen, um ein zur operationalen Semantik von WS-BPEL entsprechendes Ausführungsverhalten zu erhalten. Die resultierenden individuellen Prozess-Segmente werden auf den ausgewählten Prozess-Engines installiert.

4. Nach einer Instantiierung des Prozesses ist eine dezentrale Ausführung des WS-BPEL-Prozesses nach den zuvor festgelegten Strategien möglich.

Die dargestellte Vorgehensweise ist an der Ausführung von Produktionsprozessen orientiert, welche die Festlegung der Partitionierung eines Prozesses und die Zuordnung zu Ausführungseinheiten zur Deployment-Zeit motivieren [Wut10]. Durch die spätere Transformation des originären Prozessmodells kommt es in der Modellierungsphase des Prozesses nicht zu einer unerwünschten Vermischung von fachlicher und technischer Ausführungslogik, was die inhaltliche Anpassbarkeit des Prozessmodells vereinfacht. Die verschiedenartige Ausführung von einzelnen Prozessinstanzen in Hinblick auf ihre Verteilung ist hierbei jedoch nicht vorgesehen. Zudem ist eine Anpassung der zur Zeit des Deployments festgelegten Verteilungskonfiguration durch die statische Verteilung der Prozesssegmente zur Laufzeit nicht mehr möglich. Die Anpassungsfähigkeit der Prozessausführung zur Laufzeit ist somit bei diesem Ansatz begrenzt.

5.5 Verteilung durch Migration von Prozessinstanzen

Die Migration von laufenden Prozessinstanzen stellt eine Metapher für die natürliche Arbeitsweise von Personen dar, um gemeinsam auf der Basis eines vorgegebenen Vorgehensplans eine aus mehreren Teilschritten bestehende Aufgabe durchzuführen [CRW98]. Ein bekanntes Beispiel hierfür ist das Dokumentenmanagement, bei dem eine Aktenmappe mit einer Beschreibung der durchzuführenden Aufgaben und den hierfür erforderlichen Informationen zur Bearbeitung eines Vorgangs von einem Mitarbeiter an den nächsten weitergereicht wird. Ein anderes Beispiel ist der Aufbau einer komplexen Maschine nach einer strukturierten Aufbauanleitung, bei welcher der für den Aufbau verantwortliche Mitarbeiter während seiner Arbeit zur Durchführung einer anderen Aufgabe abberufen wird. Als Reaktion auf dieses Ereignis kann er seinen bisherigen Fortschritt bei der Bearbeitung der Teilaufgaben vermerken und die Aufbauanleitung an eine andere Person weitergeben, welche den Aufbau der Maschine an der entsprechenden Stelle fortsetzt. In beiden Fällen kann dabei die als nächstes verantwortliche Person durch den aktuellen Bearbeiter bestimmt oder aber durch globale Regelungen vorgegeben werden. Das hinter der technischen Realisierung dieser Vorgehensweise stehende softwaretechnische Entwicklungsmuster ist das *Prozessmuster* mit den dieser Sichtweise entsprechenden Metaphern der „Vorgangsmappe“ und des „Laufzettels“ (vgl. [Gry96, Zül98]).

Mit der Übergabe des Kontrollflusses werden bei einer Migration von Prozessinstanzen entweder nur Instanzdaten des laufenden Prozesses oder aber zusätzlich die gesamten Kontrollflussinformationen in Form des Prozessmodells zwischen unterschiedlichen Ausführungseinheiten ausgetauscht. Teilweise ist daher eine statische Vorverteilung des Prozessmodells notwendig. Ansätze, bei denen die gesamte Prozessbeschreibung migriert wird, sind durch Einflüsse verteilter Transaktionen (vgl. [GR92]) und mobiler Softwareagenten gekennzeichnet, bei denen ausführbarer Programmcode auf andere Ausführungsplattformen transferiert werden kann (vgl. Abschnitt 3.9.4) [Sch01, CR04]. Die Migration von Prozessinstanzen auf Anwendungsebene ist dabei abzugrenzen von allgemeinen Ansätzen zur *Code-Migration* (vgl. [TvS03]) und der ebenfalls als Migration bezeichneten Aktualisierung, Umstellung oder Portierung von Daten und/oder Softwareanwendungen (vgl. [Mor07, Hil07, BvR08]). In diesem Abschnitt werden insbesondere solche bestehenden Ansätze untersucht, welche die Migration von Prozessinstanzen zur Flexibilisierung verteilter Prozessausführung einsetzen.

5.5.1 Ansatz von CICHOCKI und RUSINKIEWICZ

Basierend auf einigen früheren Arbeiten schlagen die Autoren CICHOCKI und RUSINKIEWICZ [CR97, CRW98, CR04] sogenannte *migrierende Workflows* als Repräsentation von Prozessinstanzen vor, die mit ihrer Prozessbeschreibung, ihren Daten und ihrem aktuellen Ausführungszustand

von einer Ausführungseinheit zur nächsten transferiert werden. Während der Ausführung werden durch die Prozessinstanzen hierbei Anfragen nach benötigten Funktionalitäten spezifiziert. Diese werden in Abhängigkeit von lokalen Regeln und Richtlinien durch die verfügbaren Ausführungseinheiten bereitgestellt und in einem globalen Verzeichnis offeriert, so dass dieses zur Laufzeit abgefragt werden kann. Jede Aktivität eines Prozesses kann somit mit einem bestimmten Ort bzw. einer bestimmten Ausführungsumgebung assoziiert werden. Eine Migration findet demnach immer dann statt, wenn die assoziierte Ausführungsumgebung nicht der aktuellen Ausführungsumgebung entspricht. Neben den jeweils aktuellen Instanzdaten werden dabei zudem Logdaten zum Ausführungsverlauf weitergereicht, um die Wiederherstellung des Prozesses im Fehlerfall, die Abrechnung oder die Analyse der Prozessausführung zu unterstützen [CRW98, CR04].

Bei jeder Prozessinstanz kann nach CICHOCKI und RUSINKIEWICZ die Menge der Aktivitäten sowie die Zuordnung von Ausführungsumgebungen dynamisch zur Laufzeit des Prozesses geändert werden. Dazu schlagen sie eine Implementierung basierend auf *ECA*-Regeln vor (vgl. Abschnitt 3.6), welche abhängig von Prozess- und Zustandsdaten den Aufruf von (lokalen) Ressourcen auslösen. Basierend auf deren Rückgabewerten können dann weitere Aktionen ausgeführt werden, wie zum Beispiel das Tätigen weiterer Aufrufe oder die Migration der Prozessinstanz. Die Anfrage einer bestimmten Funktionalität durch eine Prozessinstanz kann dabei ebenfalls als Ereignis einer ECA-Regel angesehen werden. Der Bedingungsteil der lokalen Richtlinie kann alle anderen für die Ausführung relevanten Faktoren enthalten, so dass durch Auswertung der Regel entweder die Bearbeitung der Anfrage durch die lokale Ausführungsumgebung ausgelöst oder die Anfrage abgelehnt werden kann [CRW98]. Des Weiteren können die beteiligten Ausführungseinheiten Einfluss auf den weiteren Verlauf des Prozesses nehmen. Als wesentliche Beispiele nennen CICHOCKI und RUSINKIEWICZ die Akquisition von fehlenden Daten, die Umleitung der Anfrage zu einer anderen Ausführungsumgebung (Migration) oder das Einfügen und Ausführen eines Subprozesses an der Stelle der auszuführenden Aktivität [CR97, CRW98, CR04].

Ein besonderer Fokus der Arbeit liegt auf der Untersuchung, Spezifikation und Durchsetzung transaktionaler Eigenschaften von migrierenden Prozessinstanzen [Cic99, CR04]. Um den Verlust bereits geleisteter Arbeitsschritte und das ineffiziente Sperren von Ressourcen über mehrere Aktivitäten hinweg zu vermeiden, bieten sich nach CICHOCKI und RUSINKIEWICZ vor allem Forward-Recovery-Mechanismen und Kompensationsstrategien für die verteilte Ausführung von Prozessen an [CR04] (vgl. Abschnitt 2.4.5). Der Zugriff auf globale Daten erfolgt bei ihrem Ansatz über eine Serialisierung der Zugriffe einzelner Prozessinstanzen bzw. paralleler Pfade mit Hilfe von hierarchisch geordneten Zeitstempeln (vgl. [BG80, CR04]). Sollen parallele Pfade des Prozesses von verschiedenen Ausführungseinheiten bearbeitet werden, so wird eine komplette Kopie der laufenden Prozessinstanz erzeugt und mit den entsprechenden Laufzeitdaten auf die ausgewählten Ausführungseinheiten ver-

teilt. Die Konsistenz von Daten, welche in mehreren parallelen Pfaden zugegriffen werden, wird über die Spezifikation von *Datenklassen* erreicht, welche angeben, wie die nebenläufig bearbeiteten Variablen am Ende der parallelen Bearbeitung synchronisiert werden sollen. Dabei werden verschiedene Möglichkeiten zur Einschränkung der Serialisierbarkeit als Korrektheitskriterium der Datenkonsistenz und das daraus resultierende Flexibilisierungspotential für die verteilte Ausführung von Prozessinstanzen diskutiert.

Der Ansatz von Cichocki und Rusinkiewicz eröffnet ein großes Maß an Flexibilität, da jede einzelne Prozessinstanz zur Laufzeit schrittweise partitioniert und auf Basis von globalen Richtlinien in Form des Prozesses sowie auf Basis lokaler Richtlinien in Form von individuellen Policies der Ausführungseinheiten verteilt werden kann. Somit können sowohl die Vorgaben des Prozessmodellierers als auch der ggf. autonomen Ausführungseinheiten weitestgehend berücksichtigt werden. Das auf ECA-Regeln basierende Konzept erlaubt zudem eine zeitnahe Reaktion auf die aktuellen Bedingungen der beteiligten Ausführungssysteme und eine Automatisierbarkeit des Anpassungsvorgangs. Sowohl die Korrektheit der Ausführung als auch die Erkennung und Behandlung von Fehlerfällen werden durch das vorgeschlagene Transaktionskonzept [CR04] angemessen adressiert.

Problematisch erscheint hingegen in Anbetracht der in Abschnitt 4.4 identifizierten Anforderungen, dass für die Spezifikation der zu verteilenden Prozessmodelle eine proprietäre Prozessbeschreibungssprache vorgeschlagen wird, welche speziell auf den hier vorgestellten Verteilungsansatz ausgerichtet ist. Diese ist zum einen noch sehr an dem Konzept mobiler Agenten und dem *aktiven* Suchen und Anfragen von Ressourcen ausgerichtet, welches nicht der in dieser Arbeit verfolgten Sichtweise einer *passiven* Beschreibung von durchzuführenden Aufgaben entspricht. Bestehende Prozessmodelle in unter Umständen standardisierter Repräsentation müssten somit zunächst in die vorgeschlagene Form gebracht werden, bevor sie auf diese Weise flexibel verteilt ausgeführt werden können. Eine spontane Verteilung zuvor zentral ausgeführter Prozesse ist somit nicht ohne weiteres möglich. Zudem werden auch in diesem Ansatz fachliche und technische Beschreibungen vermischt, was eine rein fachliche Weiterentwicklung der Prozessmodelle erschwert.

5.5.2 Ansatz von VAN DER AALST

Aus Gründen der Flexibilität favorisiert VAN DER AALST bei seiner Klassifizierung möglicher Verteilungsmodelle (vgl. Abschnitt 4.3.3) die *Verteilung durch lose Kopplung* und die *Verteilung durch Übergabe von Prozessinstanzen*. Letztere erlaubt die Verteilung eines Prozessmodells an verschiedene Ausführungseinheiten, die Instantiierung des Prozessmodells auf einer dieser Ausführungseinheiten und bei Bedarf den Transfer der laufenden Prozessinstanz auf ein anderes System [vdA99]. Hierfür spezifiziert er Richtlinien, welche bestimmen, wann eine Prozessinstanz an welche andere Ausführungseinheit zugewiesen werden soll. Zentrale (teilwei-

se konfliktierende) Aspekte sind dabei insbesondere die Minimierung der Anzahl notwendiger Transferschritte, die gleichmäßige Auslastung aller Ausführungseinheiten, die Minimierung der Ausführungszeit und die Minimierung der Ausführungskosten [vdA99]. Basierend auf diesen Anforderungen und einem einfachen Zustandsmodell werden sechs Richtlinien definiert [vdA99]:

- **Persistent Transfer Policy**: Eine wartende oder blockierte Prozessinstanz wird migriert, wenn durch die Migration ein Zustand erreicht wird, in dem der Prozess weiter ausgeführt werden kann.
- **Strongly Persistent Transfer Policy**: Eine blockierte Prozessinstanz wird migriert, wenn durch die Migration ein Zustand erreicht wird, in dem der Prozess weiter ausgeführt werden oder zumindest auf seine Ausführung warten kann.
- **Time-Out Transfer Policy**: Eine Prozessinstanz wird migriert, wenn seit einer definierten Zeitspanne keine Aktivität ausgeführt wurde und durch die Migration ein Zustand erreicht wird, in dem der Prozess weiter ausgeführt werden kann oder zumindest auf seine Ausführung wartet.
- **Load-balancing Transfer Policy**: Eine zur Zeit nicht aktive Prozessinstanz wird migriert, wenn der Prozess nach der Migration nicht blockiert ist und durch die Migration eine gleichmäßigere Auslastung erreicht werden kann.
- **Cost-driven Transfer Policy**: Eine Prozessinstanz wird migriert, wenn die Kosten am neuen Ausführungsort geringer sind.
- **Intelligent Transfer Policy**: Diese Richtlinie stellt eine Zusammensetzung der oben genannten Strategien dar und zieht nach Möglichkeit auf Basis einer Vorhersage auch zukünftige Entwicklungen (z. B. der Auslastungssituation) in die Entscheidungsfindung mit ein.

In dem beschriebenen Ansatz müssen sich die an der Ausführung beteiligten Parteien zunächst über einen gemeinsamen Prozess einig werden. Um auch inhaltliche Anpassungen zu erlauben, schlägt VAN DER AALST daher die Verwaltung von Prozessmodellen in speziellen Prozess-Repositories vor, welche Vorlagen zur Ableitung neuer oder erweiterter Prozessmodelle in einem organisationsübergreifenden Kontext enthalten. Dabei können die Prozessmodelle auch über lokale Anpassungen ergänzt werden, welche später nur bei der Wahl eines bestimmten Ausführungsortes tatsächlich ausgeführt werden [vdA99].

Des Weiteren adressiert die Arbeit das Problem heterogener Ausführungssysteme unterschiedlicher Hersteller und das Fehlen einer einheitlichen Repräsentation zum Austausch von Zustandsinformationen. Um auf Zustandsinformationen zugreifen und erzeugen zu können, wird eine Architektur vorgeschlagen, welche die Prozessmanagementsysteme der potentiell teilnehmenden Parteien um eine zusätzliche Schicht zur Integration (*Swap-In*) bzw. zur

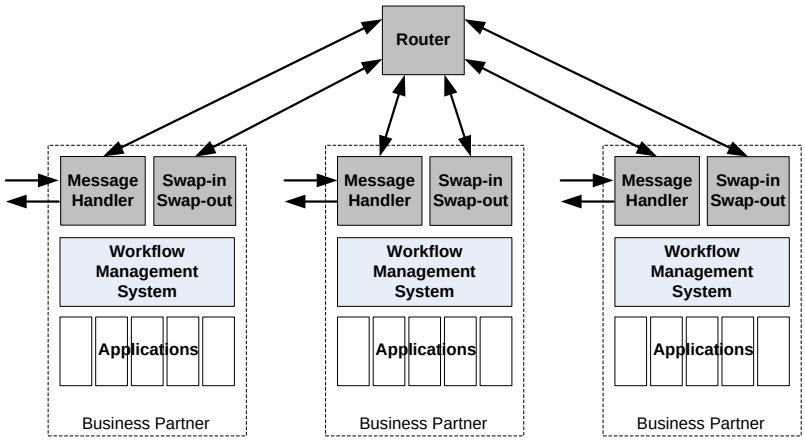


Abbildung 5.14: Architektur zum Transfer von Prozessinstanzen (vdA99)

Extraktion (*Swap-Out*) von laufenden Prozessinstanzen und ihrer Zustandsinformationen erweitert (vgl. Abbildung 5.14). Zudem existiert eine Komponente zum Weiterleiten von Ereignissen und Nachrichten, welche an eine bereits migrierte Prozessinstanz gerichtet sind.

Solange der Prozess an einem einzigen Ausführungsort bearbeitet wird, ist keine globale Kontrolle der Prozessausführung notwendig. Zur Durchführung der Migration wird hier jedoch eine zentrale Kontrollkomponente (*Router*) benötigt, welche den Transfer und den aktuellen Ort der Prozessinstanzen verwaltet und die Kommunikation zwischen den Kooperationspartnern organisiert (vgl. Abbildung 5.14). Durch die Vermeidung einer dezentralen Kommunikation werden bei diesem Ansatz viele Probleme in Hinblick auf die Interoperabilität der Systeme vermieden. Die bereits genannten Nachteile zentraler Infrastrukturen bezüglich Autonomie, Verantwortlichkeit und Ausfallsicherheit gelten hier jedoch entsprechend. Obwohl die Verteilung der Prozessinstanz zur Laufzeit vorgenommen werden kann, wird das Prozessmodell bei diesem Ansatz zudem bereits vor der Laufzeit verteilt. Dies begrenzt die Anpassung auf Ausführungssysteme, welche zu diesem Zeitpunkt schon berücksichtigt werden konnten. Eine weitergehende fachliche Anpassung des Prozessmodells wird dabei erschwert, da im Anpassungsfall die Änderung an alle beteiligten Systeme propagiert werden muss. Durch die Notwendigkeit, ganze Prozessmodelle vorsorglich speichern zu müssen, ist auch die Anwendbarkeit dieses Ansatzes für mobile Systeme mit eingeschränkten Ressourcen fraglich.

5.5.3 ADEPT Distribution

Das Forschungsprojekt ADEPT (*Application Development Based on Encapsulated Pre-Modeled Process Templates*) [BD97, RD98] hat zum Ziel, die Leistung und Skalierbarkeit von prozessorientierten Anwendungen durch die Aufteilung eines Prozesses auf mehrere Ausführungeinheiten (*Workflow Server*) zu optimieren. Dabei liegt ein besonderer Fokus auf der Entlastung des Kommunikationsnetzwerks und dementsprechend auf der Vermeidung von Kom-

munikationsvorgängen, die resultierend aus der Nutzung gewisser Netzwerkverbindungen (z. B. aufgrund der Zahlung von Nutzungsentgelten oder geringer Bandbreiten) hohe Kosten verursachen. Hierzu werden Prozessmodelle zur Entwicklungszeit der prozessorientierten Anwendung partitioniert und so auf die verfügbaren Ausführungeinheiten verteilt, dass möglichst viele Kommunikationsvorgänge innerhalb der lokalen Teilnetze der Ausführungsumgebung ablaufen können und somit der globale Kommunikationsaufwand minimiert wird.

Die Autoren BAUER und DADAM [BD00, BRD01] erweitern diesen auf einer statischen Zuordnung basierenden Ansatz um die Möglichkeit, die Zuordnung der Prozesspartitionen zu Ausführungseinheiten zur Laufzeit des Prozesses beeinflussen zu können. Die Motivation ihrer Arbeit ist, dass die Zuweisung der Prozessausführung im Kontrollfluss nachgelagerter Aktivitäten häufig erst während der Ausführung vorgelagerter Aktivitäten bestimmt wird. BAUER und DADAM schlagen daher die Nutzung von logischen Zuweisungsausdrücken vor, welche bei der Modellierung des Prozesses als Rahmenbedingungen der Ausführung definiert und bei der Ausführung des Prozesses evaluiert werden. Ein Beispiel ist die Zuweisungsregel, dass „Aktivität k durch eine Ausführungseinheit verwaltet werden soll, welche sich im selben Teilnetz befindet wie die Ressource, welche die Vorgängeraktivität x ausgeführt hat“. Die Autoren geben hierzu eine Reihe von typischen Zuweisungsausdrücken an, welche von ihrem Ansatz unterstützt werden [BD00].

Um zu vermeiden, dass nach der Ausführung jeder einzelnen Aktivität aufwendige Analysen zur Auswahl der nachfolgend zuständigen Ausführungseinheit durchgeführt werden müssen, schlagen BAUER und DADAM eine hybride Strategie aus statischer Vorberechnung der Verteilung und der dynamischen Integration von Laufzeitaspekten vor. Dabei wird auf der Basis eines Kostenmodells eine optimale statische Zuordnung errechnet, welche um unrentable Wechsel der Ausführungsumgebung reduziert und mit den errechneten Wahrscheinlichkeiten der auf Laufzeitdaten basierenden Serverzuordnungen zusammengeführt wird (vgl. [BD00]). Der Fokus liegt dabei stets auf der Vermeidung von Kommunikationsaufwand, so dass dieses Ziel einer möglichst flexiblen Zuordnung zur Laufzeit voransteht. Die Prozessmodelle werden daher zur Entwicklungszeit repliziert auf allen relevanten Ausführungseinheiten gespeichert [BRD01] und bei Bedarf die Prozessinstanzdaten zwischen diesen vorausgewählten Ausführungseinheiten migriert. Das Ergebnis stellt somit eine in Hinblick auf diesen Aspekt optimierte variable Zuordnung von Aktivitäten zu Ausführungseinheiten dar, die dennoch zur Entwicklungszeit der Anwendung unter den gegebenen Rahmenbedingungen berechnet wurde. Es steht daher zur Laufzeit nur soviel Flexibilität zur Anpassung zur Verfügung, wie durch die Zuweisungsausdrücke und die daraus entstehende Verteilungskonfiguration vorgegeben wurde.

Ein interessanter Aspekt der weiterführenden Arbeiten der Autoren BAUER, REICHERT und DADAM [BRD01, RB07] ist die gemeinsame Betrachtung von verteilter Prozessausführung und der Gewährleistung einer Anpassbarkeit

des Prozesses auf inhaltlicher Ebene. Für eine Beurteilung der Zulässigkeit von Änderungen wird der aktuelle Ausführungszustand der betreffenden Prozessinstanz benötigt. Da im vorliegenden Ansatz nur die Instanzdaten migriert werden und die Prozessmodelle statisch zur Entwicklungszeit verteilt werden, stellt vor dem Hintergrund der variablen Serverzuordnung zudem die genaue Bestimmung der Ausführungseinheiten, die von inhaltlichen Änderungen der laufenden Prozessinstanz betroffen sind bzw. sein würden, ein Problem dar. Bauer und Dadam stellen hierfür ein Verfahren vor, bei dem der Initiator des Prozesses als Manager der Prozessinstanz die an der Ausführung dieser Prozessinstanz beteiligten Ausführungseinheiten verwaltet. Soll eine inhaltliche Modifikation am Prozess durchgeführt werden, so werden über Sperrmechanismen alle weiteren Migrationen dieser Prozessinstanz untersagt, die Menge der aktiven Ausführungseinheiten bestimmt, anhand des Fortschritts des Prozesses über die Zulässigkeit der beabsichtigten Änderung entschieden und – im positiven Fall – die dynamische Änderung bei allen aktiven Ausführungseinheiten durchgeführt. Nach der Freigabe der Sperren kann die verteilte Prozessausführung fortgesetzt werden. Bei weiteren Migrationen dieser Prozessinstanz müssen die erfolgten Änderungen am Prozessmodell nun noch allen weiteren an der Ausführung beteiligten Ausführungseinheiten mitgeteilt werden. Hierzu wird die Änderungshistorie der Prozessinstanz verwendet, welche bei der Migration der Instanzdaten ohnehin mitgeführt wird [BRD01, RB07].

Insgesamt lässt sich festhalten, dass durch die Minimierung des Kommunikationsaufwands in Form der statischen Vorverteilung des Prozessmodells sowohl in technischer als auch in fachlicher Hinsicht Flexibilisierungspotential ungenutzt bleibt bzw. eine Anpassung erschwert wird. Zudem muss berücksichtigt werden, dass sowohl die Verteilung des Prozessmodells (unmittelbar) vor der Laufzeit des Prozesses als auch die Durchsetzung von inhaltlichen Anpassungen zur Laufzeit Kommunikationsaufwand verursacht, welcher bei der Verteilung individueller Prozessinstanzen in eine Aufwandsbetrachtung mit einbezogen werden muss.

5.5.4 OSIRIS

Aufgrund der Restriktionen von zentralen Infrastrukturen beschreiben die Autoren SCHULER et al. [SWSS03, SWSS04, Sch04b] einen dezentralen Ansatz für die verteilte Ausführung von Prozessen nach dem Peer-to-Peer-Modell. Vor dem Hintergrund dienstorientierter Architekturen stellen sie mit ihrem Projekt *OSIRIS* (*Open Service Infrastructure for Reliable and Integrated Process Support*) einen Ansatz vor, welcher ein verteiltes Prozessmanagementsystem auf der Basis einer *Hyperdatenbank* realisiert. Zur Laufzeit können basierend auf diesem Ansatz Instanzdaten laufender Prozesse zwischen den Prozessteilnehmern (*Peers*) migriert werden, um auf elektronische Dienste von registrierten Anbietern zuzugreifen. Dazu wird auf jedem Peer eine Hyperdatenbankschicht (*HDB-Layer*) als lokale Infrastruktur installiert, welche die Koordi-

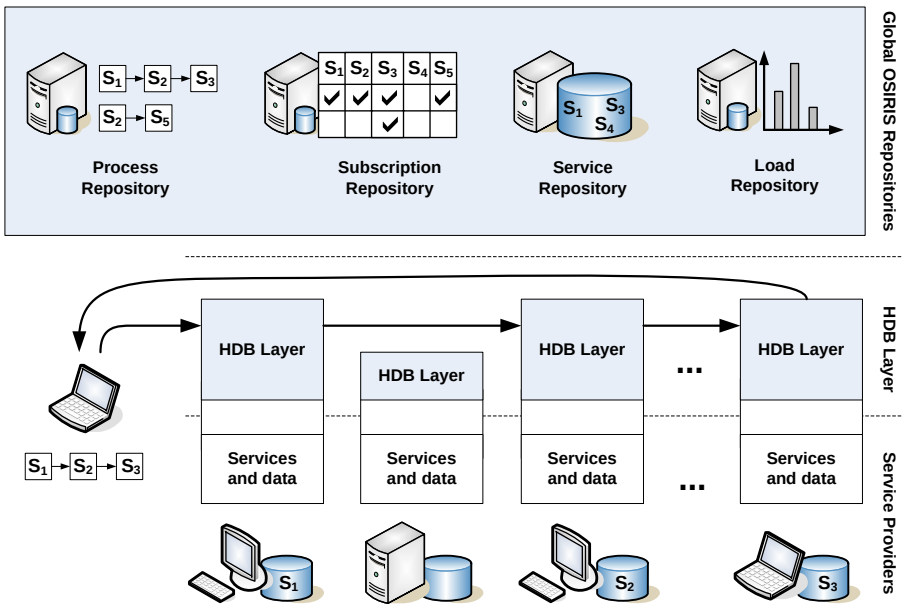


Abbildung 5.15: Architektur von OSIRIS (SWSS03, SWSS04)

nation einer Prozessinstanz für jeweils eine Aktivität übernimmt und nach Abarbeitung der Aktivität den Transfer der Instanzdaten zum Peer eines Anbieters der nachfolgenden Aktivität organisiert [Sch04b]. Um eine dynamische Einbindung von Diensten zu unterstützen, übernimmt diese Schicht auch das Auffinden, das Auswählen und das Aufrufen von Diensten auf der Basis von Metadaten. Relevante Metadaten über im System verfügbare Dienste, die Auslastung einzelner Knoten und die verfügbaren Prozessmodelle werden dazu durch globale Systemdienste verwaltet und mittels Replikation auf die Peers verteilt (*Global OSIRIS Repositories*) . Als Ergebnis bestehen auf den lokalen Knoten des Systems jeweils lokale Versionen dieser Metadaten, welche bei Bedarf aktualisiert werden können. Für den Bezug dieser Informationen existiert ein Publish-Subscribe-Mechanismus, welcher die prozessausführenden Knoten über aktuelle Ereignisse (wie zum Beispiel die Modifikation eines Prozessmodells) informiert [SWSS03]. Die Prozessnavigation ist jedoch von der Replikation der Metadaten entkoppelt, so dass die Ausführung des Prozesses auf den lokal vorhandenen Informationen der einzelnen Peers basiert und nicht durch eine zentrale Instanz koordiniert werden muss [Sch04b]. Abbildung 5.15 zeigt eine zusammenfassende Darstellung dieser Architektur.

Analog zur dienstorientierten Sichtweise kann ein über OSIRIS verteilter Prozess wieder als einzelner Dienst zur Verfügung gestellt und von einem Peer angeboten werden. Für die Modellierung der Prozesse führen die Autoren eine eigene speziell angepasste Sprache ein, welche neben typischen Kontrollflusskonstrukten (vgl. Abschnitt 2.4.1) insbesondere die Definition von transaktionalen Eigenschaften durch Kompensation oder Wiederholung von Aktivitäten besonders berücksichtigt. Zur Durchsetzung der damit verbundenen

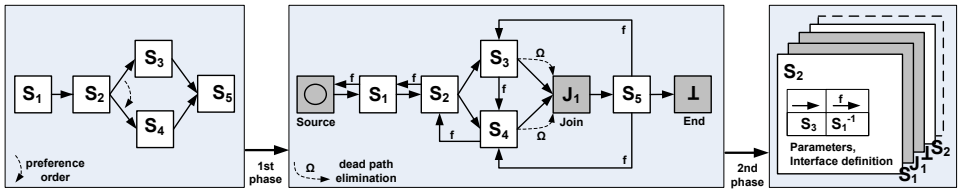


Abbildung 5.16: Transformation eines Prozessmodells in Teilprozesse minimaler Partitionen (SWSS04)

Ausführungsgarantien stellt die Hyperdatenbankschicht Schnittstellen und Dienste zur Verfügung, die an die Funktionalität von klassischen Datenbanken angelehnt sind. Dabei entspricht ein über die Hyperdatenbank ausgeführter Prozess einer Transaktion, und die Kontrolle von Nebenläufigkeit und Atomarität der Prozesse entspricht der Nebenläufigkeitskontrolle und den Wiederherstellungsvorgängen (*Recovery*) in einer Datenbank. Zur Anfrage und Modifikation von Informationen verwendet die Hyperdatenbank Dienstaufrufe. Dienste zur Merkmalsextraktion und die Organisation der Merkmale in Suchdiensten entsprechen der internen Indexkomponente einer Datenbank. Die Optimierung der Ausführung von Prozessen und Suchvorgängen stellt schließlich einen ähnlichen Vorgang wie die klassische Anfrageoptimierung in Datenbanken dar [SSS+04].

Um die Replikation aller vollständigen Prozessmodelle auf allen Peers zu vermeiden, werden die Prozessmodelle bei ihrem Deployment in bedarfsgerechte Teilprozesse (*Execution Units*) zerlegt, welche sowohl um transaktionale Eigenschaften (z. B. Kompensation) als auch um Informationen zur Navigation des Kontrollflusses erweitert werden. Diese Vorgehensweise ist in Abbildung 5.16 an einem Beispiel verdeutlicht. Der hier dargestellte Prozess mit den Dienstaufrufen S_1, S_2, S_3, S_4 und S_5 wird dabei zunächst in einer ersten Phase um statische Start- (*Source*) und Endpunkte (*End*) ergänzt. Falls im Prozessmodell alternative oder parallele Pfade zusammengeführt werden, wird eine spezielle *Join-Aktivität* (J_1) integriert, welche zur Ausführungszeit die Synchronisation der Pfade und ggf. der entstehenden Instanzdaten erlaubt. Im Fall einer alternativen Verzweigung kann eine Präferenz für einen bestimmten Pfad modelliert werden: Im dargestellten Beispiel soll der alternative Pfad über den Dienstaufruf S_4 nur ausgeführt werden, wenn Dienstaufruf S_3 nicht erfolgreich durchgeführt werden kann. Für die entsprechend notwendige *Dead Path Elimination* (vgl. Abschnitt 2.7.2) werden weitere Kanten eingefügt. In der zweiten Phase wird das Prozessmodell partitioniert. Aus dem transformierten Prozessmodell entsteht dabei eine Navigationstabelle, welche sowohl den regulären Kontrollfluss des Prozesses als auch die notwendigen Fehlerbehandlungsmaßnahmen für die resultierende Prozesspartition enthält. Das Beispiel zeigt die Partition S_2 mit einem Verweis auf die reguläre Ausführung von S_3 sowie die Kompensation von S_1 (S_1^{-1}), welche im Fehlerfall durchgeführt werden soll [SWSS04].

Dienstanbieter können sich über einen zentralen Verzeichnisdienst über ihre angebotenen Dienste für die Ausführung bestimmter Prozesspartitionen registrieren. Die entsprechenden Prozesspartitionen werden dann als lokale Version repliziert und können in Folge dezentral aufgerufen und ausgeführt werden. Da vor jedem Dienstaufruf nicht-funktionale Aspekte überprüft werden können (zum Beispiel zur Lastverteilung), ist ein Wechsel der Ausführungseinheit und der von ihr verwalteten Ressourcen auch zur Laufzeit möglich. Der Ansatz zeichnet sich daher durch eine sehr hohe Flexibilität bei der verteilten Ausführung aus.

Die Architektur von OSIRIS hebt die bestehende Unterscheidung zwischen funktionalen Diensten als Ressource und als eigenständige Ausführungseinheit der Prozessausführung nahezu auf. Als Konsequenz ist jeder Dienstanbieter selbst für die Ausführung des jeweiligen Teilprozesses verantwortlich, welcher Zugriff auf seine lokalen Dienste erfordert. Zur Ausführung der Prozesse wird eine hybride Strategie aus Partitionierung des Prozessmodells und der Migration von Instanzdaten verwendet, welche eine Aufteilung des Prozesses zur Zeit des Deployments erfordert. Hierzu wird der Prozess transformiert und nach einzelnen Dienstaufrufen zerlegt, was zu einer maximal verteilten Prozessausführung führt (vgl. Abschnitt 5.6). Änderungen am Prozessmodell oder die abweichende Ausführung einer einzelnen Prozessinstanz erfordern somit den Austausch aller relevanten Prozesspartitionen bei allen registrierten Peers. Es ist zudem die Nutzung einer proprietären Sprache und eines entsprechend spezialisierten Prozessmanagementsystems erforderlich, was einen hohen Einstiegsaufwand zur Nutzung der verteilten Prozessausführung bedeutet.

5.5.5 Ansatz von KUNZE et al.

Zur Realisierung der kontextbasierten Kooperation zur Unterstützung verteilter Prozesse in mobilen Umgebungen (vgl. Abschnitt 4.2.4) schlagen KUNZE et al. [Kun05, KZL06, KZL07b, Kun08] eine spezielle Prozessbeschreibungssprache auf der Basis von XPDL (vgl. Abschnitt 5.2) vor, welche während der Ausführung aktuelle Zustandsinformationen aufnehmen und zu jedem erlaubten Zeitpunkt an eine andere Ausführungseinheit übergeben werden kann. Hierbei wird bei der Instantiierung des Prozesses das Prozessmodell kopiert und als explizite Darstellung der Prozessinstanz mit ihren jeweiligen Laufzeitdaten repräsentiert. Jede in der Prozessbeschreibung definierte Aktivität enthält dazu ein Zustandsprädikat, welches basierend auf dem Lebenszyklusmodell von LEYMANN und ROLLER (vgl. Abschnitt 2.7.2) den aktuellen Zustand der einzelnen Aktivität während der Prozessbearbeitung angibt. Die Gesamtheit der Zustände aller Aktivitäten definiert dabei den Zustand der betreffenden Prozessinstanz, so dass nach einer Migration des Prozesses dessen Ausführung direkt an der Position im Kontrollfluss anknüpfen kann, an welcher die Ausführung der vorhergehenden Ausführungseinheit beendet wurde. Für die Auswahl von Ausführungseinheiten können dabei zusätzlich Strategi-

en innerhalb der Prozessbeschreibungssprache angegeben werden, welche die Migration zu Ausführungseinheiten mit bestimmten benutzerdefinierten Eigenschaften (z. B. mit einer bestimmten geographischen Position) veranlasst [KZL06, KZL07b].

Bei dem von KUNZE et al. vorgestellten Ansatz kann dabei sowohl die Auswahl der Ressourcen zur Ausführung einer einzelnen Aktivität (d. h. elektronische Dienste oder menschliche Prozessteilnehmer) als auch die Auswahl des für die Ausführung des Prozesses verantwortlichen Geräts mit seiner Prozess-Engine dynamisch zur Laufzeit des Prozesses vorgenommen werden. Die Angabe der für die Ausführung einer Aktivität benötigten Funktionalität geschieht dabei über *abstrakte Dienstklassen*, welche Syntax und Semantik von funktional gleichwertigen Diensten durch einen eindeutigen Identifikator beschreiben [KZL07a]. Dabei können – je nach den Fähigkeiten des prozessausführenden Geräts – auch Dienste unterschiedlicher Technologien (z. B. Java RMI, Corba oder Web Services) zugegriffen werden, sofern sie der gesuchten Dienstklasse entsprechen. Die Migration des Prozesses auf andere Geräte erhöht somit die Wahrscheinlichkeit, dass ein geeigneter Dienst durch einen lokalen oder entfernten Dienstaufruf zugegriffen werden kann [ZKL09a].

Für die dezentrale Ausführung von migrierenden Prozessen nach dem beschriebenen Modell ist eine spezielle Prozess-Engine nötig, welche die Zustandsinformationen aus der Prozessbeschreibung extrahieren, die Ausführung der verbleibenden Aktivitäten durchführen und den aktuellen Prozesszustand vor einer Migration wieder in die Prozessbeschreibung einfügen kann. Um in einer dynamischen mobilen Umgebung andere Geräte mit ihren Diensten und aktuellen Eigenschaften (wie zum Beispiel Ort, Auslastung oder Kosten) auffinden zu können, integrieren KUNZE et al. ein generisches Kontextdatenmodell und -managementsystem, welches die für die jeweilige Prozessausführung relevanten Kontextinformationen überwacht und dem Prozessmanagementsystem zur Verfügung stellt [KZTL08]. Das Kontextmanagementsystem enthält als Teil eines verteilten Verzeichnisdienstes zudem ein lokales Verzeichnis, welches die jeweils vom Gerät angebotenen Dienste auflistet und bei einer Anfrage nach einer bestimmten Dienstklasse eine technische Dienstbeschreibung zum Zugriff des Dienstes bereitstellen kann. Der gesamte Ansatz ist somit auf die verteilte Ausführung von Prozessen in lokalen (Ad-hoc-)Netzwerken ohne zentrale Kontrollinstanz ausgerichtet und weist eine sehr hohe Flexibilität für die Verteilung von Prozessinstanzen auf.

Durch die spezielle Ausrichtung des Ansatzes ist die Anwendbarkeit für die spontane Verteilung zentral ausgeführter Prozessmodelle vor allem durch die proprietäre Prozessbeschreibungssprache begrenzt, da die verteilt auszuführenden Prozesse zunächst in das vorgeschlagene Format transferiert werden müssen. Dieses beruht zwar auf dem Standard XPDL, hebt aber die Trennung zwischen fachlichen und technischen Anweisungen sowie zwischen Prozessmodell und Prozessinstanzen weitestgehend auf. So enthält bereits jedes Prozessmodell Konstrukte zur Beschreibung des Zustands etwaiger Instanzen und zur Definition von Verteilungsinformationen. Aus diesem Grund liegen

auch die Rollen des Prozessmodellierers und Prozessinitiators bei diesem Ansatz relativ eng zusammen, so dass hier keine unterschiedlichen Vorgaben für beide Gruppen unterschieden werden.

5.6 Vollverteilte Prozessausführung

Vollverteilte Prozessmanagementsysteme stellen eine Variante der verteilten Prozessausführung dar, bei welcher es zu einer Verteilung des Prozesses mit Bildung minimaler Partitionen kommt [BD99]. Hierbei gibt es in der Regel keine dedizierten Ausführungseinheiten, welche den Kontrollfluss des Prozesses verwalten, da jede Ressource selbst für die Ausführung ihrer Aktivität und die Weiterleitung der Prozessinstanz an eine geeignete Ressource zur Bearbeitung der Nachfolgeaktivität verantwortlich ist. Dazu muss theoretisch jede Ressource ihr eigenes (vollwertiges oder leichtgewichtiges) Prozessmanagementsystem besitzen, was aus Gründen der zumeist hohen Anschaffungs- und Wartungskosten solcher Systeme oft keine realistische Annahme darstellt. Folglich besteht die zuvor angenommene Zuordnung von (mehreren) Ressourcen zu einer übergeordneten Ausführungseinheit nicht und der Prozess muss nach Beendigung *jeder* Aktivität zur nächsten Ressource migriert werden. Obwohl diese Art der Verteilung eine dezentrale Prozessausführung erlaubt und eine sehr gute Skalierbarkeit und Lastverteilung aufweist, wird durch die inhärente Festlegung der Zerteilung des Prozesses in einzelne Aktivitäten die Bildung von individuell sinnvollen Prozesspartitionen für eine verteilte Ausführung des Prozesses stark begrenzt. Es kann hierbei zudem kaum eine Optimierung hinsichtlich der Reduzierung des Kommunikationsaufwands vorgenommen werden, da nach jeder Aktivität nach einer geeigneten Ressource zur Bearbeitung der Nachfolgeaktivität gesucht werden muss.

Bei der vollverteilten Prozessausführung kann unterschieden werden, ob die Verteilung zur Entwicklungs- oder zur Laufzeit des Prozesses geschieht. Der Vollständigkeit halber wird mit *Exotica/FMQM* (vgl. Abschnitt 5.6.1) ein Ansatz zur statischen Verteilung beschrieben, während die anderen Ansätze (vgl. Abschnitte 5.6.2 bis 5.6.4 Varianten für die Realisierung einer dynamischen Verteilung darstellen. Letztere können durch die Bildung minimaler Partitionen zur Laufzeit als ein Spezialfall der dynamischen Prozessmigration angesehen werden.

5.6.1 Exotica/FMQM

Parallel zu dem bereits beschriebenen Ansatz *Exotica/FMDC* stellen die Autoren ALONSO et al. [AAA+95] mit *Exotica/FMQM* (*FlowMark on Message Queue Manager*) einen vollverteilten Ansatz für die Ausführung prozessorientierter Anwendungen vor. Hierbei wird die prozessorientierte Anwendung vor ihrer Ausführung partitioniert und die einzelnen Aktivitäten den potentiell für die Ausführung geeigneten Ressourcen zugeordnet. Nach der Beendigung jeder Aktivität wird die Ressource für die Ausführung der nachfolgen-

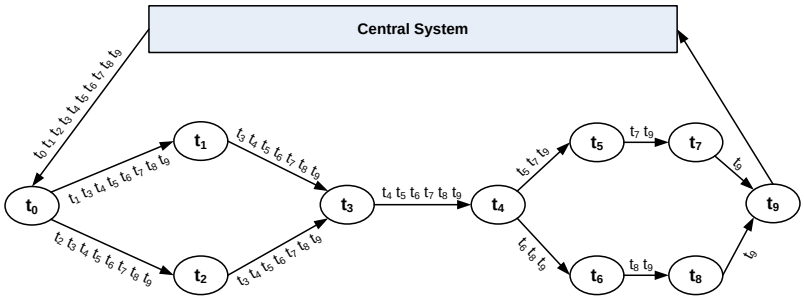


Abbildung 5.17: Dezentralisiertes Prozessmanagement durch selbstbeschreibende Prozesse (ACMM07)

den Aktivität bestimmt. Dazu werden Nachrichten an alle Ressourcen versendet, welche potentiell für die Ausführung verantwortlich sein können. Die Nachricht wird dabei jeweils in eine persistente Warteschlange der betreffenden Ressource eingefügt, aus welcher die Ressource die Nachricht entnehmen und über die Ausführung der Aktivität entscheiden kann. Eine Synchronisation erfolgt dabei über das Transaktionskonzept des Warteschlangensystems [AAA+95, BD99]. Da diese Art der Zuordnung sehr aufwendig und mit einem hohen Kommunikations- und Koordinationsaufwand verbunden ist, ist eine fixe Zuordnung von Aktivitäten zu Ressourcen die einzige Möglichkeit, das Verfahren effizient zu gestalten, was zusammen mit der statischen Vorverteilung in einer sehr geringen Flexibilität des Ansatzes resultiert [BD99, Sch01].

5.6.2 Ansatz von ATLURI et al.

Zur Unterstützung einer vollständig dezentralen Prozessausführung mit dynamischer Partitionierung schlagen die Autoren ATLURI et al. [ACMM07] ein Peer-to-Peer-basiertes Verteilungsmodell vor. Hierbei verfügt jede Ressource, welche potentiell an der Ausführung des Prozesses mitwirkt, über eine spezielle Adapterkomponente (*Workflow Stub*) zur Verarbeitung, Modifikation und Weiterleitung der Prozessbeschreibung. Während der Bearbeitung durch eine solche Adapterkomponente wird diese Prozessbeschreibung entsprechend angepasst, so dass der resultierende Prozess genau den noch auszuführenden Teil des Gesamtprozesses durch den als nächstes auszuwählenden Prozessteilnehmer darstellt. Es kommt dabei zu einer dynamischen Partitionierung der Prozessbeschreibung mit abnehmender Größe der Fragmente, welche aus diesem Grund als *selbstbeschreibende Prozesse* bezeichnet werden. Die durch die Prozessausführung entstehenden Ergebnisse werden schließlich durch den letzten Prozessteilnehmer an das aufrufende System zurückgemeldet, welches nach ATLURI et al. eine zentrale Komponente darstellt. Abbildung 5.17 verdeutlicht diese Vorgehensweise an einem Beispiel zur Ausführung eines Prozesses mit den Aktivitäten t_0 bis t_9.

Der vorgestellte Ansatz der dynamischen Partitionierung ist gut für dezentrale Infrastrukturen geeignet und besitzt eine hohe Flexibilität. Die genaue

Abbildung des formalen Modells auf die zur Umsetzung angestrebten Prozessbeschreibungssprachen (z. B. WS-BPEL) wird jedoch nicht thematisiert. So wird zum Beispiel der Umgang mit erweiterten Kontrollflussstrukturen (wie z. B. Ereignissen) nicht betrachtet. Zudem ist das dynamische Partitionieren durch den Verlust von Informationen gekennzeichnet, welche zwar für eine fortschreitende Ausführung des Prozesses nicht mehr notwendig sind, jedoch auf diese Weise auch nicht mehr für eine etwaige Fehlerbehandlung zur Verfügung stehen.

5.6.3 Ansatz von MONTAGUT und MOLVA

Die Autoren MONTAGUT und MOLVA [MM05c] adressieren in ihrer Arbeit die verteilte Ausführung von WS-BPEL-Prozessen auf mobilen Geräten. Insbesondere steht bei ihren Ansatz im Vordergrund, dass in mobilen Umgebungen kein Gerät als zentraler Koordinator der Ausführung dienen kann und dass eine dauerhafte Zuordnung von Rollen zu mobilen Geräten aufgrund der dynamischen Eigenschaften mobiler Umgebungen nicht vorteilhaft ist.

Um eine dynamische Zuordnung von Aktivitäten zu Geräten zu erlauben, besitzt jedes Gerät neben den von ihm individuell angebotenen (mobilitätsbezogenen) Diensten eine eigene Prozess-Engine. Dienste des Geräts können dabei nur über die lokale Prozess-Engine von außen zugegriffen werden. Um anderen (mobilen) Geräten auch komplexe Dienste anbieten zu können, können diese durch die Spezifikation von nur lokal sichtbaren WS-BPEL-Prozessen komponiert werden (*Private Processes*). Die durch die internen Prozesse erbrachten Funktionalitäten können über eine Ontologie bei einem Dienst zur Rollenauflösung (*Role Discovery Service*) registriert werden. Zur Laufzeit kann dann eine vollverteilte Ausführung von öffentlichen WS-BPEL-Prozessen stattfinden (*Public Processes*), welche Schritt für Schritt durch die Prozess-Engines der einzelnen Geräte verwaltet wird. Dabei wird nach jeder ausgeführten Aktivität erneut der Dienst zur Rollenauflösung angefragt, um anhand einer semantischen Dienstbeschreibung ein geeignetes Gerät für die jeweils nachfolgende Aktivität aufzufinden. Neben der Zuordnung auf Basis der gesuchten Funktionalität können bei dem vorgestellten Ansatz auch Sicherheitsaspekte, wie zum Beispiel die Berechtigung zur Ausführung einer bestimmten Aktivität des öffentlichen Prozesses berücksichtigt werden [MM05c].

Durch die Kombination von internen und öffentlichen Prozessen kann dieser Ansatz als hybride Lösungsstrategie aus vollverteilter Prozessausführung eines öffentlichen Prozesses und der Ausführung von partitionierten privaten Prozessschritten angesehen werden. Durch diese Vorgehensweise werden die Vorteile einer lokal autonomen Anpassung von gekapselten Prozessteilen im privaten Prozess und die dynamische Zuweisung von Aktivitäten des öffentlichen Prozesses zu Ressourcen kombiniert. Hinsichtlich der Flexibilität der Verteilung ergibt sich jedoch nur eine eingeschränkte Anpassungsfähigkeit, da die Verteilung des öffentlichen Prozesses von zur Entwick-

lungszeit festgelegten Schnittstellen der lokalen Prozesse abhängt. Weitergehende Überlegungen zu Synchronisation und Fehlerbehandlung sind wie bei dem in Abschnitt 5.6.2 beschriebenen Verfahren nicht Teil der vorgestellten Arbeit [MM05c].

5.6.4 Ansatz von YU und YANG

Die Autoren YU und YANG [YY07] kritisieren in ihrer Arbeit die statische Zuordnung von Prozesspartitionen zu Ausführungseinheiten, da hierdurch eine Neuzuordnung während der Ausführung des Prozesses erschwert wird und vorab auch Ressourcen belastet werden, welche bei der tatsächlichen Ausführung des Prozesses aufgrund des individuellen Ausführungsverlaufs alternativer Pfade ggf. gar nicht zugegriffen werden. Sie stellen daher ähnlich wie ATLURI et al. (vgl. Abschnitt 5.6.2) einen formalen Ansatz vor, welcher die vollständig dezentrale Ausführung von prozessorientierten Anwendungen über das *Weiterreichen der verbleibenden Prozessausführung* (*Continuation Passing*) thematisiert. Die für die Ausführung der als nächstes anstehenden Aktivität verantwortliche Ausführungsumgebung wird hierbei von dem aktuell für die Prozessausführung verantwortlichen System ausgewählt und bei Bedarf eine lokal aufbereitete Version der noch auszuführenden Prozessbeschreibung zusammen mit den notwendigen Instanzdaten des Prozesses an das ausgewählte System migriert. Die verbleibende Prozessausführung wird jeweils durch zwei mögliche Fortführungen des Prozesses in Form einer *Fortführung im Erfolgsfall* und einer *Fortführung im Fehlerfall* zur Laufzeit angepasst, wobei letztere die durchzuführenden Schritte für eine mögliche Kompensation der Prozessausführung enthält [YY07].

Der Ansatz von YU und YANG besitzt eine sehr hohe Flexibilität und ist daher gut für dynamische Umgebungen geeignet, in denen Prozesse auf einem bestimmten Knoten des Systems gestartet und – aufgrund von spontanen Ereignissen zur Laufzeit des Prozesses – von anderen Systemen fortgeführt werden müssen. Auch bei diesem Ansatz wird die originäre Prozessbeschreibung schrittweise modifiziert, was hier jedoch zur Laufzeit durch die einzelnen Prozessteilnehmer geschieht. Die Anpassung der Verteilung geschieht somit auf der Ebene der Prozessinstanzen, so dass diese einzeln während ihrer Ausführung anpassungsfähig sind. Die Autoren geben jedoch zu bedenken, dass die Migration von Prozessinstanzen durch das dezentrale Weiterreichen der verbleibenden Prozessausführung schwer zu überwachen und zu verwalten ist sowie hohe Anforderungen an Sicherheitsmechanismen stellt, um die transferierten Informationen vor unbefugtem Zugriff zu schützen [YY07].

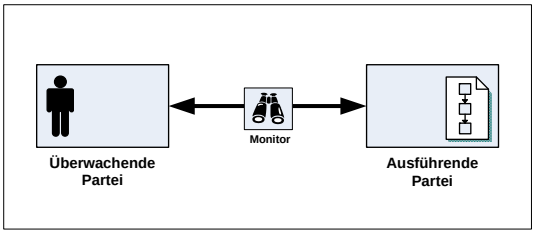
5.7 Überwachung und Steuerung verteilt ausgeführter Prozesse

Wie in Kapitel 3 erarbeitet wurde, geht einer adäquaten Anpassung von prozessorientierten Anwendungen stets die Erhebung von geeigne-

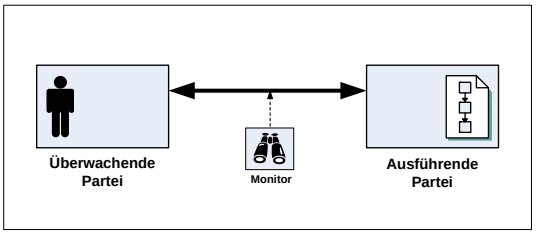
ten Informationen über die Umgebung, die (potentiellen) Ressourcen und Ausführungseinheiten des Prozesses sowie über die Prozessausführung selbst voraus. Dabei steht nach einer Betrachtung kontextbasierter Ansätze (vgl. Abschnitt 3.5) und ereignisgesteuerter Systeme (vgl. Abschnitt 3.7) zur Flexibilisierung der Prozessausführung im Vordergrund, dass die für die Anpassung eines Prozesses individuell relevanten Informationen möglichst unmittelbar nach dem Auftreten beobachteter Geschehnisse verarbeitet werden sollten, um eine zeitnahe Anpassung der Prozessausführung durchführen zu können. Für die Flexibilität eines verteilt ausgeführten Prozesses ist es somit entscheidend, dass die für den Beobachter relevanten Details der Prozessausführung individuell überwacht werden können und dass, sofern eine Anpassung erforderlich ist, geeignete Maßnahmen für eine individuelle Steuerung verteilt ausgeführter Prozesse zur Verfügung stehen. Zudem sollten die Maßnahmen zur Überwachung und Steuerung von Prozessen möglichst auch selbst zur Laufzeit anpassbar sein, da sich die Menge der individuell relevanten Informationen aufgrund dynamischer Umgebungsänderungen jederzeit verändern kann.

Im Gegensatz zu zentral ausgeführten (nicht-verteilten) Prozessen unterliegt die Überwachung und Steuerung von verteilt ausgeführten Prozessen besonderen Herausforderungen, welche insbesondere bei organisationsübergreifenden Prozessen zum Tragen kommen. Dabei ist die Sichtbarkeit auf entfernt ausgeführte Prozesspartitionen einerseits aufgrund der Autonomie der beteiligten Organisationen [WKK+10] und andererseits aufgrund der Heterogenität der verwendeten Prozessmanagementsysteme oft erheblich eingeschränkt. Somit ist es für einen externen Beobachter kaum möglich, eigenständig Informationen am Ort der Ausführung zu erheben oder sogar auf eine entfernt ablaufende Prozessausführung Einfluss zu nehmen. Eine Analyse der gesamten Ausführung einer Prozessinstanz ist daher oft erst nach deren Beendigung möglich, wobei auch hier die Rekonstruktion verschiedener Details bei der Beteiligung mehrerer Ausführungseinheiten schwierig ist. So kann zum Beispiel oft nur die Dauer der gesamten Prozessausführung abgeleitet werden, nicht aber die Dauer der Ausführung einzelner Prozesspartitionen. In vielen Fällen muss zudem auf die Korrektheit der freiwillig bereitgestellten Daten der beteiligten Parteien vertraut werden (vgl. [KWA99a, CCSD04, JBF07]).

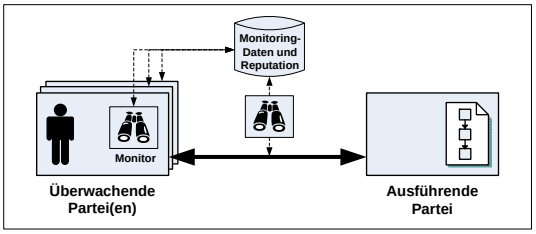
Abbildung 5.18 zeigt mögliche Varianten zur Überwachung von entfernt ausgeführten Prozesspartitionen, welche den Stand der Forschung in diesem Bereich widerspiegeln. Hierbei kann danach unterschieden werden, ob lediglich das *extern sichtbare Verhalten* des Kooperationspartners beobachtet wird (z. B. durch das Nachvollziehen des Nachrichtenaustauschs) (vgl. Abbildungen 5.18a bis 5.18c) oder ob auch interne Details der entfernten Prozessausführung erlangt und verarbeitet werden können (vgl. Abbildungen 5.18d und 5.18e). Im Folgenden werden basierend auf dieser Einteilung verschiedene Ansätze zur Überwachung und Steuerung von verteilten und insbesondere von verteilt ausgeführten Prozessen betrachtet und in Hinblick auf ihre Eignung für dynamisch verteilte Prozesse untersucht. Dabei wird auch berücksichtigt, wie viel



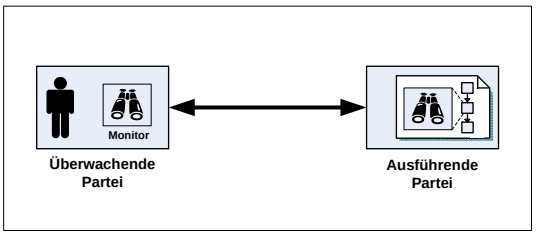
(a) Zentrales Monitoring des extern sichtbaren Verhaltens durch dauerhafte Proxy-Komponenten



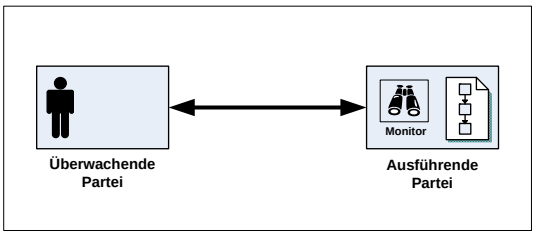
(b) Zentrales Monitoring des extern sichtbaren Verhaltens durch die bedarfsmäßige Entnahme von Stichproben



(c) Verteiltes Monitoring auf Basis von Reputationssystemen und gemeinsamer Datenhaltung



(d) Dezentrales Monitoring durch Erweiterung der Prozessbeschreibung um Monitoring-Aktivitäten



(e) Dezentrales Monitoring durch Bereitstellung von Informationen durch den Kooperationspartner

Abbildung 5.18: Varianten der Überwachung von entfernt ausgeführten Prozesspartitionen (tlw. nach(JBF07))

Flexibilität die Verfahren bei der Auswahl und der dynamischen Anpassung von zu beobachtenden bzw. zu steuernden Parametern bereitstellen.

5.7.1 Überwachung des Nachrichtenaustausches

Eine erste Variante von Ansätzen zur Überwachung von verteilten, insbesondere dienstorientierten Softwarearchitekturen stützt sich auf die Beobachtung von Kooperationspartnern durch das Nachvollziehen des Nachrichtenaustausches, welcher aus der Bearbeitung einer an den Kooperationspartner ausgelagerten Aufgabe resultiert. Der Nachrichtenaustausch steht dabei stellvertretend für den Aufruf von Ressourcen, welche zur Erfüllung einer (komplexen) Teilaufgabe involviert werden. Die Autoren SAHAI et al. [SMOW02] stellen hierzu einen Ansatz vor, welcher es erlaubt, SOAP-Nachrichten mit Management-Informationen (wie Zeitstempeln oder Dienstidentifikatoren) anzureichern, um darauf basierend Aussagen über die Effizienz oder die Topologie der Aufgabenbearbeitung zu erhalten. Die gesammelten Informationen werden hierbei jedoch in der Regel mit dem Nachrichtenfluss transportiert und erreichen den Beobachter der (Prozess-)Ausführung daher erst mit der Antwortnachricht des Kooperationspartners, d. h. bei Abschluss der entfernt ausgeführten Aufgabe.

Der allgemeine Ansatz der direkten Beobachtung des Nachrichtenaustausches zur Laufzeit eines Dienstes bzw. eines Prozesses durch eine feste Proxy-Komponente (vgl. Abbildung 5.18a) oder durch das (gelegentliche) Abhören der Nachrichten (vgl. Abbildung 5.18b) hat in der Regel nur einen geringen Mehrwert, da hierbei nur das ohnehin nach außen sichtbare Verhalten überprüft werden kann. In vielen Fällen ist es zudem schwierig, die Semantik der ausgetauschten Nachrichten zu erkennen oder die Nachrichten der richtigen Prozessinstanz zuzuordnen [KSK07]. Bei einer Verschlüsselung der Nachrichtenübertragung ist eine Überwachung nicht möglich. Zudem ist diese Variante des Monitorings entweder sehr aufwendig (z. B. im Fall einer festen Proxy-Komponente) oder liefert nur sehr ungenaue Resultate (z. B. bei der bedarfsmäßigen Entnahme von Stichproben) [JBF07].

5.7.2 Sammlung und Austausch von Erfahrungswerten

Um die Autonomie von Kooperationspartnern nicht einzuschränken, schlagen die Autoren KLINGEMANN, WÄSCH und ABERER [KWA99a] vor, nur das extern beobachtbare Verhalten von Kooperationspartnern zu analysieren. Dazu wird empfohlen, Informationen über die frühere Nutzung von Diensten zu sammeln, daraus ein abstraktes Modell dieses Dienstes zu bilden und mittels Markov-Modellen für kontinuierliche Zeit eine Vorhersage zukünftiger Verhaltensweisen zu bestimmen [KWA99a]. Bei dynamisch verteilt ausgeführten Prozessen kann ein solcher Ansatz jedoch nicht angewendet werden, da hier in der Regel keine umfangreichen eigenen Erfahrungen mit einem bestimmten Dienstanbieter vorausgesetzt werden können. Zudem stützt sich der Ansatz auf Dienste

mit relativ stabilem Verhalten [KWA99a], wovon in dynamischen Umgebungen ebenfalls nicht ausgegangen werden kann.

Um bestehende Erfahrungswerte zu aggregieren und gemeinsam nutzbar zu machen, schlagen die Autoren JURCA, BINDER und FALTINGS [JF05, JBF07, JFB07] ein reputationsbasiertes Verfahren vor, welches den Austausch von gesammelten Daten zu Erfahrungswerten hinsichtlich von QoS-Kriterien in dienstorientierten Architekturen erlaubt. Hierbei werden bei jedem Dienstaufruf durch den Dienstnutzer relevante Daten wie Ausführungsdauer, Verfügbarkeit oder Zuverlässigkeit erhoben und periodisch an ein (zentral oder dezentral gehaltenes) Reputationssystem gemeldet, welches die Daten für jeden einzelnen Dienst bzw. Dienstanbieter aggregiert [JBF07, JFB07]. Das beschriebene Vorgehen entspricht der in Abbildung 5.18c dargestellten Variante von Überwachungsmaßnahmen in verteilten Systemen.

Eine reputationsbasierte Überwachung hat den Vorteil, dass (bei einer ausreichenden Menge an Teilnehmern) ein repräsentatives Bild über die Qualität eines Dienstes erreicht und kostengünstig für alle (potentiellen) Dienstnutzer verfügbar gemacht werden kann. Zudem wird hierbei vermieden, dass der Dienstanbieter falsche Informationen verbreiten kann, um die Qualität seiner Dienstleistung positiv darzustellen – wie dies etwa bei der Variante des dezentralen Monitorings durch Bereitstellung von Informationen durch den Kooperationspartner (vgl. Abbildung 5.18e) möglich wäre. Eine Verfälschung der Daten ist hier aber dennoch durch die Weitergabe von falschen Informationen von Seiten der Dienstnutzer möglich, was wiederum erweiterte Konzepte zu deren Absicherung erforderlich macht [JBF07, JFB07]. In Hinblick auf die Überwachung von verteilt ausgeführten Prozessen zum Zweck der Flexibilisierung der Prozessausführung ist dieser Ansatz jedoch insbesondere aufgrund der fehlenden Aktualität der Monitoring-Daten nicht geeignet. Hierfür relevante Daten wie der Aufenthaltsort oder die Auslastung eines Systems ändern sich in dynamischen Umgebungen zu schnell, als dass sie von einem Reputationssystem (welches in erster Linie Durchschnittswerte auf Basis vergangener Erfahrungen liefert) angemessen erfasst und kommuniziert werden können. Zudem liegt auch diesem Ansatz nur das extern beobachtete Verhalten des Kooperationspartner zugrunde.

5.7.3 Ansatz von BARESI, GHEZZI und GUINEA

Die Autoren BARESI, GHEZZI und GUINEA [BGG04, BG05] stellen in ihren Arbeiten einen Ansatz für die Überwachung von WS-BPEL-Prozessen vor, der auf der Integration zusätzlicher Aktivitäten beruht (vgl. Abbildung 5.18d), welche während der Ausführung der Prozessinstanzen aktuelle Informationen an ein zentrales Monitoring-System zurückmelden können. Die Anreicherung eines fachlichen Prozessmodells mit Monitoring-Aktivitäten kann dabei auf Basis von bestehenden Dienstgütevereinbarungen durch eine automatische Transformation des WS-BPEL-Modells geschehen. Abbildung 5.19 visualisiert die entsprechende Vorgehensweise. Hierbei werden zur Entwicklungszeit die zu

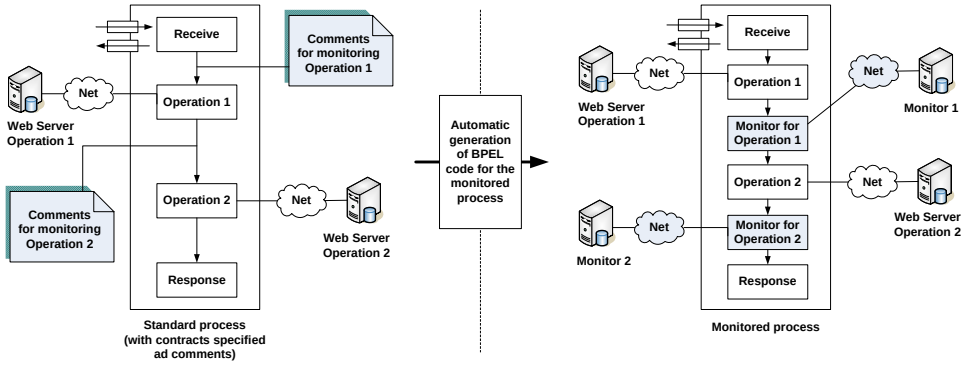


Abbildung 5.19: Transformation eines Prozessmodells zur Anreicherung des Kontrollflusses mit Monitoring-Aktivitäten (BGG04)

überwachenden Prozessmodelle mit Annotationen versehen, welche in Form von Kommentaren in die Prozessbeschreibung eingefügt werden. Da WS-BPEL über die Möglichkeit zur Kommentierung von Kontrollflusskonstrukten verfügt, wird hierdurch die Konformität des Prozessmodells zum Standard nicht verletzt. Die Annotationen werden im weiteren Verlauf durch einen Präprozessor in verschiedene zusätzliche WS-BPEL-Konstrukte transformiert. Dabei wird die Überwachung von nicht-funktionalen Eigenschaften wie Timeouts und technischen Fehlern in WS-BPEL als Ereignis- bzw. Fehlerbehandlungsroutine abgebildet und die Überwachung von funktionalen Anforderungen als zusätzliche Monitoring-Aktivitäten realisiert, welche zum Beispiel den Inhalt von Variablen an einen Monitoring-Dienst weiterleiten können. Durch eine lose Kopplung von Monitoring-Aktivität und Monitoring-Dienst kann die Implementierung des Monitoring-Dienstes (ggf. sogar zur Laufzeit) geändert werden, ohne dass Änderungen an der Prozessbeschreibung erforderlich werden. BARESI, GHEZZI und GUINEA bezeichnen ihren Ansatz daher als *nicht-invasiv* [BGG04]. Wie in Abbildung 5.19 zu sehen ist, enthält das letztendlich von einer Prozess-Engine ausgeführte Prozessmodell jedoch rein technische Aktivitäten, deren Ein- und Ausgabedaten sowie deren Positionierung statisch vor der Ausführung des Prozessmodells festgelegt werden müssen. Nach dem Verständnis dieser Arbeit und aus einer technischen Sichtweise stellt dieser Ansatz daher durchaus ein invasives Verfahren dar. Derselbe Kritikpunkt gilt auch für ein ähnliches Verfahren von ROTH, SCHIEFER und SCHATTEN [RSS06b], welche durch die Integration von Sondierungspunkten echtzeitnahe Fortschrittsinformationen an ein zentrales Monitoring-System liefern.

In einer weiterführenden Arbeit [BG05] erweitern BARESI und GUINEA ihr ursprüngliches Konzept um sogenannte *Monitoring-Regeln*, welche analog zum oben genannten Ansatz zum Zeitpunkt des Deployments in den WS-BPEL-Prozess eingearbeitet werden können. Über die Spezifikation von *Prioritäten*, *Zeitfenstern* oder *Dienstanbietern* erlauben es diese Regeln, den Grad der Überwachung zur Laufzeit dynamisch an den jeweiligen Kontext anzupassen. Dazu wird zur Auswertung jeder Regel ein *Monitoring-Manager* aufgeru-

fen, welcher darüber entscheidet, ob Überwachungsmaßnahmen durchgeführt werden sollen. Im *Monitoring-Manager* kann dazu zur Laufzeit festgelegt werden, welche Auswirkungen das Auslösen einer Regel haben soll. Zum Beispiel kann auf diese Weise definiert werden, dass Dienste eines bestimmten Dienstanbieters zu einer bestimmten Zeit nicht überwacht werden müssen.

Der Ansatz bietet insgesamt nur eine geringe Flexibilität bei der initialen Art und Auswahl der zu überwachenden Daten. So kann durch die Platzierung zweier Monitoring-Aktivitäten vor bzw. nach einer funktionalen Aktivität zum Beispiel die Dauer der Ausführung dieser Aktivität überwacht oder der aktuelle Zustand der Prozessausführung in Erfahrung gebracht werden (siehe auch [RSS06b]). Weitere insbesondere für eine verteilte Ausführung interessante Parameter, wie der aktuelle Ort der Ausführung oder die aktuelle Auslastung der Prozess-Engine, bleiben jedoch bei diesem Ansatz dem Prozessmanagementsystem der ausführenden Organisation vorbehalten.

Eine Steuerung der Prozessausführung durch die vorausschauende Integration von Aktivitäten, welche zum Beispiel die Werte einzelner Prozessvariablen verändern, ist bei diesem Ansatz – obwohl nicht explizit beschrieben – prinzipiell möglich. Ebenso wie bei der Überwachung ist der Umfang an Steuerungsmaßnahmen hierbei jedoch relativ begrenzt und bezieht sich nur auf veränderliche Werte der Prozessinstanz. Ein Einfluss auf die Verteilung des Prozesses ist somit kaum möglich und weitere Steuerungsmöglichkeiten, wie zum Beispiel das Abbrechen der Prozessausführung, sind nur mit großem Modellierungsaufwand auf diese Weise zu realisieren. Da die Integration solcher technischer Aktivitäten zur Entwicklungszeit des Prozesses oder zumindest *vor* der Verteilung der Prozesspartitionen erfolgen muss, ist zudem zur Laufzeit keine weitere Anpassung der Überwachungs- oder Steuerungsmaßnahmen mehr möglich.

Ein weiterer schwerwiegender Nachteil ist die Vermischung von fachlichen Beschreibungen mit Aktivitäten zur Überwachung der Prozessausführung. Zum einen stellt die zunehmende Komplexität der Prozessbeschreibung ein großes Problem dar, insbesondere wenn ein sehr detailliertes Monitoring gewünscht wird. Soll zum Beispiel der Fortschritt des Prozesses lückenlos überwacht werden, so ist auf diese Weise die Integration von mindestens einer Monitoring-Aktivität pro Transition des Prozesses erforderlich, was den Umfang der Prozessbeschreibung mindestens verdoppelt (vgl. [RSS06b, ZBH+10]). Zum anderen verzögert sich die Prozessausführung durch die Ausführung der Monitoring-Aktivitäten und den Aufruf der entsprechenden Monitoring-Dienste bzw. des Monitoring-Managers entsprechend. Im schlimmsten Fall kann es dabei in Abhängigkeit der Leistungsfähigkeit der Monitoring-Komponenten und der dieser Kommunikation zugrunde liegenden Netzwerkverbindungen zu erheblichen Verzögerungen der funktionalen Prozessausführung kommen, insbesondere, wenn Monitoring-Komponenten temporär nicht zu erreichen sind [MRD08]. Für die Offline-Bearbeitung von Prozesspartitionen auf mobilen Geräten ist dieser Ansatz daher nicht in Betracht zu ziehen.

Ein großer Vorteil des vorgestellten Ansatzes liegt jedoch darin, dass der Prozessinitiator (auf Basis der bereitgestellten Möglichkeiten des Ansatzes) beliebige Überwachungsmaßnahmen für jeden seiner Prozesse – sowohl auf Prozessmodell- als auch (theoretisch) auf Prozessinstanzebene – nach seinen individuellen Anforderungen zusammenstellen kann. Durch die Integration der Monitoring-Aktivitäten auf Basis der verwendeten (Standard-)Prozessbeschreibung sind hierfür keine Anpassungen an Prozessmanagementsystemen erforderlich. Da die Überwachung der Prozessausführung für das entfernte Ausführungssystem gewissermaßen transparent erfolgt, muss der Überwachung auch keine aufwendige Verhandlung über die Bereitstellung von Informationen vorausgehen, welche die Flexibilität für dynamische Anpassungen weiter einschränken würden.

5.7.4 Ansatz von LUDWIG, DAN und KEARNEY

Eine andere Variante der elektronischen Überwachung von Kooperationspartnern beruht auf der Annahme, dass die kooperierenden Parteien einander technische Möglichkeiten zur Verfügung stellen, um die Ausführung von extern durchgeführten Funktionalitäten in einer verabredeten Art und Weise zu beobachten (vgl. Abbildung 5.18e). Basierend auf dem Ansatz von Crossflow (vgl. Abschnitt 5.2.3) schlagen die Autoren LUDWIG, DAN und KEARNEY mit *Cremona* (*Creation and Monitoring of Agreements*) [LDK04] eine Architektur für die Implementierung von *WS-Agreement* [ACD+04, ACD+07] vor, welche neben der Verhandlung von konkreten Dienstgütevereinbarungen auch die Bereitstellung der zu deren Überwachung erforderlichen Monitoring-Schnittstellen umfasst.

Die Aushandlung der Dienstgütekriterien kann dabei zur Laufzeit als Erweiterung der Interaktionen im klassischen SOA-Dreieck (vgl. Abschnitt 3.4.2) erfolgen. Hierfür stellt WS-Agreement ein anwendungsunabhängiges Format für Vereinbarungen (*Agreements*), ein grundlegendes Verhandlungsprotokoll und eine Schnittstellenbeschreibung zur Überwachung der vereinbarten Kriterien zur Verfügung [LDK04, ACD+04, ACD+07]. Zur Vereinfachung wird angenommen, dass die Kooperationspartner einander bereits bekannt sind bzw. dass es sich um eine bereits bestehende Geschäftsbeziehung handelt. Der Fokus von sowohl WS-Agreement als auch von Cremona liegt zudem auf der Dienstgütevereinbarung und Überwachung von einzelnen (atomaren) Diensten, so dass sowohl die Vereinbarung als auch die Überwachung entweder nur sehr allgemein beschrieben werden können oder durch komplexe domänenabhängige Inhalte gekennzeichnet sind. So ist zum Beispiel die Angabe des Zustands eines Dienstes bei WS-Agreement durch ein sehr einfaches Zustandsmodell begrenzt (vgl. Abbildung 5.20). Um dennoch detailliertere Vereinbarungen zu erreichen, schlagen die Autoren von Cremona die Spezifikation eines Implementierungsplans für Vereinbarungen (*Agreement Implementation Plan*, kurz *AIP*) vor, welcher automatisiert verarbeitet werden kann [LDK04]. Beispiele für solche Implementierungspläne sind unter ande-

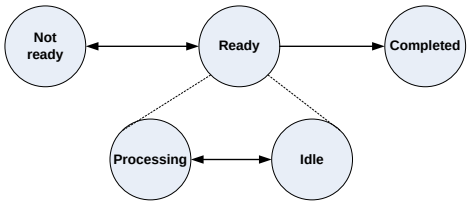


Abbildung 5.20: Zustandsmodell von WS-Agreement zur Beschreibung des möglichen Zustands eines Dienstes (ACD+07)

rem Prozessbeschreibungen im Sinne dieser Arbeit. Die Implementierung einer entsprechenden Monitoring-Komponente mit geeigneten Funktionalitäten zur Überwachung von Prozessen ist jedoch nicht Teil von WS-Agreement oder von Cremona.

Interessant ist jedoch in jedem Fall die Möglichkeit, die Art und Weise der Überwachung während der Verhandlung funktionaler und/oder nichtfunktionaler Aspekte zu inkludieren, um einen sowohl für die individuellen Bedürfnisse als auch für die Autonomie der beteiligten Parteien angemessenen Grad an Überwachung und – im weiterführenden Sinne – auch der Steuerung von gemeinsam ausgeführten Prozessen zu erreichen. Die technische und inhaltliche Umsetzung solcher Maßnahmen in Bezug auf verteilt ausgeführte prozessorientierte Anwendungen bleibt hierdurch jedoch weiterhin offen.

5.7.5 Monitoring abstrakter Sichten von Prozessen

Um zu einem Kompromiss zwischen der Autonomie der prozessausführenden Parteien und der Nachvollziehbarkeit der Prozessausführung zu gelangen, schlagen die Autoren CASTELLANOS, CASATI, SHAN und DAYAL [CCSD04] ein Konzept vor, welches die Definition von *abstrakten Sichten* über intern ausgeführte Prozesse erlaubt (ähnlich auch bei SCHULZ und ORLOWSKA [SO04]). Eine abstrakte Sicht stellt dabei selektiv Informationen zur Ausführung von ausgewählten Prozessen bzw. Prozessinstanzen zur Verfügung, welche einerseits die Vertraulichkeit interner Aspekte nicht verletzen und dem jeweiligen Dienstnutzer ein speziell auf dessen Bedürfnisse angepassten Einblick in die Prozessausführung gewährt. Um die Individualität verschiedener Dienstnutzer zu adressieren, können dabei verschiedene abstrakte Sichten desselben Prozesses für unterschiedliche Nutzer oder Nutzergruppen eingerichtet werden [CCSD04]. Aufbauend auf diesen abstrakten Sichten erlaubt das Konzept in diesem Rahmen zudem die Definition von nutzerspezifischen Metriken. Zur Realisierung wird mit dem *Abstract Process Monitor* ein spezielles Software-Werkzeug vorgeschlagen, welches den Benutzer auf Seiten des Dienstanbieters bei der Modellierung von abstrakten Prozessen unterstützt und die Zuordnung von Ereignissen der internen Informations- und Kommunikationssysteme auf den Beginn bzw. die Fertigstellung von Aktivitäten im abstrakten Prozess erlaubt.

Die Definition von abstrakten Sichten über Prozesse ist ein sehr elegantes Konzept, um die individuellen Bedürfnisse der Kooperationspartner sowohl in Hinblick auf Vertraulichkeit als auch in Hinblick auf die Bereitstellung von speziell relevanten Daten zu berücksichtigen. Aufgrund der aufwendigen Modellierung der abstrakten Sichten ist das Konzept jedoch insbesondere für länger andauernde Geschäftsbeziehungen geeignet, welche eine Wiederverwendbarkeit der entwickelten Modelle erlauben. Bei dynamisch verteilt ausgeführten Prozessen ist die Struktur dieser Prozesse jedoch den beteiligten Kooperationspartnern bekannt, so dass die Bildung der abstrakten Sichten keinen zusätzlichen Vorteil bringt. Die Überwachung abstrakter Sichten von Prozessen ist somit eher für die Ausführung gekapselter Subprozesse und lose gekoppelter Verteilungsmodelle (vgl. Abschnitt 4.3.3) geeignet.

5.7.6 Peer-to-Peer-basiertes Monitoring

Das Projekt *P2E2* [BKK+04, KW05, WW08] umfasst die dezentrale Modellierung, Ausführung und Überwachung von organisationsübergreifenden Prozessen auf Basis einer Peer-to-Peer-Infrastruktur mit verteilten Datenbanken. Hierbei kann zunächst jeder der Teilnehmer innerhalb der Infrastruktur Prozesse oder Prozessabschnitte spezifizieren (Schritt 1), wobei die Verteilung der Ausführung über die Kapselung von Subprozessen als elektronische Dienste geschieht. Während der Ausführung (Schritt 2) melden die beteiligten Peers Ablaufdaten ihrer Teilprozesse an eine ausgewählte *P2E2-Datenbank* (Schritt 3). Um die Autonomie der Teilnehmer zu berücksichtigen, ist die Menge und Auswahl der zu meldenden Daten freiwillig. Über eine Publish-Subscribe-Architektur wird ein Teil der Daten an andere P2E2-Datenbanken im verteilten System weitergemeldet (Schritt 4). Bei Bedarf können die Teilnehmer schließlich die Ablaufdaten der Prozesse abfragen. Dazu stellen sie eine Anfrage an einen *Prozess-Performance-Manager*, welcher hierzu die erforderlichen Daten von den P2E2-Datenbanken abruft oder andere Prozess-Performance-Manager damit beauftragt [BKK+04] (Schritt 5). Der beschriebene Ablauf mit den erläuterten Schritten ist exemplarisch in Abbildung 5.21 dargestellt.

Der hier beschriebene Ansatz stellt eine hybride Variante zwischen der Bereitstellung von Informationen durch die prozessausführende Partei (vgl. Abbildung 5.18e) und der gemeinsamen Datenhaltung zwischen Dienstnutzern (vgl. Abbildung 5.18c) dar, wobei hier die typische Charakteristik von Peer-to-Peer-Systemen zum Tragen kommt, dass jeder Teilnehmer sowohl die Rolle des Dienstnutzers als auch die Rolle des Dienstanbieters (bzw. des Beobachters und der zu beobachtenden prozessausführenden Partei) annehmen kann. Hierbei sind jedoch auch Vor- und Nachteile beider Varianten zu berücksichtigen. Zum einen besteht das bereits angesprochene Problem der Vertrauenswürdigkeit von freiwillig bereitgestellten Informationen (vgl. [JBF07, JFB07]), zum anderen besteht die Gefahr, dass es durch die Verteilung der Daten auf die P2E2-Datenbanken und vor allem durch die Suche nach den „richtigen“ Daten durch die *Prozess-Performance-Manager* zu

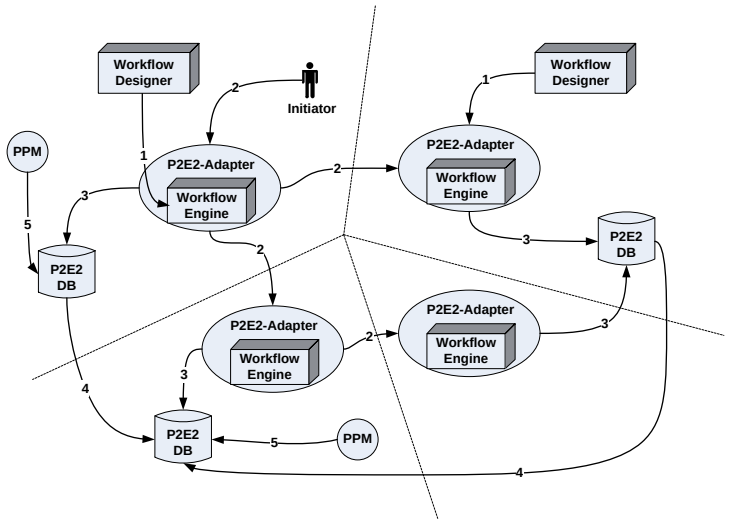


Abbildung 5.21: Architektur des P2E2-Ansatzes (BKK+04)

Zeitverzögerungen und zu ggf. unnötigem Kommunikationsaufwand kommen kann. Zudem ist eine spontane Anfrage von Monitoring-Daten (z. B. zum Fortschritt eines Prozesses) nicht möglich, wenn nicht im Voraus die entsprechenden Daten abonniert wurden.

5.7.7 Ansatz von WETZSTEIN et al.

In vielen Fällen beruhen Unternehmenskooperationen auf dem Auslagern ausgewählter Teilprozesse an externe Geschäftspartner, wobei die Durchführung des ausgelagerten Prozesses dennoch in einer mit dem Partner abgestimmten Art und Weise von der hierarchisch übergeordneten Organisation überwacht werden soll. Um die Verhandlung und den Austausch von Monitoring-Daten zu unterstützen, stellen WETZSTEIN et al. [WKK+10] einen Ansatz zur organisationsübergreifenden Überwachung von Prozessausführungen in dienstbasierten Choreographien vor. Dabei wird auf Basis einer zugrunde liegenden BPEL4Chor-Spezifikation (vgl. [DKLW07]) eine Vereinbarung ausgehandelt (*Monitoring-Agreement*), welche die Art von Ereignissen festhält, die zur Überwachung der entfernt ausgeführten Prozessschritte erforderlich sind [WKK+10]. Die Autoren unterscheiden dabei zwischen einfachen Ereignissen, welche durch Zustandsänderungen der in der Choreographie definierten abstrakten Prozesse auftreten, und komplexen Ereignissen, welche durch *Complex Event Processing* (*CEP*) (vgl. Abschnitt 3.7.5) aus einfachen Ereignissen zu höherwertigen Informationen abgeleitet werden können. Da hierfür die Korrelation von Ereignissen verschiedener Partner innerhalb einer Choreographie erforderlich ist, wird zudem die Notwendigkeit eines eindeutigen Identifikators für jede Instanz einer Choreographie (*Choreography Instance Identifier*) motiviert [WKK+10].

Für die Erstellung von Monitoring-Agreements schlagen die Autoren zwei mögliche Vorgehensweisen vor: Zum einen kann das Monitoring-Agreement

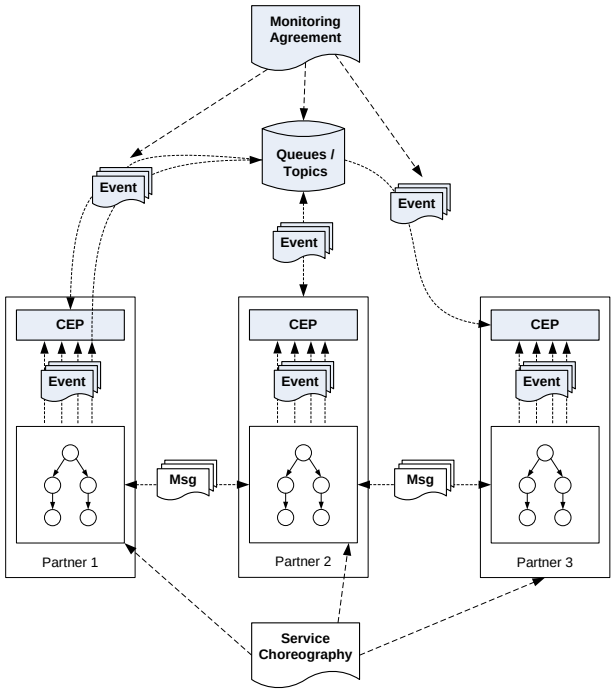


Abbildung 5.22: Austausch von Ereignissen zum Monitoring von Choreographien (WKK+10)

zur Entwicklungszeit der Choreographie erstellt und in diesem Zuge festgelegt werden, welche erforderlichen Ereignisse auf der Ebene der Infrastruktur veröffentlicht bzw. verarbeitet werden sollen (*Top-Down-Ansatz*). Zum anderen besteht prinzipiell die Möglichkeit, die individuellen Anforderungen und Angebote der (potentiellen) Geschäftspartner bezüglich der Bereitstellung von Ereignissen zu spezifizieren und zur Laufzeit einen Matchmaking-Prozess durchzuführen, welcher in einem Monitoring-Agreement resultiert (*Bottom-Up-Ansatz*). Die Durchführung des Monitorings geschieht durch ein Deployment des Monitoring-Agreements auf den jeweiligen Partnersystemen, was das Abonnement bzw. das Veröffentlichen der relevanten Ereignisse und eine Verarbeitung der Ereignisse zur Laufzeit zur Folge hat [WKK+10]. Abbildung 5.22 visualisiert einen Überblick über den Ansatz im Allgemeinen.

Das dargestellte Konzept setzt die Forderung nach einer ereignisbasierten Überwachung von Prozessen im Rahmen des *Event-driven (Business) Process Management* (vgl. Abschnitt 3.7) für verteilt ausgeführte Prozesse um und erlaubt somit in erster Linie einen zeitnahen Empfang von individuell relevanten Ereignissen der an der Prozessausführung beteiligten Parteien. Die auf Publish-Subscribe basierende Architektur würde dabei prinzipiell auch das Abonnement von Ereignissen zur Laufzeit von Prozessen und damit eine dynamische Anpassung der zu überwachenden Parameter erlauben. Als Ausgangspunkt für eine Anpassung der Prozessausführung ist ein solcher, ausschließlich auf der passiven Beobachtung von Ereignissen begrenzter Ansatz jedoch nicht ausreichend, da hierbei lediglich Anpassungsbedarf abgeleitet, nicht aber

eine Anpassung selbst durchgeführt werden kann. Neben relevanten Schnittstellen zur Steuerung der Prozessausführung fehlen zudem Möglichkeiten zum Abruf von Monitoring-Daten nach dem *Pull-Modell*, die das hier umgesetzte *Push-Modell* ergänzen [WKK+10]. Somit wären zum Beispiel auch spontane Anfragen nach dem Status eines bestimmten Prozesses *bei Bedarf* möglich, ohne dafür bei jeder Zustandsänderung aller Prozessinstanzen jeweils eine Nachricht senden, empfangen und verarbeiten zu müssen.

5.7.8 Ansatz von MOSER, ROSENBERG und DUSTDAR

Im Gegensatz zur Anpassung von Prozessmodellen oder -instanzen schlagen die Autoren MOSER, ROSENBERG und DUSTAR [MRD08] vor, die Überwachungs- und Ausführungslogik in die Ausführungsumgebung zu verlagern. Motivation für Ihre Arbeit sind die fehlende Anpassbarkeit von ausführbaren WS-BPEL-Prozessen sowie die fehlenden Mittel zur Überwachung laufender WS-BPEL-Instanzen in aktuellen Ausführungsumgebungen. Als wesentliches konzeptuelles Ziel wird angestrebt, dass das ursprünglich auf der Prozess-Engine installierte Prozessmodell auch bei einer notwendigen technischen Anpassung erhalten bleibt und die darin enthaltene fachliche Logik nicht mit technischen Anweisungen vermischt wird. Der vorgestellte Ansatz *VieDAME* (*Vienna Dynamic Adaptation and Monitoring Environment for WS-BPEL*) erweitert dazu die Prozess-Engine *ActiveBPEL*[1] um eine zusätzliche Softwareschicht, welche einerseits beim Aufruf eines konkreten Dienstes die aufzurufende Dienstinstanz austauschen kann, eine semantisch äquivalente Dienstinstanz (ggf. über eine Adapterkomponente) einbinden kann und den entsprechend durch diese Softwareschicht erfolgten Nachrichtenaustausch überwachen kann [MRD08].

Um benutzerdefinierte Vorgaben zu berücksichtigen, kann spezifiziert werden, ob ein Dienst als *substituierbar* gekennzeichnet werden soll. Substituierbare Dienste können zur Laufzeit automatisch (z. B. auf Basis vorgegebener Policies) durch syntaktisch oder semantisch äquivalente Dienste ersetzt werden. Dazu werden alternative Dienste in einem zentralen Dienstverzeichnis gehalten, welches vom Benutzer über eine Webanwendung verwaltet werden kann. Um möglichst wenig in den Quellcode der manipulierten Prozess-Engine eingreifen zu müssen, schlagen die Autoren die Implementierung der benötigten Adapterkomponenten mittels *Aspekt-orientierter Programmierung* (*AOP*) vor. Zur Laufzeit kann dann entschieden werden, ob die Standard-Funktionalitäten der Prozess-Engine oder die erweiterten Funktionalitäten zur Anpassung und Überwachung der Prozessausführung genutzt werden sollen [MRD08].

Der Ansatz von Moser, Rosenberg und Dustar bietet eine hohe Flexibilität und vereint mit der Überwachung und der Anpassung von laufenden Prozessinstanzen wichtige Voraussetzungen für eine automatisierbare technische Anpassungsfähigkeit von prozessorientierten Anwendungen ohne diese dabei

[1] Produkt von *Active Endpoints*, siehe http://www.active-endpoints.com.

selbst in ihrer fachlichen Struktur zu modifizieren. Somit sind auch spontane Änderungen zur Laufzeit eines Prozesses möglich. Der Ansatz ist jedoch vollständig auf die Betrachtung einer einzelnen lokalen Prozess-Engine und der Auswahl und Einbindung von einzelnen Diensten ausgerichtet und verfolgt somit nur eine ressourcenbezogene Sichtweise der Verteilung. Zudem wird bei der Überwachung des lokalen Prozesses wiederum nur das nach außen sichtbare Verhalten durch den Nachrichtenaustausch des Prozesses berücksichtigt.

5.7.9 Ansatz von VAN LESSEN et al.

Das Fehlen von Möglichkeiten zum Zugriff auf eine entfernt ausgeführte Prozesspartition macht die Auseinandersetzung mit Ansätzen zur Steuerung von Prozessmanagementsystemen erforderlich, welche ein ausreichendes Maß an Interoperabilität zur Anpassung verteilt ausgeführter Prozesse unterstützen. Eine ganze Reihe von Operationen zur Verwaltung von Prozessmodellen und -instanzen werden dabei oft bereits durch das lokale Prozessmanagementsystem zur Verfügung gestellt. Hierzu gehören typischerweise Funktionen zum Starten und Abbrechen von Prozessinstanzen, zum Abrufen des Prozessfortschritts oder zur Änderung von Prozessvariablen (vgl. [Str09, Wun09]). Der Ansatz von VAN LESSEN et al. [vLLM+08] strebt eine *Management-API* zum einheitlichen Zugriff auf WS-BPEL-Prozessmodelle und -instanzen an, um von herstellerspezifischen Management-Funktionalitäten zu abstrahieren und die Portabilität von Management-Anwendungen auf der Basis einer einheitlichen Schnittstelle zu erhöhen. Dazu werden sowohl Prozessmodelle als auch Prozessinstanzen als *verwaltbare Ressourcen* (vgl. Abschnitt 3.8.3) betrachtet, welche jeweils in gleicher Art und Weise zugegriffen und manipuliert werden können. Die hierfür erforderlichen Funktionalitäten können dabei entweder direkt auf dem unterliegenden Datenmodell (hier dem WS-BPEL-Prozessmodell) operieren, oder den bereits existierenden herstellerspezifischen Schnittstellen zugeordnet werden [vLLM+08]. Abbildung 5.23 zeigt die Vereinheitlichung des internen Datenmodells einer WS-BPEL-Prozess-Engine in Hinblick auf Prozessmodelle und Instanzen.

Verwaltbare Ressourcen können in sich geschachtelt sein und somit zum Beispiel die Beziehung zwischen Prozessinstanzen und Aktivitätsinstanzen geeignet abbilden. Der Ansatz erlaubt zudem einen einheitlichen Zugriff auf Ereignisse, welche im Kontext der Ausführung laufender Prozessinstanzen erzeugt werden. Hierbei können sich Beobachter für Ereignisse bestimmter Ressourcen registrieren, z. B. einzelne Aktivitäten eines Prozessmodells bei der Ausführung aller Prozessinstanzen oder auch nur einzelne Prozessinstanzen ereignisbasiert überwachen. Für die Anpassung von Parametern laufender Prozessinstanzen wird zudem das Konzept der *blockierenden Ereignisse* vorgeschlagen, welche verhindern, dass die Prozessausführung vor oder während der Durchführung von Anpassungsmaßnahmen fortgesetzt wird [vLLM+08]. Der Zugriff auf historische Daten des Audit Trails kann ebenfalls auf Basis verwaltbarer Ressourcen geschehen, indem das interne Datenmodell laufender

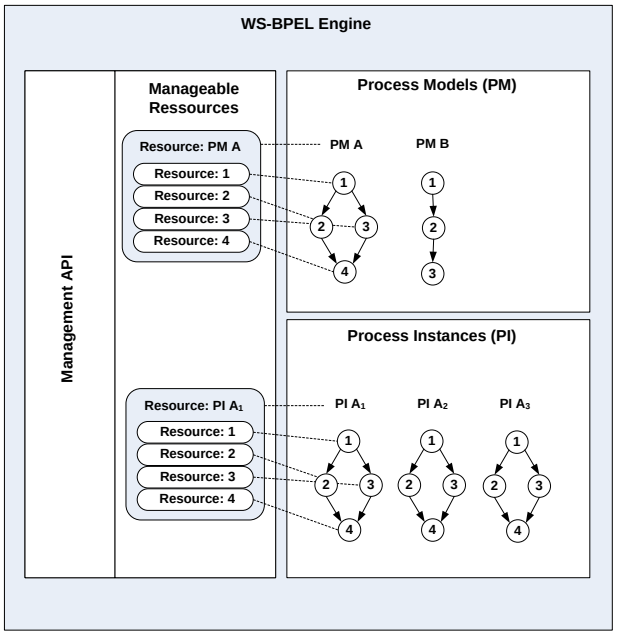


Abbildung 5.23: Repräsentation von WS-BPEL-Prozessmodellen und -instanzen als verwaltbare Ressourcen (*Manageable Resources*) (vLLM+08)

Prozessinstanzen auch auf die persistent gespeicherten Vergangenheitswerte angewendet wird.

Durch die Integration einer einheitlichen Management-API mit ereignisbasierter Überwachung von Prozessmodellen und Prozessinstanzen, der Berücksichtigung historischer Daten und der Bereitstellung sowohl passiver Überwachungs- als auch aktiver Steuerungsfunktionalitäten wird durch diesen Ansatz ein hohes Maß an Interoperabilität für das Management verteilt ausgeführter Prozesse zur Verfügung gestellt. Abfragen und Änderungen in Bezug auf Prozessmodelle und -instanzen sind jederzeit möglich und das Abonnement von Ereignissen kann gegebenenfalls zur Laufzeit verändert oder ergänzt werden. Für eine umfassende Akquisition von Daten ist dieser Ansatz zur Anpassung verteilt ausgeführter Prozesse jedoch noch um Funktionalitäten zu ergänzen, welche das ausführende Prozessmanagementsystem selbst (z. B. dessen Auslastung) oder dessen Kontext (z. B. den aktuellen geographischen Ort) integrieren. Zudem ist die strenge Ausrichtung auf WS-BPEL für ein Konzept zur Anpassung von Prozessen im Allgemeinen zu speziell und lässt die Anwendung auf andere (Standard-)Prozessbeschreibungssprachen und -managementsysteme offen. Schließlich ist zu klären, wie das Management von Prozessausführungssystemen unterstützt werden kann, welche nur temporär an ein Kommunikationsnetzwerk angebunden sind (zum Beispiel Mobilgeräte).

5.8 Benutzungsschnittstellen für verteilte Prozesse

Während die dynamische Auswahl und Einbindung von Ressourcen im Fall von Softwareanwendungen zumindest bei einer Realisierung durch dienstorientierte Architekturen durch die Nutzung selbstbeschreibender Schnittstellen adressiert werden kann (vgl. Abschnitt 3.4), ist die Repräsentation einer geeigneten Benutzungsschnittstelle bei einem dynamisch verteilt ausgeführten Prozess noch genauer zu untersuchen. Dabei besteht das Hauptproblem darin, dass bei einem dynamisch verteilten Prozess nicht zur Entwicklungszeit vorhergesehen werden kann, auf welchem Gerät ein menschlicher Benutzer zur Laufzeit an der Bearbeitung einer Aktivität beteiligt sein wird (vgl. Abschnitt 4.2.3). Die konkrete Repräsentation der Interaktion mit dem Benutzer, welche für die Bearbeitung der entsprechenden Aktivität erforderlich ist, muss daher zur Laufzeit an den aktuellen Kontext des Benutzers anpassbar sein. Da diese Problematik bei den betrachteten Ansätzen zur Verteilung der Prozessausführung nicht berücksichtigt wurde, werden im Folgenden verschiedene Ansätze anderer Forschungsbereiche zur Realisierung von Benutzungsschnittstellen untersucht, welche in verteilten Prozessen und insbesondere im Rahmen des Prozessmanagements auf mobilen Geräten zum Einsatz kommen können.

5.8.1 Integration von Aufgabenbeschreibungen

Um die flexible Verteilung eines Prozesses mit Benutzerinteraktionen zu ermöglichen, muss zunächst überprüft werden, auf welche Weise die Beschreibung der Aufgabe, die der Benutzer auszuführen hat, in den Prozess integriert werden kann. Hierbei können verschiedene Architekturmodelle unterschieden werden [CR97, KKL+05, AAD+07b], welche in Abbildung 5.24 gegenübergestellt sind. Abbildungen 5.24a und 5.24b zeigen zwei Varianten, in denen die Aufgabenbeschreibungen für den Benutzer in den Prozess selbst eingebettet und somit fester Teil der Prozessspezifikation sind. Hierbei werden Sprachelemente zur Abbildung von Formularen oder Fenstern integriert, welche dem Benutzer zur Bearbeitung der Aktivität durch die Ausführungseinheit angezeigt werden. Zudem werden relevante Eingabe- und Ausgabedaten integriert bzw. vom Benutzer entgegengenommen. Ist die Aufgabenbeschreibung Teil einer Aktivität (vgl. Abbildung 5.24a), so kann sie nur innerhalb dieses Elements zugegriffen und nicht in anderen Aktivitäten wiederverwendet werden. Alternativ kann die Aufgabenbeschreibung innerhalb des Prozesses auf einer höheren Ebene spezifiziert und von mehreren Aktivitäten des Prozesses referenziert werden (vgl. Abbildung 5.24b), was die Wiederverwendbarkeit und Wartbarkeit der Beschreibung erhöht [AAD+07b]. Beide Varianten erfordern eine hohe Homogenität der technischen Infrastrukturen, da die integrierte Beschreibung sowohl von der Ausführungseinheit überall gleich interpretiert und durch die unterliegende Hardware angemessen dargestellt werden muss. Zu-

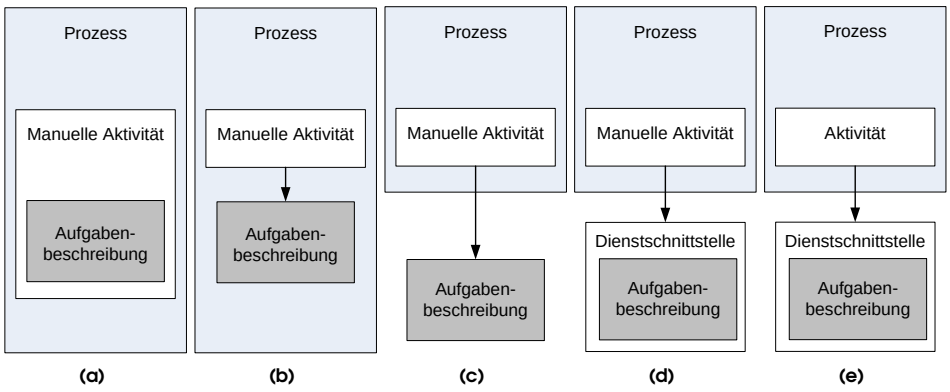


Abbildung 5.24: Varianten der Integration von Aufgabenbeschreibungen für die manuelle Aktivität eines Prozesses (tlw. nach (KKL+05, AAD+07b))

dem wird durch diese Vorgehensweise die Komplexität und der Umfang der Prozessbeschreibung unter Umständen stark erhöht.

Abbildungen 5.24c bis 5.24e zeigen Architekturmodelle, bei denen die Aufgabenbeschreibung nicht Teil der Prozessspezifikation ist, sondern prozessunabhängig in der lokalen oder einer entfernten Ausführungsumgebung gespeichert ist. Hierbei ist die Wiederverwendbarkeit der Aufgabenbeschreibung nochmals erhöht, da diese nun von Aktivitäten verschiedener Prozessmodelle referenziert werden kann. Zudem kann die Aufgabenbeschreibung (ggf. sogar zur Laufzeit des Prozesses) geändert werden, ohne dass es zu Modifikationen am Prozessmodell kommen muss. Die Variante in Abbildung 5.24c stellt eine lokal verfügbare Aufgabenbeschreibung dar, welche über eine proprietäre Schnittstelle zugegriffen werden kann, während Abbildung 5.24d die Kapselung der Aufgabenbeschreibung als (entfernter) elektronischer Dienst mit einer unter Umständen standardisierten Schnittstelle (z. B. als Web Service) visualisiert. Durch die lose Kopplung der Aufgabenbeschreibung ist bei diesen Varianten jedoch in beiden Fällen kein Prozesskontext auf Seiten der Aufgabenbeschreibung verfügbar und muss somit bei Abruf der Aufgabenbeschreibung integriert werden [AAD+07b]. Abbildung 5.24e zeigt schließlich die Integration der Benutzerinteraktion durch einen Dienstaufruf, der das Vorhandensein einer manuellen Aktivität zudem gänzlich vor dem Prozessausführungssystem verbirgt. Dies bedeutet, dass bei der Spezifikation des Prozesses transparent ist, ob die Aktivität automatisch (d. h. durch eine Softwareanwendung) oder durch einen Benutzer ausgeführt wird. Durch die Prozess-Engine wird bei der Ausführung der entsprechenden Aktivität z. B. ein Dienst aufgerufen, der einem etwaigen Benutzer im Hintergrund eine proprietäre Benutzungsschnittstelle präsentiert. Vorteile dieser Variante liegen in der Leichtgewichtigkeit und der Plattformunabhängigkeit der Prozessbeschreibung, da dort keinerlei Vorkehrungen für eine Interaktion mit dem Benutzer getroffen werden müssen. Zudem kann dem Benutzer auf diese Weise stets eine vertraute Schnittstelle geboten werden, welche auch speziell auf die

besonderen Eigenschaften und Einschränkungen der verwendeten Hardware (z. B. Mobilgerät) abgestimmt werden kann [CR97]. Auf der anderen Seite schränkt dieser Ansatz die Flexibilität bei der Gestaltung von Benutzeraufgaben in einer Prozessbeschreibung stark ein, da hierbei entweder von Seiten der Prozessausführung nur bestehende Dienste für die Interaktion mit dem Benutzer genutzt werden können oder für jede Art von Benutzerinteraktion auf jedem potentiell verwendeten Gerät zunächst ein eigener spezieller Dienst bereitgestellt werden muss.

Während die externe Spezifikation von Aufgabenbeschreibungen für stationäre Prozessmanagementsysteme mit dauerhafter Netzwerkanbindung die Flexibilität durch die genannten positiven Aspekte in Bezug auf Wiederverwendbarkeit und Wartbarkeit der Beschreibungen erhöht, ist der entfernte Abruf von zusätzlichen Beschreibungen für die Benutzerinteraktion im Kontext mobiler Geräte problematisch. In vielen Fällen werden hierbei aufwendige graphische Benutzungsschnittstellen auf einem zentralen Webserver bereitgestellt, welche über eine Internetverbindung und einen einfachen Browser zugänglich gemacht werden. Dabei sind zum einen die Darstellung der Aufgabenbeschreibung und die Eingabemöglichkeiten oft nicht für das verwendete mobile Gerät geeignet, zum anderen erschwert der Abruf der Interaktions- und Darstellungsdetails von einem entfernten System die geforderte Offline-Bearbeitung der Aktivitäten. Im Folgenden werden daher bestehende Konzepte für die potentielle Interaktion mit (mobilen) Benutzern auch während einer Offline-Bearbeitung zuvor nicht bekannter Aktivitäten untersucht.

5.8.2 WS-Human-Task und WS-BPEL4People

Da die Prozessbeschreibungssprache WS-BPEL (vgl. Abschnitt 5.2.4) lediglich die Komposition von Web Services berücksichtigt, stellen die Erweiterungen *WS-Human-Task* [AAD+07a] und *WS-BPEL4People* [AAD+07b] zusätzliche Möglichkeiten zur (abstrakten) Beschreibung von benutzergesteuerten Aufgaben, die Definition von organisatorischen Rollenbeziehungen und die Umsetzung von typischen Interaktionsmustern zur Verfügung.

Ziel von WS-Human-Task ist es zum einen, die Portabilität von Aktivitäten mit Benutzerinteraktion (*Human Interactions*) zu erreichen, so dass sowohl Aufgabenbeschreibungen (*Human Tasks*) als auch Benachrichtigungen (*Notifications*) plattformunabhängig spezifiziert und auf den Systemen unterschiedlicher Hersteller in gleicher Weise zur Verfügung gestellt werden können. Zum anderen soll Interoperabilität zwischen den technischen Komponenten erreicht werden, welche an der Interpretation, Darstellung und Auswertung der Benutzerinteraktion beteiligt sind. Hierzu werden einheitliche Nachrichtenformate und Kommunikationsprotokolle vorgeschlagen [AAD+07a]. Die Darstellung einer Aktivität mit Benutzerinteraktion erfolgt über die Verknüpfung einer *Rendering*-Methode, welche über einen eindeutigen Identifikator den Typ der Darstellung angibt, der für diese Aktivität gewählt werden soll. Dabei können mehrere Rendering-Methoden für jede Aktivität existieren, so dass

für verschiedene Ausführungsumgebungen jeweils eine geeignete Darstellung gewählt werden kann. Zudem existieren eine Reihe von Metadaten, welche zum Beispiel den Namen, die Rollenzuordnung, die Priorität und die zu bearbeitenden Daten spezifizieren [AAD+07a].

Darauf aufbauend integriert die Spezifikation WS-BPEL4People [AAD+07b] die von WS-Human-Task beschriebenen Aktivitäten mit Benutzerinteraktion in die WS-BPEL-Prozessbeschreibung. Hierzu repräsentiert das neue Element *PeopleActivity* innerhalb von WS-BPEL einen Behälter für eine Aktivität, welche eine Interaktion mit einem menschlichen Benutzer erfordert. Des Weiteren werden zusätzliche Konstrukte zur Spezifikation von Mustern zur Zuweisung von Aufgaben zu Benutzern (vgl. Abschnitt 2.6 oder [RHE05]) und zur Übergabe von Prozessvariablen bereitgestellt. Hierzu werden insbesondere die für die Benutzerinteraktion zu verwendenden Ein- und Ausgabeparameter der übergeordneten WS-BPEL-Spezifikation übergeben und den Feldern der Aufgabenbeschreibung zugeordnet, um eine dynamische Zuweisung zu erlauben [AAD+07b]. Durch beide Spezifikationen kann somit ein Abstraktionsgrad geschaffen werden, welcher die Einbindung der Benutzerinteraktion im Rahmen der in den Abbildungen 5.24a bis 5.24d gezeigten Varianten erlaubt.

Obwohl die Ziele der Portabilität und Interoperabilität von Aktivitäten mit Benutzerinteraktion im Vordergrund stehen, liegt der Fokus beider Spezifikationen auf der Bereitstellung der in WS-BPEL fehlenden Konzepte, welche bei anderen Prozessbeschreibungssprachen (z. B. BPMN 2.0 oder XPDL 2.1, vgl. Abschnitt 5.2.4) zum Teil vorhanden sind. Dabei abstrahieren sie vollständig von der Repräsentation der Benutzeraufgaben und stellen somit lediglich ein Rahmenwerk für deren genauere Beschreibung dar [ZVBK09, ZBV09].

5.8.3 Ansatz von CHANDE und PAJUNEN

In vielen der betrachteten Arbeiten zum Prozessmanagement im Kontext mobiler Geräte wird die Problematik heterogener Benutzungsschnittstellen für (verteilte) Prozesse nicht explizit berücksichtigt (z. B. [AGK+95, CR97, Sch01, BMM04, HSH+06, SHH+07, SRG08, Kun08]). Dies liegt in vielen Fällen daran, dass nur rein dienstbasierte Prozesse (z. B. WS-BPEL-Prozesse) betrachtet werden (z. B. [BMM04, HHGR06, HGR07]) oder die Prozesse (teilweise) zentral und/oder statisch auf dem mobilen Gerät ausgeführt werden, ohne dass es zu einer dynamischen Verteilung kommt (z. B. [HHGR06, SRG08]).

Im Gegensatz dazu adressieren die Autoren CHANDE und PAJUNEN [PC07, CP07] in ihren Arbeiten die Konzeption einer vollwertigen Prozess-Engine für mobile Geräte, welche explizit die Offline-Bearbeitung von Prozessen mit allen benötigten Daten und die lokale Durchführung von Benutzerinteraktionen erlaubt. Basierend auf einer dienstorientierten Architektur kann der Benutzer eines mobilen Geräts *XHTML*-Formulare bearbeiten, welche durch eine lokale WS-BPEL-Engine in Abhängigkeit eines vordefinierten Kontrollflusses angezeigt werden. Dabei erfolgt die Anzeige der Formulare und die Verarbeitung der vom Benutzer eingegebenen Ergebnisdaten durch eine Kommunikati-

on zwischen einem als Dienst gekapselten Browser und der WS-BPEL-Engine über SOAP-Nachrichten. Neben dem Browser können auf diese Weise auch noch andere lokale Anwendungen auf dem mobilen Gerät integriert werden, zum Beispiel zur Anzeige von Kartenmaterial oder Kalenderterminen [PC07]. Um die Benutzerfreundlichkeit während der Prozessausführung zu erhöhen, schlagen die Autoren mit *ActiveForms* [CP07] einen weitergehenden Ansatz vor, der für verschiedene mobile (prozessorientierte) Anwendungen eine einheitliche Benutzungsschnittstelle darstellen soll. Ziel hierbei ist es, die Zahl der Interaktionen mit verschiedenen Anwendungen durch eine weitestgehende Automatisierung möglichst zu minimieren. Die Realisierung erfolgt wiederum über eine lose gekoppelte Laufzeitumgebung zur Darstellung von *XHTML* oder *XForms* [CP07] und WS-BPEL-Prozesse. Eine Integration mit Konzepten von *WS-Human-Task* [AAD+07a] und *WS-BPEL4People* [AAD+07b] ist hierbei nicht vorgesehen.

Die Verwendung von webbasierten Markup-Sprachen wie *(X)HTML*, *XForms* oder *VoiceXML* beschränkt die Interaktionsmöglichkeiten mit dem Benutzer auf eine bestimmte vorgegebene *Modalität* (d. h. zum Beispiel auf Text oder Sprache), so dass eine kontextbasierte Interaktion zur Laufzeit des Prozesses nur eingeschränkt möglich ist (vgl. Abschnitt 3.5.5). Des Weiteren kann aufgrund der Heterogenität und der eingeschränkten Leistungsfähigkeit von mobilen Geräten nicht davon ausgegangen werden, das auf jedem Gerät ein geeigneter Browser zur Verfügung steht, welcher die gewählte Darstellungsweise unterstützt. Der Aufruf von konkreten Diensten zur Kapselung der Benutzerinteraktion ist ebenfalls problematisch, da bei einer dynamisch verteilten Ausführung des Prozesses konkrete Dienstzugriffspunkte unter Umständen nicht im Voraus bekannt sind [ZVBK09, ZBV09].

5.8.4 Kontextbasierte Anpassung von Benutzungsschnittstellen

Um eine stärker kontextbasierte Anpassungsfähigkeit von Benutzungsschnittstellen für prozessorientierte Anwendungen zu erreichen, schlagen die Autoren ARDISSONO et al. [AFG+07, AGP08] ein rollenbasiertes Konzept auf Basis von Formatvorlagen mit unterschiedlichem Inhalt und Layout vor. Sobald eine Aufgabe einem bestimmten Benutzer zugewiesen wird, wird auf der Grundlage von Kontexteigenschaften des Benutzers und des verwendeten Geräts durch eine zentrale Komponente eine passende Formatvorlage gewählt und daraus eine geeignete Ein- und Ausgabeschnittstelle generiert. Durch Gruppierung und Aufteilung des Inhalts in mehrere Teilmengen kann das Layout dabei an unterschiedliche Displays und Schriftgrößen angepasst werden. Zusätzliche Informationen können bei Bedarf, z. B. über eine entsprechende Schaltfläche, angefordert werden [AFG+07, AGP08]. In einem ähnlichen Ansatz nutzen die Autoren MODAFFERI et al. [MBCP05] den oben beschriebenen Ansatz der Definition von *kontextabhängigen Bereichen* (vgl. Abschnitt 3.5.3), um eine abstrakte Aktivität zur Laufzeit zu konkretisieren und dabei angepasste Benutzeroberflächen mit einer geeigneten Repräsentation zuzuweisen. Abbildung 5.25

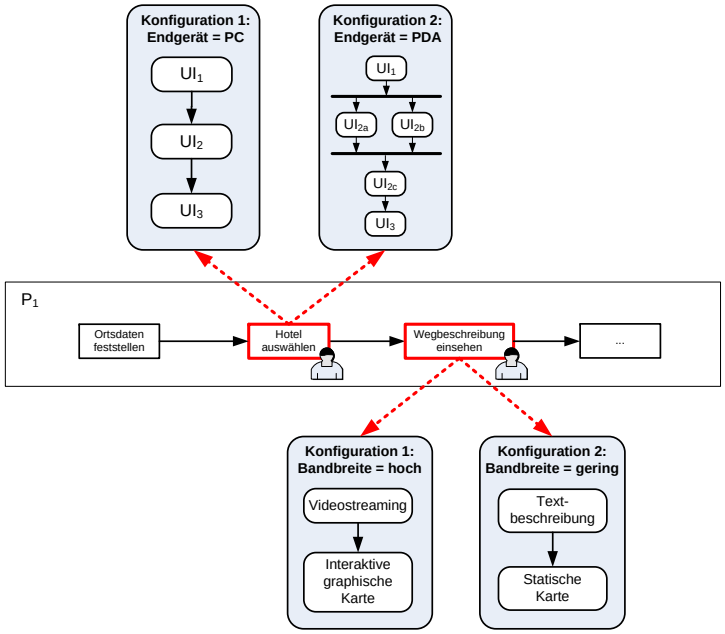


Abbildung 5.25: Anpassung der Benutzungsschnittstelle durch kontextabhängige Bereiche (MBCP05)

zeigt ein entsprechendes Beispiel für den Austausch eines interaktiven graphischen Stadtplans gegen eine textbasierte Beschreibung von Orten im Falle von unzureichender Übertragungsbandbreite [MBCP05].

Beide Ansätze erlauben sehr weitreichende Anpassungen sowohl des Inhalts als auch der Darstellung einer Benutzerinteraktion. In beiden Fällen wird jedoch auch eine Menge im Voraus festgelegter Konfigurationen benötigt, welche zur Laufzeit der prozessorientierten Anwendung zugewiesen werden kann. Dies entspricht dem Aufbau mehrerer Handlungsalternativen zur Erhöhung der A-priori-Flexibilität (vgl. Abschnitte 3.3.1 bzw. 3.3.2). Insbesondere bei der Integration einer großen Anzahl von heterogenen Endgeräten führt die Einbettung von entsprechend vielen verschiedenen Repräsentationen zu einer problematischen Erhöhung der Komplexität der Prozessbeschreibung. Bei dem Ansatz von ARDISSONO et al. [AFG+07, AGP08] wird daher hierfür eine zentrale Komponente verwendet. Eine Offline-Bearbeitung von Aufgaben bei entsprechender Berücksichtigung aktueller Kontextdaten ist dabei jedoch wie bereits in Abschnitt 5.8.1 diskutiert wurde, nicht ohne weiteres möglich.

Die Autoren CHAARI et al. [CELS07] wählen aus den genannten Gründen einen anderen Ansatz, welcher basierend auf einem Ontologiemodell für Kontextinformationen die umfassende Anpassung von (mobilen) Anwendungen an den Kontext in dynamischen Umgebungen verfolgt. Dabei wird auch die Anpassung der Präsentation von Benutzungsschnittstellen adressiert [CL05, CEL06], welche bei diesem Ansatz der Kommunikation mit Diensten einer dienstorientierten Architektur dienen. Auf Basis der erwarteten Ein- bzw. Ausgabeparameter eines zu verwendenden Dienstes kann hierbei die Ableitung einer abstrakten Beschreibung der Benutzungsschnittstelle automatisiert wer-

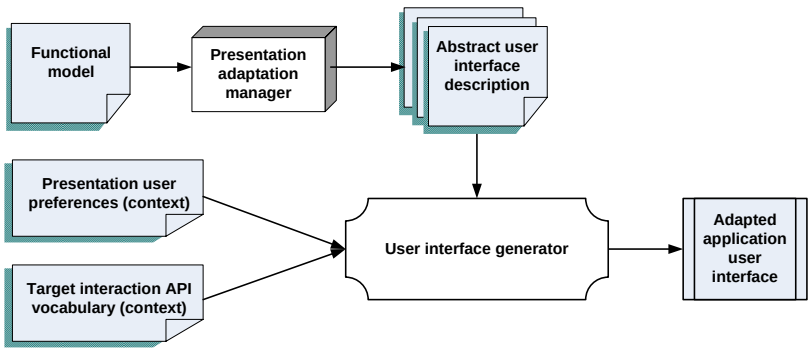


Abbildung 5.26: Architektur zur kontextbasierten Anpassung von Benutzungsschnittstellen (CELS07)

den. Abbildung 5.26 zeigt die von CHAARI et al. vorgeschlagene Architektur, bei der auf Basis dieser abstrakten Beschreibung (*Abstract User Interface Description*), der auf der Zielplattform tatsächlich vorhandenen Konstrukte zur Repräsentation der Benutzungsschnittstelle (*Target Interaction API Vocabulary*) und der individuellen benutzerspezifischen Einstellungen (*Presentation User Preferences*) eine angepasste Benutzungsschnittstelle (*Adapted Application Interface*) zur Verfügung gestellt werden kann [CELS07]. Hierbei kann zudem ein Teil der Benutzungsschnittstelle statisch vorverarbeitet, in einem lokalen Repository zwischengespeichert und bei Bedarf durch geringe Änderungen an den aktuellen Kontext angepasst werden [CL05, CEL06].

Die gewählte Sprache zur Beschreibung der Benutzerinteraktionen (*Abstract Window Description*, kurz *AWD*) [CL05, CEL06, CELS07] ist jedoch relativ restriktiv und fokussiert die visuelle Darstellung von einfachen Fenstern, um Eingaben des Benutzers an einen funktionalen Dienst zu senden und dessen Rückgabewerte zu repräsentieren. Die unvorhergesehene Interaktion mit einem Prozessmanagementsystem zur Bearbeitung beliebiger manueller bzw. interaktiver Aufgaben erfordert jedoch komplexere Möglichkeiten zur Beschreibung von Benutzungsschnittstellen. Der jedoch hinsichtlich der Flexibilität durchaus vorteilhafte Ansatz einer abstrakten Beschreibung von Interaktionselementen und die dynamische Ableitung spezialisierter Benutzungsschnittstellen wird im folgenden Abschnitt weiter vertieft.

5.8.5 Abstrakte und modellbasierte Benutzungsschnittstellen

Eine Verbesserung der Flexibilität von Benutzungsschnittstellen im Kontext dynamisch verteilt ausgeführter Prozesse kann durch die Einführung eines zusätzlichen Zwischenschritts in Form einer abstrakten Beschreibung der relevanten Interaktion erreicht werden, welche zur Entwicklungs- oder sogar zur Laufzeit der Anwendung verfeinert und auf den aktuellen Kontext des Benutzers angepasst werden kann. Entsprechende Vorgehensweisen werden derzeit bei der Entwicklung von (nicht zwingend verteilten) Anwendungen für mobile Geräte angewandt, wenn die mobile Anwendung für verschiedene End-

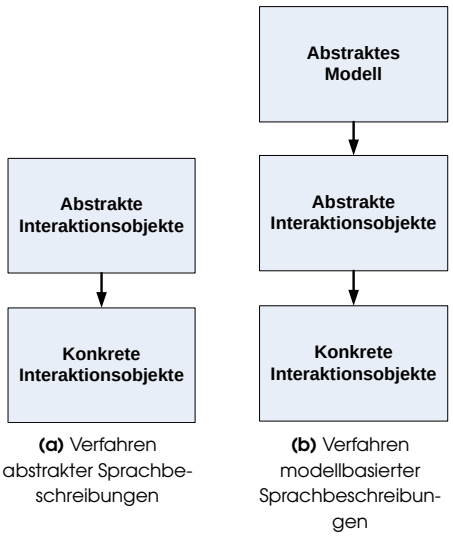


Abbildung 5.27: Verfahren zur Entwicklung und Verfeinerung von Bedienoberflächen (tlw. nach (SWT05, Vil08))

geräte entwickelt und die Art der *Bedienoberfläche* auf das spezielle Gerät abgestimmt werden soll [Pat00, SWT05]. Die Aufgabenstellung entspricht somit im Wesentlichen der Problematik der dynamisch an verschiedene Endgeräte verteilten Prozesse aus Abschnitt 4.2.3.

Beschreibungssprachen zur schrittweisen Entwicklung und Verfeinerung von Bedienoberflächen können in *abstrakte* und *modellbasierte* Ansätze klassifiziert werden [SWT05, Vil08] (vgl. Abbildung 5.27). Abstrakte Ansätze sind durch ein zweistufiges Verfahren gekennzeichnet, wobei zunächst generische (plattformunabhängige) Elemente zur Beschreibung der Benutzeroberfläche spezifiziert werden (*abstrakte Interaktionsobjekte*), welche dann im weiteren Verlauf der Entwicklung an die konkreten Möglichkeiten des Zielsystems angepasst werden können (*konkrete Interaktionsobjekte*). Die abstrakte Beschreibung kann in vielen Fällen automatisch in eine konkrete Beschreibung umgewandelt werden. Hierzu kann die abstrakte Beschreibung entweder kompiliert oder interpretiert werden, was entsprechend der Vorgehensweise Auswirkungen auf deren Anpassbarkeit zur Laufzeit einer Anwendung besitzt [SWT05]. Beispiele für abstrakte Beschreibungssprachen sind die *User Interface Markup Language* (*UIML*) [AH04] und die *XML User Interface Language* (*XUL*) [GHHW01].

Modellbasierte Ansätze verfügen über eine weitere Abstraktionsebene, welche die Entwicklung von Bedienoberflächen vereinfachen soll [Pat00]. Hierbei können die Modellierer einer Anwendung zunächst ein (graphisches) Modell über die Benutzer und ihre Aufgaben erstellen, wobei sie durch ein (graphisches) Werkzeug unterstützt werden können. Für die Entwicklung von Bedienoberflächen werden hierbei (abhängig vom konkreten Ansatz) insbesondere die Elemente *Aufgabe* (*Task*), *Dialoge*, *Präsentationen* und *Benutzer* als

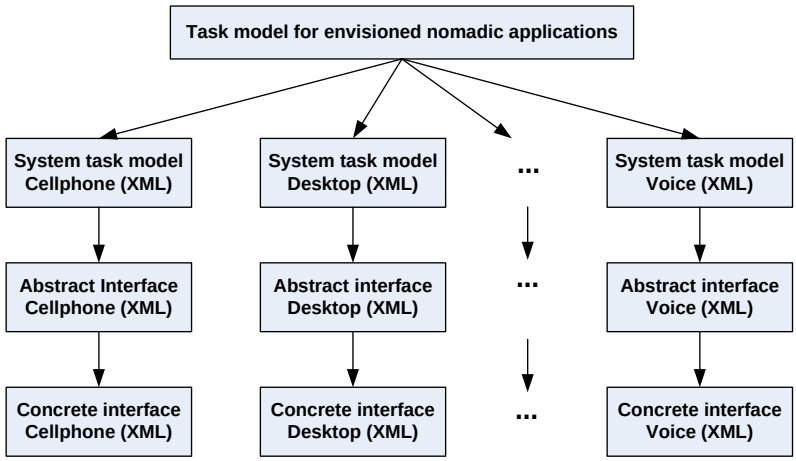


Abbildung 5.28: Transformation eines abstrakten CTT-Modells in verschiedene konkrete Benutzungsschnittstellen (MPS04)

besonders relevant angesehen [Pat00, Vil08]. Auf Basis eines solchen Modells der Beziehungen dieser Entitäten untereinander erfolgt dann in einem zweiten Schritt die Abbildung auf abstrakte Interaktionsobjekte und in Folge die Transformation auf konkrete Interaktionsobjekte [SWT05].

Ein Beispiel für einen modellbasierten Ansatz zur Beschreibung von Bedienoberflächen ist das *ConcurTaskTree*(*CTT*)-Modell [Pat00], welches die Beschreibung von Benutzeraufgaben durch hierarchische Dekomposition in Form eines Baumes erlaubt. Der Wurzelknoten des Baumes beschreibt dazu die auszuführende Aufgabe auf der höchsten Abstraktionsebene (z. B. die Durchführung einer Bestätigung). Die Knoten der Ebene n stellen dabei schrittweise Verfeinerungen der Aufgabe auf der Ebene $n-1$ dar, so dass die Blätter der Baumstruktur schließlich die Interaktionselemente repräsentieren, welche notwendig sind, um die übergeordnete Aufgabe zu erfüllen (z. B. die Präsentation des zu bestätigenden Sachverhalts und die Entgegennahme der Eingabedaten des Benutzers). Auf dieser Basis lassen sich verschiedenartige Benutzeroberflächen für unterschiedliche Plattformen beschreiben und auch durch multimodale Interaktionsmöglichkeiten unterstützen (vgl. Abbildung 5.28). Die Transformation kann dabei semiautomatisch erfolgen, so dass auf jeder Transformationsebene eine Schnittstelle für den Anwendungsentwickler zur Verfügung steht (vgl. [MPS04]). Die Übertragung von CTT-basierten Verfahren für die Flexibilisierung verteilt ausgeführter Prozesse wird in Abschnitt 6.4 genauer vorgestellt.

5.9 Ergebnis der Flexibilitätsbetrachtung

Für eine Beurteilung der Flexibilität verteilt ausgeführter Prozesse wurden in diesem Kapitel allgemeine und aktuelle Grundlagen zur Interoperabilisierung der Prozessausführung, verschiedene Arten von Ansätzen zur Verteilung, Ansätze zur Überwachung und Steuerung verteilt ausgeführter Pro-

zesse und Ansätze zur Anpassung von Benutzungsschnittstellen untersucht. Das Hauptaugenmerk der hier durchgeführten Flexibilitätsbetrachtung liegt dabei auf der Beurteilung der bestehenden Verteilungsansätze. Diese wurden nach ihrem übergeordneten Verteilungsmodell in allgemeine Choreographien, Ansätze zur Partitionierung von Prozessen, Ansätze zur Migration von Prozessinstanzen und vollverteilte Ansätze klassifiziert, wobei auch hybride Ansätze mit Einflüssen mehrerer Verteilungsmodelle betrachtet wurden (vgl. Abschnitte 5.3 bis 5.6). Tabelle 5.1 zeigt eine entsprechende Übersicht der betrachteten Ansätze zur Verteilung der Prozessausführung mit der Zuordnung der in den Abschnitten 4.3.1 und 4.3.2 identifizierten Ziele und Eigenschaften. Tabelle 5.2 fasst die Bewertung dieser Ansätze in Hinblick auf die in Abschnitt 4.4.2 erarbeiteten Anforderungen zusammen.

Innerhalb der Gruppierung des zugrunde liegenden Verteilungsmodells besitzen die betrachteten Ansätze relativ ähnliche Eigenschaften. So bedingen Ansätze, bei denen eine *(physische) horizontale Partitionierung* (vgl. Abschnitt 4.3.3) des Prozessmodells stattfindet, in der Regel eine Verteilung der Partitionen zur Entwicklungszeit oder spätestens vor dem Ausführungsbeginn des Prozesses. Die Verteilung ist dementsprechend weitestgehend auf das Prozessmodell festgelegt, so dass für verschiedene Prozessinstanzen keine unterschiedliche Verteilung erzielt werden kann bzw. nur unter erheblichem Aufwand realisiert werden könnte. Eine dynamische (Um)Verteilung wird daher auch von keinem der betrachteten Ansätze mit einer horizontalen Partitionierung unterstützt. Dies gilt insbesondere für die Choreographien im Allgemeinen, für welche in der Regel passende Schnittstellen für die Interprozesskommunikation manuell eingerichtet werden müssen. Die Partitionierungsansätze von VAN DER AALST, BARESI et al., KHALAF et al. und WUTKE et al. sowie die Ansätze *MENTOR* und *CiAN* unterstützen zwar eine teilweise automatisierte Erstellung der Prozesspartitionen, allerdings sind die hierfür vorgesehenen Vorgehensweisen eher auf eine längerfristig gleichartige Nutzung der resultierenden Strukturen ausgelegt, z. B. im Kontext der genannten Produktionsprozesse (vgl. auch Abschnitt 2.2). Aufgrund der Komplexität, der Dauer der Anpassung und der oftmals erforderlichen technischen und organisatorischen Abstimmung zwischen den beteiligten Parteien ist der Anpassungsvorgang bei diesen Ansätzen daher für eine individuelle Anwendung auf einzelne Prozessinstanzen zu aufwendig. Zudem wird hierbei nicht davon ausgegangen, dass Prozesse bereits instantiiert wurden, bevor eine Anpassung der Verteilung notwendig ist. Diese Eigenschaften spiegeln sich auch in der eher negativen Bewertung dieser Ansätze in Tabelle 5.2 in Hinblick auf die Anforderungen $D_{1.1}$ bis $D_{1.3}$ (Umgang mit Unvorhersehbarkeit hinsichtlich Verteilungszeitpunkten) sowie Anforderung $D_{3.1}$ (Individuelle Verteilung für Prozessinstanzen) wider. Entsprechend werden bei diesen Ansätzen natürlich auch keine Vorgaben des Prozessinitiators für die Verteilung des Prozesses berücksichtigt (vgl. Anforderung $D_{10.2}$), während etwaige Vorgaben des Prozessmodellierers direkt in die Modellierung des Prozessmodells einfließen können (vgl. Anforderung $D_{10.1}$). Dieses bedeutet jedoch in den meisten Fällen eine Vermischung

Ansatz		Verteilungsziel	Eigenschaften der Verteilung								
			Granularität	Abstraktionsebene	Zielort	Infrastrukturen	Koordination	Ausführung von Partitionen	Synchronisation	Zeitpunkt	Durchführung
Choreo-graphie	Choreographie (statisch)	Kooperation	beliebig	Prozessmodell	organisations-übergreifend	i.d.R. stationär	dezentral	sequentiell und parallel	i.d.R. unmittelbar	Entwicklungszeit	i.d.R. manuell
	Choreographie (dynamisch)	Kooperation	beliebig	Prozessmodell	organisations-übergreifend	i.d.R. stationär	dezentral	sequentiell und parallel	i.d.R. unmittelbar	Entwicklungszeit	i.d.R. manuell
(physische) Partitionierung	Exotica/FMDC [AGK+95, AGK+96]	Mobilität	einzelne Aktivität	Prozessinstanz	organisations-intern	hybrid	zentral	sequentiell und parallel	unmittelbar	Laufzeit	unterstützt
	MENTOR [WWWD96, WW97, MWW+98]	Kooperation	beliebig	Prozessmodell	organisations-intern	stationär	Partitionierung: zentral; Ausführung: dezentral	sequentiell und parallel	unmittelbar oder verzögert	Entwicklungszeit	unterstützt
	Ansatz von VAN DER AALST [Aal99]	Kooperation	beliebig	Prozessmodell	organisations-übergreifend	stationär	Partitionierung: zentral; Ausführung: dezentral	sequentiell und parallel	k.A.	Entwicklungszeit	unterstützt
	MOBILE [Sch01,JSH+01]	Mobilität	beliebig	Prozessinstanz	organisations-intern	hybrid	zentral	sequentiell und parallel	unmittelbar oder verzögert	Partitionierung: Instantiierung; Zuweisung: Laufzeit	unterstützt
	Ansatz von BARESI, MAURINO und MODAFFERI [BMM04, MM05b, MM05a, BMM06]	Mobilität	beliebig	Prozessmodell	organisations-intern	hybrid	dezentral	sequentiell oder parallel	unmittelbar	Deployment oder Laufzeit	automatisch
	MobiWork [HSH+06, SHH+07]	Mobilität	beliebig	Prozessinstanz	organisations-intern	mobil	Partitionierung: zentral; Ausführung: dezentral	sequentiell oder parallel	k.A.	Partitionierung: Instantiierung; lokale Umverteilung: Laufzeit	automatisch
	CIAN [SRF06, SRG08]	Mobilität	beliebig	Prozessmodell	organisations-intern	mobil	Partitionierung: zentral; Ausführung: dezentral	sequentiell oder parallel	unmittelbar	Interne Prozesse: Entwicklungszeit; Öffentliche Prozesse: Deployment	unterstützt
	Ansatz von KHALAF und LEYMANN [KL06, Kha08]	Kooperation	beliebig	Prozessmodell	organisations-übergreifend	stationär	Partitionierung: zentral; Ausführung: dezentral	sequentiell oder parallel	unmittelbar	Deployment	automatisch
	Ansatz von WUTKE, MARTIN und LEYMANN [MWL08, WML09a,WML09b, Wut10]	Kooperation, Qualitäts- und Effizienzsteigerung	beliebig	Prozessmodell	organisations-übergreifend	stationär	Partitionierung: zentral; Ausführung: dezentral	sequentiell oder parallel	unmittelbar	Deployment	automatisch
Migration von Prozessinstanzen	Ansatz von CICHOCKI und RUSINKIEWICZ [CR97, CRW98, CR04]	Abbildung von natürlich verteilten Abläufen	beliebig	Prozessinstanz	k.A.	hybrid	dezentral	sequentiell oder parallel	unmittelbar oder verzögert	Laufzeit	automatisch
	Ansatz von VAN DER AALST [Aal99]	Qualitäts- und Effizienzsteigerung	beliebig	Prozessinstanz	k.A.	stationär	zentral	sequentiell	-	Replikation des Prozessmodells: Deployment; Verteilung der Prozessinstanzen: Laufzeit	automatisch
	ADEPT Distribution [BD00, BRD01]	Qualitäts- und Effizienzsteigerung	beliebig	Prozessinstanz	k.A.	stationär	Partitionierung: zentral; Ausführung: dezentral	sequentiell und parallel	unmittelbar	Partitionierung: Deployment; Zuweisung: Laufzeit	automatisch
	OSIRIS [SWSS03,SWSS04, Sch04b, Sch05b]	Dezentralität	beliebig	Prozessinstanz	k.A.	hybrid	dezentral	sequentiell und parallel	unmittelbar	Partitionierung: Deployment; Zuweisung: Laufzeit	automatisch
	Ansatz von KUNZE et al. [Kun05, KZL06, KZL07b, Kun08]	Zugriff auf lokale Ressourcen	beliebig	Prozessinstanz	organisations-übergreifend	hybrid	dezentral	sequentiell und parallel	k. A.	Laufzeit	automatisch
Vollverteilte Prozessausführung	Exotica/FMQM [AAA+95]	Dezentralität	minimale Partitionen	Prozessmodell	k.A.	stationär	Partitionierung: zentral, Ausführung: dezentral	sequentiell und parallel	unmittelbar	Deployment	unterstützt
	Ansatz von ATLURI et al. [ACMM07]	Dezentralität	minimale Partitionen	Prozessinstanz	organisations-übergreifend	stationär	dezentral	sequentiell und parallel	k.A.	Laufzeit	automatisch
	Ansatz von MONTAGUT und MOLVA [MM05c]	Mobilität, Dezentralität	minimale Partitionen	Prozessinstanz	k.A.	mobil	dezentral	sequentiell und parallel	k.A.	Laufzeit	unterstützt
	Ansatz von YU und YANG [YY07]	Dezentralität	minimale Partitionen	Prozessinstanz	k.A.	stationär	dezentral	sequentiell und parallel	k.A.	Laufzeit	automatisch

Tabelle 5.1: Ergebnis der Analyse verschiedener Verteilungsansätze in Bezug auf die Ziele und Eigenschaften dynamisch verteilt ausgeführter Prozesse

von technischen und fachlichen Beschreibungen oder erfordert andere technische Anpassungsmaßnahmen, wie z. B. eine Transformation des Prozessmodells in eine verteilt ausführbare Form (vgl. Anforderungen $D_{12.1}$ und $D_{12.3}$), was oft die generelle Anpassbarkeit auf der Ebene des fachlichen Prozessmodells oder die Nutzung bestehender Prozessmanagementsysteme erschwert.

Ergänzend hierzu berücksichtigen die Partitionierungsansätze für mobile oder hybride Infrastrukturen, im Speziellen *Exotica/FMDC*, *MOBILE* und *MobiWork*, auch die verteilte Ausführung von einzelnen Prozessinstanzen. Der Verteilungsansatz von *Exotica/FMDC* verfügt hierbei zwar über die beste Bewertung in Hinblick auf den Umgang mit Unvorhersehbarkeit hinsichtlich Verteilungszeitpunkten (Anforderungen $D_{1.1}$ bis $D_{1.3}$) und der individuellen Verteilung einzelner Prozessinstanzen (Anforderung $D_{3.1}$), unterstützt jedoch im Gegensatz zu den anderen Ansätzen mit der Auslagerung von nur einzelnen Aktivitäten eine sehr eingeschränkte Granularität. Der Ansatz von *MOBILE* erlaubt durch seine dreistufige Anpassung der Zuordnung von Prozesspartitionen zu Prozessteilnehmern eine dynamische Umverteilung, jedoch nur in Hinblick auf die Zuweisung bereits vor der Laufzeit determinierter Prozesspartitionen. Bei *MobiWork* werden die Prozesspartitionen ebenfalls vor der Laufzeit des Prozesses festgelegt und nur die lokal zugeordneten Partitionen können später individuell verschoben werden. Alle drei Ansätze ermöglichen daher eine dynamische Zuweisung von Ausführungseinheiten zur Laufzeit, nicht jedoch eine dynamische Partitionierung des Prozesses im Sinne dieser Arbeit.

Im Gegensatz zu den Ansätzen zur Verteilung durch Fragmentierung zeichnet sich eine *vertikale Partitionierung* (vgl. Abschnitt 4.3.3) naturgemäß durch eine höhere Individualität bei der verteilten Ausführung von Prozessinstanzen aus. Eine rein vertikale Partitionierung des Prozesses lässt jedoch nur die Zuordnung einer ganzen Prozessinstanz an eine bestimmte Ausführungseinheit zu, was den Ansprüchen dieser Arbeit in Hinblick auf die zu unterstützende beliebige Granularität der Verteilung nicht gerecht wird. Besser geeignet sind Ansätze, bei denen im Rahmen einer *logischen Verteilung* die Prozessinstanz von einer Ausführungseinheit zur nächsten migriert werden kann. Die Ansätze von CICHOCKI et al., VAN DER AALST, KUNZE et al. sowie *ADEPT Distribution* und *OSIRIS* unterstützen die genannten Eigenschaften (vgl. Tabelle 5.1), wobei nur die Ansätze von CICHOCKI et al. und KUNZE et al. auch die Zerlegung des Prozesses zur Laufzeit ermöglichen. Der Ansatz von VAN DER AALST unterstützt zwar ebenfalls den Umgang mit Unvorhersehbarkeit hinsichtlich des Verteilungszeitpunkts, ist aber hierbei von einer Vorverteilung des Prozessmodells vor der Instantiierung des Prozesses und einer einschränkenden zentralen Infrastruktur abhängig. Ebenso werden die Ansätze *ADEPT Distribution* und *OSIRIS* durch eine vorgelagerte Phase zur Partitionierung des Prozesses in ihrer Flexibilität eingeschränkt.

Die betrachteten Ansätze zur *vollverteilten Prozessausführung* sind prinzipiell durch ein sehr hohes Maß an Unterstützung für dynamische Umgebungen und dezentrale Architekturen gekennzeichnet. Hierbei ist jedoch die Granula-

rität der Verteilung auf die Bildung minimaler Partitionen festgelegt, was insbesondere den Kommunikationsaufwand zwischen den (potentiell) beteiligten Parteien unverhältnismäßig erhöht. Die betrachteten Ansätze zur Begegnung dieser Probleme sind wiederum durch Einschränkungen hinsichtlich ihrer Flexibilität gekennzeichnet, zum Beispiel durch eine statische (Vor-)Verteilung im Fall von *Exotica/FMQM* oder durch die Integration privater interner Prozessabschnitte wie im Ansatz von Montagut und Molva. Die Ansätze von Atluri et al. sowie von Yu und Yang stellen schließlich Spezialfälle einer Migration von Prozessinstanzen mit minimalen (logischen) Partitionen dar und teilen daher deren bereits oben beschriebene Eigenschaften, sind jedoch prinzipiell schlechter zu bewerten, da sie hinsichtlich der Granularität auf die Bildung minimaler Partitionen eingeschränkt sind.

Tabelle 5.2 zeigt eine Zusammenfassung der genannten Ergebnisse. Die gleichzeitige Migration von Prozessmodell und laufender Prozessinstanz stellt hierbei den einzigen bekannten Ansatz dar, um *fortwährend* dynamische Anpassungen zu erlauben und ist somit besonders gut für Umgebungen geeignet, welche besonders häufig von unvorhersehbaren Änderungen und Ereignissen betroffen sind. In vielen Fällen ist die Flexibilität dieser Vorgehensweise jedoch von der genauen Umsetzung des Verfahrens und der Entkopplung von inhaltlichen Prozessbeschreibungen und technischen Verwaltungsinformationen bedingt. Die Einführung neuer oder erweiterter Beschreibungssprachen oder die Transformation von Prozessen zur Laufzeit machen neue Werkzeuge für die Modellierung und/oder für die Ausführung der Prozesse erforderlich (vgl. Anforderungen $D_{12.1}$ und $D_{12.3}$). Um eine flexible Entscheidung über die Verteilung des Prozesses zu ermöglichen, müssten somit alle bestehenden Prozessmodelle zunächst in eine solche Sprache übersetzt werden, bevor eine Anpassung der Prozessausführung (ggf. auch nur für einzelne Instanzen) überhaupt möglich wäre. Diese Umstellung würde einen hohen Änderungsaufwand mit sich bringen, wobei auch hohe Kosten (z. B. für die Anschaffung und Wartung von Software sowie für die Schulung von Mitarbeitern) zu erwarten sind. Zudem stellen die aktuell standardisierten Prozessbeschreibungssprachen (vgl. Abschnitt 5.2) für organisationsübergreifende Kooperationen bereits einen großen Schritt zur Interoperabilisierung des Prozessmanagements und zur besseren Kommunikation über fachliche Prozesse dar. Eine zusätzliche Beschreibung organisatorischer oder technischer Aspekte sollte daher auch hier – nach dem Vorbild des klassischen nicht-verteilten Prozessmanagements – nicht direkt mit der Beschreibung des inhaltlichen Prozessablaufs vermischt werden.

Mit umfassenden Möglichkeiten sowohl zur technischen als auch zur inhaltlichen Anpassung der Prozessausführung kommt vor allem dem Ansatz von CICHOCKI und RUSINKIEWICZ in Bezug auf die Flexibilisierung der verteilten Prozessausführung eine hohe Bedeutung zu. Die Kombination migrierender Prozessinstanzen mit explizit formulierten Regeln zur ereignisbasierten Anpassung des Prozesses und seiner Verteilung wirkt sich dabei sehr positiv auf die Anpassbarkeit von Prozessen und der ge-

Ansatz		Flexibilität							Überwachung u. Steuerung				(Mobile) Infrastrukturen u. Heterogenität						
		$D_{1.1}$	$D_{1.2}$	$D_{1.3}$	$D_{2.1}$	$D_{2.2}$	$D_{2.3}$	$D_{3.1}$	$D_{4.1}$	$D_{4.2}$	$D_{5.1}$	$D_{6.1}$	$D_{7.1}$	$D_{8.1}$	$D_{8.2}$	$D_{8.3}$	$D_{9.1}$	$D_{9.2}$	$D_{9.3}$
Choreographie	Choreographie (statisch)	-	-	-	+	-	-	-	Betrachtung komplementärer Ansätze in Abschnitt 5.7 und separate Auswertung in Tabelle 5.3				+	k.A.	k.A.	k.A.	+	+	Betrachtung komplementärer Ansätze in Abschnitt 5.8
	Choreographie (dynamisch)	-	-	-	+	+	+	-					+	k.A.	k.A.	k.A.	+	+	
(physische) Partitionierung	Exotica/FMDC [AGK+95, AGK+96]	+	+	+	-	+	-	+					-	+	(-)	+	k.A.	k.A.	
	MENTOR [WWWD96, WW97, MWW+98]	-	-	-	+	(+)	(+)	-					+	k.A.	k.A.	k.A.	k.A.	k.A.	
	Ansatz von VAN DER AALST [Aal99]	-	-	-	+	-	-	-					+	k.A.	k.A.	k.A.	+	+	
	MOBILE [Sch01,JSH+01]	(+)	-	-	+	+	+	+					(-)	+	+	+	(-)	(-)	
	Ansatz von BARESI, MAURINO und MODAFFERI [BMM04, MM05b, MM05a, BMM06]	(-)	(-)	(-)	+	+	(-)	(-)					+	+	+	k.A.	-	-	
	MobiWork [HSH+06, SHH+07]	-	-	-	+	(+)	(+)	(+)					+	+	+	k.A.	-	-	
	CiAN [SRF06, SRG08]	-	-	-	+	-	-	-					+	+	+	k.A.	-	-	
	Ansatz von KHALAF und LEYMANN [KL06, Kha08]	-	-	-	+	(-)	(-)	-					+	k.A.	k.A.	k.A.	+	+	
	Ansatz von WUTKE, MARTIN und LEYMANN [MWL08, WML09a,WML09b, Wut10]	-	-	-	+	(-)	(-)	-					+	k.A.	k.A.	k.A.	+	+	
Migration von Prozessinstanzen	Ansatz von CICHOCKI und RUSINKIEWICZ [CR97, CRW98, CR04]	+	(-)	+	+	+	+	+					+	+	+	+	k.A.	k.A.	
	Ansatz von VAN DER AALST [Aal99]	(+)	(+)	(+)	+	(+)	+	+					-	-	k.A.	k.A.	+	+	
	ADEPT Distribution [BD00, BRD01]	-	-	-	+	+	+	(-)					+	k.A.	k.A.	k.A.	k.A.	k.A.	
	OSIRIS [SWSS03,SWSS04, Sch04b, Sch05b]	-	-	+	-	+	+	(-)					+	-	k.A.	k.A.	k.A.	k.A.	
	Ansatz von KUNZE et al. [Kun05, KZL06, KZL07b, Kun08]	+	(-)	+	+	+	(+)	+					+	+	(+)	k.A.	+	k.A.	
Vollverteilte Prozessausführung	Exotica/FMQM [AAA+95]	-	-	-	-	(-)	-	(-)					(+)	k.A.	-	k.A.	k.A.	k.A.	
	Ansatz von ATLURI et al. [ACMM07]	(-)	(+)	+	-	+	-	(-)					(+)	+	-	k.A.	k.A.	k.A.	
	Ansatz von MONTAGUT und MOLVA [MM05c]	-	-	-	-	+	-	-					+	+	+	k.A.	+	+	
	Ansatz von YU und YANG [YY07]	(-)	(+)	+	-	+	-	(-)					+	+	-	k.A.	k.A.	k.A.	

Ansatz		Benutzerdefinierte Vorgaben, nicht-funktionale Aspekte und Rahmenbedingungen															
		$D_{10.1}$	$D_{10.2}$	$D_{10.3}$	$D_{11.1}$	$D_{11.2}$	$D_{11.3}$	$D_{12.1}$	$D_{12.2}$	$D_{12.3}$	$D_{13.1}$	$D_{13.2}$	$D_{14.1}$	$D_{15.1}$	$D_{16.1}$	$D_{16.2}$	$D_{16.3}$
Choreographie	Choreographie (statisch)	+	-	-	+	-	+	-	Betrachtung komplementärer Ansätze in Abschnitt 5.7 und separate Auswertung in Tabelle 5.3	+	-	-	-	-	Betrachtung komplementärer Ansätze in Abschnitt 5.7 und separate Auswertung in Tabelle 5.3		-
	Choreographie (dynamisch)	+	-	-	+	-	+	-		+	+	+	-	-			-
(physische) Partitionierung	Exotica/FMDC [AGK+95, AGK+96]	-	-	+	+	+	-	+		+	+	+	+	k.A.			(+)
	MENTOR [WWWD96, WW97, MWW+98]	(+)	-	-	k.A.	+	+	(+)		-	(+)	(+)	(-)	k.A.			(+)
	Ansatz von VAN DER AALST [Aal99]	+	-	-	+	-	+	+		+	-	-	-	-			-
	MOBILE [Sch01,JSH+01]	+	+	+	+	+	+	(-)		(-)	k.A.	k.A.	k.A.	k.A.			+
	Ansatz von BARESI, MAURINO und MODAFFERI [BMM04, MM05b, MM05a, BMM06]	+	k.A.	k.A.	k.A.	k.A.	+	-		-	k.A.	k.A.	(+)	k.A.			+
	MobiWork [HSH+06, SHH+07]	k.A.	(+)	(+)	k.A.	+	+	k.A.		k.A.	-	-	(+)	k.A.			+
	CiAN [SRF06, SRG08]	k.A.	-	-	k.A.	+	+	-		k.A.	-	-	(-)	k.A.			(+)
	Ansatz von KHALAF und LEYMANN [KL06, Kha08]	+	-	-	+	k.A.	+	+		-	k.A.	k.A.	+	k.A.			(+)
	Ansatz von WUTKE, MARTIN und LEYMANN [MWL08, WML09a,WML09b, Wut10]	+	-	-	+	k.A.	+	+		-	k.A.	k.A.	+	k.A.			(+)
Migration von Prozessinstanzen	Ansatz von CICHOCKI und RUSINKIEWICZ [CR97, CRW98, CR04]	+	-	-	k.A.	-	+	-		-	+	+	-	k.A.			+
	Ansatz von VAN DER AALST [Aal99]	+	(+)	(+)	+	+	-	+		+	-	(-)	(-)	+			+
	ADEPT Distribution [BD00, BRD01]	+	(+)	-	k.A.	+	+	-		-	+	+	(-)	+			+
	OSIRIS [SWSS03,SWSS04, Sch04b, Sch05b]	(-)	(-)	(-)	k.A.	-	+	-		-	(+)	+	-	-			+
	Ansatz von KUNZE et al. [Kun05, KZL06, KZL07b, Kun08]	+	-	-	k.A.	(+)	+	-		k.A.	+	+	(+)	k.A.			+
Vollverteilte Prozessausführung	Exotica/FMQM [AAA+95]	-	-	-	k.A.	-	+	-		-	(-)	(-)	-	k.A.			+
	Ansatz von ATLURI et al. [ACMM07]	k.A.	k.A.	k.A.	k.A.	-	+	+		-	+	+	+	k.A.			+
	Ansatz von MONTAGUT und MOLVA [MM05c]	+	-	-	+	-	+	-		+	(+)	(+)	-	-			-
	Ansatz von YU und YANG [YY07]	k.A.	k.A.	k.A.	k.A.	-	+	+		-	+	+	+	k.A.			+

"+" = Kriterium voll erfüllt "(+)" = Kriterium unter geringen Einschränkungen erfüllbar "(-)" = Kriterium nur unter erheblichen Einschränkungen erfüllbar "-" = Kriterium nicht erfüllt "k.A." = keine Angabe bzw. Kriterium nicht anwendbar

Tabelle 5.2: Ergebnis der Analyse verschiedener Verteilungsansätze in Bezug auf die Anforderungen dynamisch verteilt ausgeführter Prozesse

samten Ausführungsumgebung aus. Durch die Möglichkeit zur Weiterleitung des Prozesses an andere Ausführungseinheiten können sowohl Fehler (wie z. B. die fehlende Verfügbarkeit von Ressourcen) behandelt als auch nicht-funktionale Aspekte (z. B. durch Lastverteilung oder Reduktion zu übertragender Datenmengen) optimiert werden. Die dezentrale Ausführung vermeidet Engpässe und kann somit zu einer schnelleren und zuverlässigeren Ausführung beitragen. Die Möglichkeiten zur dynamischen Einbindung weiterer Ausführungseinheiten hat zudem positive Auswirkungen auf die Skalierbarkeit des gesamten (verteilten) Systems. Der Transfer der ganzen Prozessinstanz mit allen dazugehörigen Daten verspricht zudem eine gute Anwendbarkeit des Ansatzes auf die Integration von mobilen Geräten, welche zeitweilig nicht miteinander über eine stabile Netzwerkverbindung verbunden sind. Dies zeigt auch der Ansatz von KUNZE et al., welcher speziell auf diese Problematik ausgerichtet ist. Schwierigkeiten bestehen laut CICHOCKI und RUSINKIEWICZ jedoch vor allem in der Administration und Überwachung der Prozessausführung sowie in der Bereitstellung von geeigneten Sicherheitsmaßnahmen zum Schutz der Prozessinstanzen und der teilnehmenden Ausführungsumgebungen [CRW98]. Zum einen gewährt die Migration einer ganzen Prozessinstanz jeder Ausführungseinheit Einblick in die ggf. vertrauliche fachliche Logik und die Daten des Prozesses. Zudem können Ausführungseinheiten den Prozess während der Laufzeit unerwünscht manipulieren. Zum anderen sollte auch der Zugriff auf die internen Ressourcen einer Ausführungsumgebung durch einen fremden Prozess restriktiert werden können. Neben der unerwünschten Auslastung der Ressourcen durch einen fremden Prozess steht hier vor allem der Zugriff auf sensible interne Funktionen oder Daten ein größeres Sicherheitsproblem dar.

Die Anforderungen zur Anpassung an heterogene Infrastrukturen ($D_{9.1}$ bis $D_{9.3}$) werden in den meisten der betrachteten Ansätze nicht explizit adressiert. In vielen Fällen ergeben sich hinsichtlich des Aufrufs von Ressourcen und dem Austausch von Daten jedoch implizit vielfältige Möglichkeiten aus der Nutzung dienstorientierter Konzepte, wobei prinzipiell auch dynamisch über (globale oder lokale) Verzeichnisdienste geeignete Softwarekomponenten zur Ausführung von automatisch auszuführenden Aktivitäten des Prozesses aufgefunden und eingebunden werden können. Wie der unausgefüllte Bereich zu Anforderung $D_{9.3}$ in Tabelle 5.2 zeigt, findet jedoch bei keinem der betrachteten Ansätze eine Berücksichtigung von heterogenen Benutzungsschnittstellen bei einer dynamisch verteilten Prozessausführung statt. Bestehende Ansätze für eine entsprechende Anpassung der Benutzungsschnittstellen erfordern entweder eine zentralisierte Infrastruktur und eine dauerhafte Konnektivität, um eine geeignete Repräsentation der Schnittstelle zur Laufzeit ableiten zu können, oder setzen auf die Einbettung standardisierter Beschreibungen, welche zur Laufzeit nicht oder nur eingeschränkt an den aktuellen Kontext des Benutzers angepasst werden können. In Abschnitt 5.8 wurde jedoch auch festgestellt, dass vielfältige Ansätze aus dem Bereich der kontextbasierten Anpassung von Benutzungsschnittstellen über Dienstschnittstellen und/oder ab-

strakte Beschreibungen von Benutzeroberflächen ein großes Flexibilisierungspotential bieten. Eine Übertragung dieser Konzepte wird daher unter anderem auch Inhalt des nächsten Kapitels dieser Arbeit sein.

Wie in Kapitel 3 gezeigt wurde, stellt schließlich insbesondere das Erkennen von Handlungsbedarf und das (ggf. automatisierte) Ableiten von Anpassungsmaßnahmen eine wichtige Voraussetzung für die Anpassung von Prozessen dar. Wie auch Tabelle 5.2 durch die markierten Bereiche deutlich zeigt, gehen die betrachteten Ansätze zur Verteilung der Prozessausführung jedoch kaum auf die Erhebung und Verarbeitung der für die Ableitung von Anpassungen relevanten Informationen ein. Um auch die Anforderungen $D_{4.1}$ bis $D_{6.1}$ zur zeitnahen Bereitstellung und Erhebung von Informationen durch kontinuierliche Überwachung bzw. individuelle Abfrage, zur Vorhersage zukünftiger Entwicklungen sowie zur Steuerung der Prozessausführung zu evaluieren, die Invasivität solcher Überwachungsmaßnahmen in Bezug auf die Prozessmodellierung zu überprüfen (vgl. Anforderung $D_{12.2}$) und die Möglichkeit einer Automatisierbarkeit des gesamten Anpassungsvorgangs einzuschätzen (vgl. Anforderungen $D_{16.1}$ bis $D_{16.2}$), wurden zusätzlich eine Reihe spezieller Ansätze zur Überwachung und Steuerung verteilt ausgeführter Prozesse betrachtet (vgl. Abschnitt 5.7).

Tabelle 5.3 zeigt das Ergebnis der vorgenannten Untersuchung als Ergänzung zu den entsprechend gekennzeichneten Bereichen von Tabelle 5.2. Viele der betrachteten Ansätze verfügen dabei nicht über eine angemessen zeitnahe Bereitstellung bzw. Erhebung der benötigten Informationen (vgl. Anforderungen $D_{4.1}$ und $D_{4.2}$). Dies liegt zum einen daran, dass die für die Ableitung von Anpassungen individuell *relevanten* Informationen gar nicht zugegriffen oder ausgetauscht werden können, da diese nicht zum extern beobachtbaren Verhalten zählen oder durch Annotation des Prozessmodells erlangt werden können. Dies betrifft insbesondere das Monitoring des Nachrichtenaustausches, die Sammlung und den Austausch von Erfahrungswerten sowie den Ansatz von BARESI, GHEZZI und GUINEA. Die Ansätze von LUDWIG, DAN und KEARNEY, das Monitoring abstrakter Sichten sowie der Ansatz von WETZSTEIN bedienen sich einer vorgelagerten Verhandlungsphase, um die relevanten Informationen zugänglich machen zu können. Der Ansatz von LUDWIG, DAN und KEARNEY fokussiert jedoch stark auf der Verhandlung und das Festschreiben der Vereinbarungen, während die technische Realisierung des Monitorings kaum Berücksichtigung findet. Der Ansatz von WETZSTEIN kann hinsichtlich einer kontinuierlichen Überwachung interessanter Aspekte durch eine ereignisbasierte Architektur zeitnah über die relevanten Daten verfügen und diese durch Complex Event Processing auch automatisiert verarbeiten, stellt jedoch keine Konzepte für den Ad-hoc-Zugriff von Informationen nach dem Pull-Modell bereit. Die fast bei allen Ansätzen fehlenden Steuerungsmöglichkeiten für ein aktives Eingreifen in die Prozessausführung wurden durch die Lösungsvorschläge von MOSER et al. und VAN LESSEN et al. aufgegriffen. Beide Ansätze stellen ihre Management-Funktionalitäten jedoch

	Kontinuierliche Überwachung (Push-Modell)	Individuelle Abfrage im Bedarfsfall (Pull-Modell)	Vorhersage zukünftiger Entwicklungen	Bereitstellung von Management-funktionalitäten	Trennung fachl. und techn. Logik	Automatische Erhebung und Verarbeitung	Automatische Ableitung von Anpassungen
	$D_{4.1}$	$D_{4.2}$	$D_{5.1}$	$D_{6.1}$	$D_{12.2}$	$D_{16.1}$	$D_{16.2}$
Monitoring des Nachrichtenaustausches	(-)	-	-	-	(+)	-	-
Sammlung und Austausch von Erfahrungswerten [KWA99b] und [JF05, JBF07, JFB07]	-	(-)	(+)	-	+	(-)	-
Ansatz von BARESI, GHEZZI und GUINEA [BGG04, BG05]	(-)	(-)	-	(-)	-	+	-
Ansatz von LUDWIG, DAN und KEARNEY [LDK04]	(+)	(+)	k.A.	-	+	k.A.	k.A.
Monitoring abstrakter Sichten von Prozessen [CCSD04]	(+)	(+)	-	-	(+)	-	k.A.
Peer-to-Peer-basiertes Monitoring [BKK+04, KW05, WW08]	(-)	(-)	k.A.	-	+	(+)	-
Ansatz von WETZSTEIN et al. [WKK+10]	+	-	(+)	-	+	+	-
Ansatz von MOSER, ROSENBERG und DUSTDAR [MRD08]	(-)	-	-	(-)	+	(+)	(+)
Ansatz von VAN LESSEN et al. [vLLM+08]	(+)	(+)	-	(+)	+	k.A.	k.A.

"+" = Kriterium voll erfüllt
"-" = Kriterium nicht erfüllt
"k.A." = keine Angabe bzw. Kriterium nicht anwendbar
"(+)" = Kriterium unter geringen Einschränkungen erfüllbar
"(-)" = Kriterium nur unter erheblichen Einschränkungen erfüllbar

Tabelle 5.3: Ergebnis der Analyse verschiedener Ansätze zur Überwachung und Steuerung der (verteilten) Prozessausführung in Bezug auf die Anforderungen dynamisch verteilt ausgeführter Prozesse

nur lokal bereit und müssten (mehr oder weniger aufwendig) für eine organisationsübergreifenden Zugriff angepasst werden.

Als Konsequenz der geringen Erfüllung der Anforderungen $D_{4.1}$ und $D_{4.2}$ können auch die anderen Anforderungen des ergänzenden Kriterienkataloges nur sehr eingeschränkt unterstützt werden. Insbesondere eine geeignete automatische Ableitung von Anpassungsbedarf (vgl. Anforderungen $D_{16.1}$ bis $D_{16.2}$ sowie die proaktive Anpassung durch Vorhersage zukünftiger Entwicklungen werden bisher wenig unterstützt. Für die Flexibilisierung verteilter Prozessausführung ist daher auch in diesem Gebiet eine Erarbeitung geeigneter Lösungsvorschläge erforderlich.

5.10 Zusammenfassung

In diesem Kapitel wurde untersucht, ob sich bestehende Ansätze für die verteilte Ausführung von Prozessen auch für die in Kapitel 4 vorgestellten Prozesse mit besonders hohen Anforderungen an eine dynamische Verteilung nutzbar machen lassen. Dabei kann festgehalten werden, dass die Flexibilität prozessorientierter Anwendungen in technischer Hinsicht durch eine Verteilung der Prozessausführung prinzipiell erhöht werden kann, jedoch bei den meisten der betrachteten Ansätze hieraus auch wieder weitestgehend starre Strukturen resultieren, welche die *fortwährende Anpassbarkeit* der Verteilung zur Laufzeit des Prozesses einschränken. Zusammengefasst können die folgenden wichtigen Erkenntnisse angeleitet werden:

- Die strenge Kapselung von mehreren internen Prozessschritten hinter einer öffentlichen Schnittstelle erschwert den Austausch von Informa-

tionen über die Prozessausführung und verhindert eine veränderliche Partitionierung des Prozesses. Für die Ausführung verschiedener Prozessmodelle oder die Unterstützung verschiedener Partitionierungsvarianten müssen hierbei jeweils unterschiedliche Schnittstellen vorgehalten werden. Dies ist nur möglich, wenn alle auszuführenden Prozessmodelle und Partitionierungsvarianten im Voraus bekannt sind, was für die meisten der genannten Fallbeispiele in Abschnitt 4.2 nicht zutrifft.

- Eine (horizontale) physische Partitionierung des Prozessmodells erfordert in der Regel eine zentrale Planungsphase mit einer Verteilung der fragmentierten Prozesspartitionen vor der ersten Instantiierung des Prozesses. In einer dynamischen Umgebung kann jedoch nicht davon ausgegangen werden, dass die Menge der zur Laufzeit verfügbaren Ausführungseinheiten mit der Menge der in der Planungsphase verfügbaren Ausführungseinheiten übereinstimmt. Dies gilt insbesondere für mobile Ausführungseinheiten. Eine dezentrale Umverteilung der Partitionen während der Laufzeit ist hierbei zudem nur eingeschränkt auf Basis der bestehenden Prozesspartitionen oder durch den dynamischen Austausch von Ausführungseinheiten möglich.

- Eine (vertikale) Partitionierung von Prozessinstanzen mit der Möglichkeit zu einem Wechsel von Ausführungseinheiten erlaubt die individuelle Verteilung von Prozessinstanzen zur Laufzeit. Voraussetzung für eine dynamische Zuweisung von beliebigen Prozesspartitionen zu beliebigen (kompatiblen) Ausführungseinheiten ist jedoch, dass Prozessmodell und Instanzdaten den (potentiell) zu beteiligenden Ausführungseinheiten spätestens zur Ausführungszeit vorliegen. Für eine flexible Verteilung ist es hierbei ebenfalls wesentlich, dass das Prozessmodell nicht zur Entwicklungszeit physisch partitioniert und/oder an eine festgelegte Menge von Ausführungseinheiten distributiert wurde.

- Das Anreichern der technischen Repräsentation eines fachlichen Prozessmodells mit zusätzlichen (nicht-fachlichen) Aktivitäten zur Überwachung und/oder Verteilung führt zu einer ungewünschten Vermengung von fachlichen und technischen Anweisungen. Die daraus resultierende feste Partitionierung und Zuweisung zu Ausführungsumgebungen lässt sich in der Regel nur unter erheblichem Aufwand modifizieren und kann beim Anpassungsvorgang zu Verzögerungen führen. Eine fortwährende Anpassbarkeit der Prozesse auf ihrer Instanzebene sowohl in Hinblick auf die Verteilung als auch in Hinblick auf die dynamische Auswahl und Erhebung relevanter Informationen wird hierdurch erschwert.

- Die für eine verteilte Prozessausführung notwendige Kompatibilität wird aktuell durch eine zunehmende Verbreitung von standardisierten Prozessbeschreibungssprachen, gemeinsamen Protokollen und offenen Schnittstellen gefördert und sollte daher möglichst nicht durch neue

proprietäre Formate speziell zur Verteilung des Prozesses weiter eingeschränkt werden.

Die genannten Ergebnisse implizieren, dass für dynamisch verteilt ausgeführte Prozesse eine vertikale Verteilung von Prozessinstanzen einer horizontalen Verteilung von Prozessmodellen vorzuziehen ist und dass eine vorgeschaltete Verteilung des Prozessmodells oder eine physische Aufspaltung von Prozessen in beiden Fällen nicht vorteilhaft sind. Im folgenden Kapitel wird daher eine dynamische Verteilung von Prozessinstanzen auf Basis einer *logischen Partitionierung* vorgestellt, welche durch die Migration von Prozessinstanzen und das gleichzeitige Weiterreichen der zugehörigen Prozessmodelle inspiriert ist. Ziel dabei ist es, einen bestehenden Prozess unter Verwendung existierender Technologien verteilt auszuführen, ohne jedes Mal eine Anpassung des Prozessmodells oder der zugrunde liegenden organisatorischen oder technischen Infrastruktur vornehmen zu müssen. Dabei wird insbesondere auch die bislang noch nicht zufriedenstellende Integration eines solchen Verteilungsansatzes in dienstorientierte Architekturen berücksichtigt.

Der Verteilungsansatz wird durch einen entsprechend geeigneten Ansatz zur Überwachung und Steuerung verteilt ausgeführter Prozesspartitionen auf Basis der Repräsentation eines Prozessmanagementsystems als verwaltbare Ressource unterstützt, welche durch das hier vorgestellte Modell der anbieterseitigen Bereitstellung von Informations- und Steuerungsfunktionalitäten gekennzeichnet ist. Weitere Flexibilisierungsmöglichkeiten werden durch die abstrakte Beschreibung von Benutzerschnittstellen vor dem Vorbild des CTT-Modells und durch eine Integration von Vorhersagen relevanter Kontextdaten für dynamisch verteilt ausgeführte Prozesse diskutiert.

6 Strategien zur Flexibilisierung verteilter Prozessausführung

Die vorangegangenen Untersuchungen haben gezeigt, dass es zur Unterstützung vieler Anwendungsfälle vorteilhaft oder notwendig ist, dynamisch über die Verteilung von prozessorientierten Anwendungen entscheiden zu können. Eine Anpassbarkeit der Verteilung von Prozessen zu deren Laufzeit erfordert jedoch die Vermeidung starrer Strukturen, welche derzeit aus der physischen Partitionierung, der Kapselung von Prozesspartitionen und einer künstlichen Anreicherung der Prozessbeschreibung mit technischen Artefakten zur Verteilung oder Überwachung resultieren (vgl. Kapitel 5). In diesem Kapitel werden daher Möglichkeiten aufgezeigt, um die verteilte Ausführung von Prozessen unter benutzerdefinierten Rahmenbedingungen weitestgehend zu flexibilisieren. Dazu wird vorgeschlagen, in der ausführbaren Beschreibung eines Prozesses gänzlich von einer möglichen Verteilung zu abstrahieren und stattdessen die damit verbundenen Aufgaben angemessen in die ausführende Middleware zu integrieren. Nach einer kurzen Zusammenfassung grundlegender Annahmen und Voraussetzungen (vgl. Abschnitt 6.1) erfolgt dabei zur strukturierten Bearbeitung dieser Aufgabe eine Zerlegung des vorgeschlagenen Ansatzes in Teillösungsansätze für die in den Kapiteln 3 und 4 identifizierten Teilprobleme.

Zunächst wird als Ausgangsbasis hinsichtlich der Prozessbeschreibung ein nicht-invasiver Ansatz zur Verteilung verfolgt, welcher sich auf der Migration von Prozessinstanzen begründet und somit im Gegensatz zur physischen Partitionierung von Prozessen eine *logische Verteilung der Prozessausführung* propagiert. Für die Entkopplung der hierfür relevanten Instanzdaten und Verteilungsanweisungen wird eine separate Beschreibung vorgeschlagen, welche zur Entwicklungszeit oder zur Laufzeit des Prozesses erzeugt und während der verteilten Ausführung des Prozesses durch eine *Process-Management-as-a-Service-Komponente* als Ergänzung bestehender Prozessmanagementsysteme verwaltet werden kann. Des Weiteren werden die mit einer solchen Lösung verbundenen Herausforderungen wie die parallele Ausführung verteilter Prozesspartitionen oder die Erfüllung relevanter Sicherheitseigenschaften in diesem Kontext betrachtet. Die zu diesem Teilbereich erzielten Ergebnisse werden in Abschnitt 6.2 beschrieben.

Für eine flexible Überwachung und Steuerung verteilt ausgeführter Prozesse wird im zweiten Teilbereich dieses Kapitels ein Konzept vorgeschlagen, welches auf Basis der anbieterseitigen Bereitstellung relevanter Managementfunktionalitäten sowohl eine kontinuierliche Beobachtung relevanter Geschehnisse zeitnah zu ihrem Auftreten als auch eine spontane Abfrage und Anpassung von individuell und situativ relevanten Informationen erlaubt. Hierfür

wird die Abbildung eines *Prozessmanagementsystems als verwaltbare Ressource* vorgenommen, so dass die für eine verteilte Prozessausführung allgemein relevanten Informationen auf Basis eines einfachen gemeinsamen Metamodells in einheitlicher Art und Weise bereitgestellt und zugegriffen werden können. Abschnitt 6.3 stellt dementsprechend ein abstraktes Modell eines Prozessmanagementsystems als eine solche verwaltbare Ressource vor, welche durch eine dienstbasierte Management-Komponente als Erweiterung bestehender Ausführungssysteme in individuell verhandelbarer Art und Weise von den teilnehmenden Kooperationspartnern zugegriffen werden kann.

Als weiteres Teilproblem einer flexiblen Verteilung der Prozessausführung wurde eine bislang nicht hinreichend betrachtete Unterstützung von Benutzerinteraktionen identifiziert (vgl. Abschnitt 5.8). Dabei steht bei dynamisch wechselnden Umgebungsbedingungen im Vordergrund, dass die zur Laufzeit ausgewählten Ausführungseinheiten und ihr situativer Kontext zur Entwicklungszeit des Prozesses bzw. zum Zeitpunkt seiner Instantiierung noch nicht bekannt sind. In Abschnitt 6.4 wird hierzu ein Lösungsansatz vorgestellt, welcher die Definition von *abstrakten Benutzeroberflächen* nach dem *CTT-Modell* auf den Kontext verteilt ausgeführter Prozesse überträgt und die individuelle Durchführung der Interaktion mit dem Benutzer hinter einer lokalen dienstbasierten Schnittstelle kapselt. Zur Laufzeit des Prozesses kann damit die jeweils verantwortliche Instanz eines solchen *Interaktionsdienstes* auf Basis der abstrakten Beschreibung und der aktuellen Kontextinformationen eine speziell auf den Kontext des Benutzers angepasste Benutzungsschnittstelle generieren, welche sogar die Realisierung verschiedener Interaktionsmodalitäten unterstützt.

Der letzte Teil der hier dargestellten Ergebnisse behandelt den Umgang mit einer *proaktiven Anpassung der Verteilung* auf Basis einer Zukunftsprognose relevanter Ausführungskontexte. Da prozessorientierte Anwendungen im Allgemeinen über beliebige Inhalte und entsprechend hierfür relevante Kontexte verfügen können, wird mit dem Konzept der *strukturierten Kontextdatenprognose* ein Rahmenwerk zur Integration von Anwendungswissen, Prognosemethoden und Laufzeitdaten vorgeschlagen, welches sich sowohl durch eine generische Anwendbarkeit als auch durch die Einsetzbarkeit für besonders dynamische Umgebungen auszeichnet. Abschnitt 6.5 fasst die zu diesem Teilbereich erarbeiteten Ergebnisse zusammen.

Die Lösungsvorschläge zu den genannten Teilbereichen werden in Abschnitt 6.6 abschließend im Gesamtkontext dieser Arbeit betrachtet. Zusammenfassend enthält dieses Kapitel die Beschreibung der konzeptionellen Basis der hier beschriebenen Lösungsansätze, während die technische Umsetzung der Konzepte in Bezug auf entsprechende anwendungsorientierte Middleware-Komponenten Inhalt von Kapitel 7 ist. Eine Anwendung und Evaluation der Teilkonzepte in Bezug auf die eingangs vorgestellten Anwendungsszenarien und die definierten Anforderungen findet in Kapitel 8 statt.

6.1 Allgemeine Annahmen und Voraussetzungen

Die in diesem Kapitel vorgestellten Konzepte basieren auf dem aktuellen Stand der Forschung, d. h. insbesondere auf der Annahme des Vorliegens von dienstorientierten Architekturen, standardisierten Prozessbeschreibungssprachen und den zur (zentralen) Ausführung und Verwaltung von entsprechenden Prozessen geeigneten Prozessmanagementsystemen mit den in Kapitel 2 genannten Eigenschaften. Es wird insbesondere angenommen, dass

- die dynamisch zu verteilenden Prozesse in einer ausführbaren Repräsentation fachlich abgeschlossen definiert sind und eine fixe inhaltliche Struktur besitzen,
- offene oder geschlossene Systeme mit einer hinreichend gemeinsamen semantischen Interpretation der domänenspezifischen Anwendungsvorgänge bestehen und auf einer hinreichend gemeinsamen technischen Infrastruktur basieren (z. B. [KZL07a, ZDL09b]),
- eine ausreichende Menge von geeigneten (stationären oder mobilen) Ausführungseinheiten existiert, welche organisationsintern und/oder von externen (an einer Kooperation interessierten) Organisationen bzw. Organisationseinheiten bereitgestellt werden,
- eine ausreichende Menge von global oder lokal erreichbaren Dienstinstanzen existiert, welche ggf. funktional äquivalent sind und von unterschiedlichen Dienstanbietern mit potentiell unterschiedlichen nichtfunktionalen Eigenschaften angeboten werden können,
- zentrale oder lokale Verzeichnisdienste zum (dynamischen) Auffinden und Einbinden von Dienstinstanzen genutzt werden können (vgl. z. B. [ZDL09b]),
- unterstützende Kontextmanagementsysteme und domänenspezifische Kontextmodelle existieren, welche einem einzelnen Ausführungssystem Daten über seinen lokalen Kontext zur Verfügung stellen können (vgl. z. B. [KZTL08]),
- Ansätze und technische Infrastrukturen zur (dynamischen) Aushandlung von nicht-funktionalen Eigenschaften einer Dienstnutzung bestehen,
- ein für eine Kooperation ausreichendes Maß an Vertrauen zwischen den Ausführungseinheiten besteht und sich Ausführungseinheiten nicht bewusst böswillig oder in anderer Weise absichtlich destruktiv verhalten.

Im Gegensatz zu vielen der in Kapitel 5 betrachteten Ansätze zur Verteilung der Prozessausführung schlägt diese Arbeit vor, nicht jeden einzelnen Prozess für eine potentielle Verteilung anzupassen, sondern die bereits verwendeten Prozessmanagementsysteme für die damit zusammenhängenden Aufgaben geeignet zu erweitern. Dies setzt die Möglichkeit zu einer (im Idealfall

einmaligen) Anpassung der Prozessmanagementsysteme voraus, welche entweder durch die Hersteller kommerzieller Produkte oder durch die Entwickler oder Nutzer individueller oder quelloffener Systemsoftware durchgeführt werden muss. Die generelle Anpassbarkeit der Ausführungssysteme (d. h. im Speziellen die Verfügbarkeit und Erweiterbarkeit deren Quellcodes) ist daher ebenfalls eine wichtige Voraussetzung und späterer Betrachtungsgegenstand dieser Arbeit.

6.2 Verfahren zur dynamischen Verteilung der Prozessausführung

Wie in Kapitel 4 motiviert wurde, setzt ein Ansatz zur dynamischen Verteilung der Prozessausführung voraus, dass die Verantwortlichkeit für die Ausführung des Kontrollflusses zur Laufzeit des Prozesses auf Basis der aktuellen organisatorischen, technischen oder physikalischen Umgebungsbedingungen geeignet partitioniert und an eine oder mehrere Ausführungseinheiten zugewiesen werden kann. Dabei kann oft nicht davon ausgegangen werden, dass eine geeignete Granularität der Verteilung oder die konkreten, tatsächlich teilnehmenden Ausführungseinheiten bereits vor der Instantiierung des betrachteten Prozesses bekannt sind. Die bisher im Rahmen von dienstorientierten Architekturen angewendete physische Partitionierung zur Entwicklungszeit des Prozesses, die Verteilung der Partitionen zur Zeit des Deployments und die Auswahl der tatsächlichen Ausführungseinheiten zur Laufzeit des Prozesses müssen in Hinblick auf die identifizierten Flexibilitätsanforderungen daher als zu unflexibel betrachtet werden.

Das hier dargestellte Verfahren schlägt die gemeinsame Migration eines Prozessmodells mit den dazugehörigen Instanzdaten als Mittel zur Realisierung einer *logischen Partitionierung* vor, welche lediglich die Verantwortlichkeit für die Ausführung eines Prozesses in eine Menge von Teilverantwortungsbereiche zerlegt, während die ursprüngliche ausführbare Prozessbeschreibung in ihrer bei der Prozessmodellierung geschaffenen Struktur und ihrem dort festgelegten Inhalt für alle teilnehmenden Ausführungseinheiten im Ganzen bewahrt wird.

Eine solche Migration entspricht im Wesentlichen einer natürlichen Arbeitsweise von Personen zur Bearbeitung einer aus mehreren Schritten bestehenden verteilt auszuführenden Aufgabe, wie sie zum Beispiel im Rahmen der gemeinsamen Bearbeitung von (rein) papiergebundenen Dokumenten angewendet wird. Hierbei wird eine papiergebundene Aufgabenbeschreibung zu einem bestimmten Anwendungsfall als Kopie einer Dokumentenvorlage von einem Arbeitsplatz zum nächsten weitergereicht und dabei jeweils um individuelle Bearbeitungsdaten dieses Vorgangs ergänzt (vgl. [Gry96, Zül98]). Die Mitnahme des papiergebundenen Dokuments an andere Orte ermöglicht dabei auch die Mobilität eines Bearbeiters. Für eine parallele Ausführung zweier Ausführungspfade kann die Aufgabenbeschreibung kopiert oder physisch zerteilt werden. In allen Fällen entscheidet dabei jeweils der Bearbeiter der aktu-

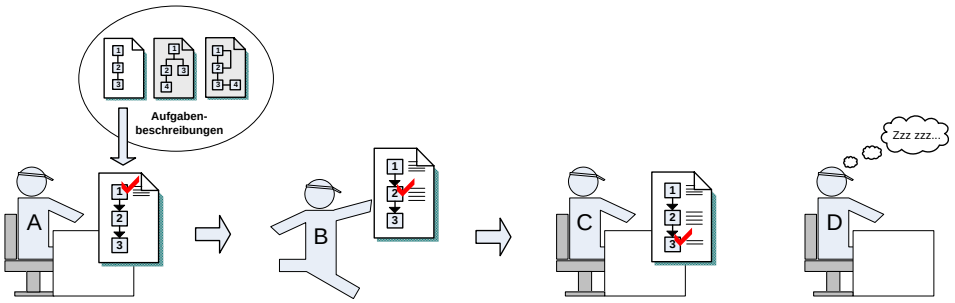


Abbildung 6.1: Leitbild der kooperativen Bearbeitung von strukturierten Aufgaben durch Weitergabe der Aufgabenbeschreibung und der individuellen Bearbeitungsdaten

ellen Aufgabe anhand der vorgegebenen Abhängigkeiten sowie der aktuellen Daten und Bedingungen selbständig, ob – und falls ja, an welchen Arbeitsplatz – die Bearbeitung der Folgeaufgaben delegiert werden soll. Die Bearbeitung der Aufgaben und die Zuweisung der Verantwortlichkeiten können auf diese Weise vollkommen dezentral durchgeführt werden [ZKML10, ZHKK10]. Abbildung 6.1 veranschaulicht diese grundlegende Arbeitsweise angelehnt an die Darstellung von GRYCZANs und ZÜLLIGHOVENs Prozessmuster mit *Vorgangsmappe* und *Laufzettel* als softwaretechnische Entwicklungsmetaphern für die objektorientierte Programmierung [Gry96, Zül98].

Eine diesem Leitbild entsprechende Vorgehensweise für die Verteilung von Prozessen erlaubt die dynamische Partitionierung eines Prozesses individuell für jede Prozessinstanz basierend auf dem jeweils aktuellen Kontext der Prozessausführung. Dabei kann die Granularität der Verteilung durch die lokale Entscheidung der Ausführungseinheiten zur Fortführung oder zur Weitergabe der Prozessausführung dynamisch festgelegt werden. Ein großer Teil des Koordinationsaufwands zwischen verschiedenen Ausführungseinheiten kann zudem vermieden werden, indem die Prozessbeschreibung mit allen relevanten Kontrollflussstrukturen und Daten zur Laufzeit nur den *tatsächlich* an der Prozessausführung teilnehmenden Ausführungseinheiten verfügbar gemacht wird, während aktuell nicht beteiligte Ausführungseinheiten nicht durch vorbereitende Maßnahmen unnötig belastet werden müssen (vgl. Person *D* in Abbildung 6.1). Die Möglichkeiten zur Offline-Bearbeitung von Prozessschritten, zur dezentralen Ausführung und zur ressourcenschonenden Verteilung von nur tatsächlich lokal auszuführenden Prozessen machen diesen Ansatz der *Prozessmigration* ebenfalls für die Anwendung auf moderne mobile oder hybride Systeminfrastrukturen anwendbar.

Die in Abschnitt 5.5 betrachteten bestehenden Ansätze zur Migration von Prozessen sind jedoch mehr oder weniger stark auf einen speziellen Zweck ausgerichtet und besitzen aufgrund ihrer speziellen Ziele in der Regel auch spezielle Beschreibungssprachen, was ihre Übertragbarkeit auf eine große Menge von bereits bestehenden, zuvor aufwendig in teilweise sogar standardisierten Prozessbeschreibungssprachen modellierten Prozessen erschwert. In diesem Abschnitt wird daher zunächst eine allgemeine *Entwicklungsmethodik*

vorgestellt, welche zwei grundlegende Vorgehensweisen zur dynamischen Verteilung von Prozessen beinhaltet (vgl. Abschnitt 6.2.1). Als Hilfsmittel für die dynamische Verteilung laufender Prozessinstanzen wird im Anschluss ein *allgemeines* Modell für die Migration von Prozessen beschrieben (vgl. Abschnitt 6.2.2). Hierbei wird einerseits basierend auf den in Kapitel 2 erarbeiteten Grundlagen eine abstrakte Beschreibung für den Zustand einer Prozessinstanz und ihrer relevanten Instanzdaten (unabhängig von einer bestimmten Prozessbeschreibungssprache) erarbeitet. Zum anderen erfolgt eine Abstraktion von Anweisungen zur Durchführung der Verteilung, um die Modellierung von Rahmenbedingungen für eine sinnvolle Zuordnung von Prozesspartitionen zu Ausführungseinheiten zu erlauben und fortwährend anpassbar zu halten.

Ausgehend von einer vollständigen Freiheit für die Migration von Prozessen werden im Folgenden sinnvolle Einschränkungen der Flexibilität zur Gewährleistung der Korrektheit der Prozessausführung und zur Berücksichtigung fachlicher oder qualitativer Vorgaben des Prozessmodellierers, Prozessinitiators oder Prozessteilnehmers erarbeitet. Dazu werden zunächst die potentiellen Eigenschaften der für die Prozessausführung relevanten Ressourcen und die Art ihrer Bindung in Hinblick auf die generelle Migrierbarkeit des Prozesses untersucht (vgl. Abschnitt 6.2.3). Es folgt eine Betrachtung der dynamischen Bildung von Prozesspartitionen mit sequentiellem Kontrollfluss (vgl. Abschnitt 6.2.4) und die Identifikation von Maßnahmen zur individuellen Verteilung von parallelen Prozesspartitionen (vgl. Abschnitt 6.2.5). Schließlich werden die Möglichkeiten zur Migration von Prozessen mit ereignisbasiertem Kontrollfluss untersucht (vgl. Abschnitt 6.2.6).

Als wesentliche Rahmenbedingung zum Schutz des Prozesses und der beteiligten Ausführungsumgebungen wird im Anschluss gezeigt, wie die Verteilung auf der Basis von benutzerdefinierten Sicherheitsrichtlinien gezielt eingeschränkt werden kann (vgl. Abschnitt 6.2.7). Die prinzipielle Integration des Verfahrens zur dynamischen Verteilung der Prozessausführung in bestehende dienstorientierte Architekturen durch den Ansatz des *Process-Management-as-a-Service* ist Inhalt von Abschnitt 6.2.8. Eine Zusammenfassung der Erkenntnisse erfolgt in Abschnitt 6.2.9.

6.2.1 Prinzipien und Entwicklungsgrundsätze

In den Abschnitten 3.9.4 und 5.5 wurden verschiedene Varianten der Migration von Computerprogrammen im Allgemeinen und von Prozessen auf Anwendungsebene im Speziellen untersucht. Auf Basis der dort gewonnenen Erkenntnisse wird in diesem Abschnitt eine Vorgehensweise motiviert, welche es erlaubt, bestehende Prozesse durch eine Anreicherung mit Migrationsdaten flexibel auf verschiedene Ausführungseinheiten zu verteilen.

Vorüberlegung

Konzeptionell bestehen für die Migration von Prozessmodell und Instanzdaten nach dem aufgezeigten Leitbild zwei hier betrachtete Lösungsansätze. Da-

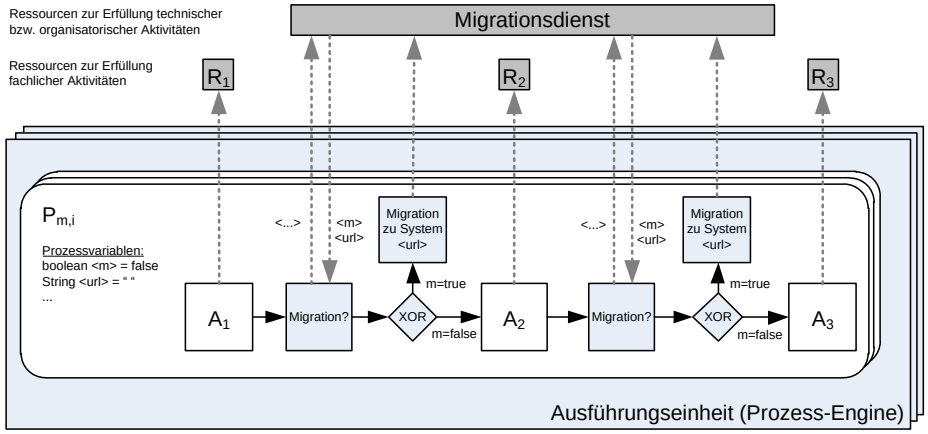


Abbildung 6.2: Alternative 1: Anreicherung der ausführbaren Prozessbeschreibung mit Migrationsaktivitäten und -daten als Ausgangspunkt einer invasiven Migration am Beispiel des Prozessmodells PM_m mit den funktionalen Aktivitäten A_1, A_2 und A_3 und einer von PM_m abgeleiteten Prozessinstanz $P_{m,i}$

bei sei im Folgenden angenommen, dass auf einer Ausführungseinheit zu jedem Prozessmodell PM_m verschiedene Prozessinstanzen $P_{m,i}$ abgeleitet werden können, wobei m die Zuordnung zum Prozessmodell und i die individuelle Prozessinstanz kennzeichnet.

Der erste mögliche Lösungsansatz umfasst die unmittelbare Integration von Migrationsanweisungen in die ausführbare Prozessbeschreibung. Ein Beispiel hierfür ist das Einfügen von speziellen Aktivitäten zur Migration des Prozesses, welche jeweils nach jeder funktionalen Aktivität aufgerufen werden. Somit lässt sich die ausführende Prozess-Engine bei der Bearbeitung des Prozesses anhalten, um (z. B. durch das Aufrufen eines lokalen oder entfernten Dienstes) über die weitere Ausführung oder die Verteilung des laufenden Prozesses zu entscheiden. Migrationsdaten und Zielorte der Verteilung können bei dieser Variante als Variablen des Prozesses integriert werden. Weitere Kontrollflusskonstrukte wie alternative Pfade oder Schleifen können genutzt werden, um die Verteilung an mehrere Kooperationspartner zu spezifizieren oder Ausnahmesituationen zu behandeln (z. B. wenn eine vorgesehene Migration aus technischen Gründen scheitert). Diese Variante wird in dieser Arbeit als *invasive Migration* bezeichnet [ZKML10, ZHKK10] und ist beispielhaft in Abbildung 6.2 dargestellt.

Die invasive Migration hat, wie auch die entsprechenden verwandten invasiven Ansätze (vgl. Abschnitte 5.4.5, 5.4.6 oder 5.7.3) den Vorteil, dass zunächst keine Änderungen an den bestehenden prozessausführenden Systemen (d. h. insbesondere an den Prozess-Engines) der teilnehmenden Organisationen vorgenommen werden müssen. Des Weiteren können viele der für die Anpassung der Prozessmodelle notwendigen technischen Anweisungen durch Elemente von Standard-Prozessbeschreibungssprachen abgebildet und so in den Prozess integriert werden. Auf der anderen Seite müssen hierfür jedoch vorab alle ausführbaren Prozessmodelle entsprechend erweitert werden, was die

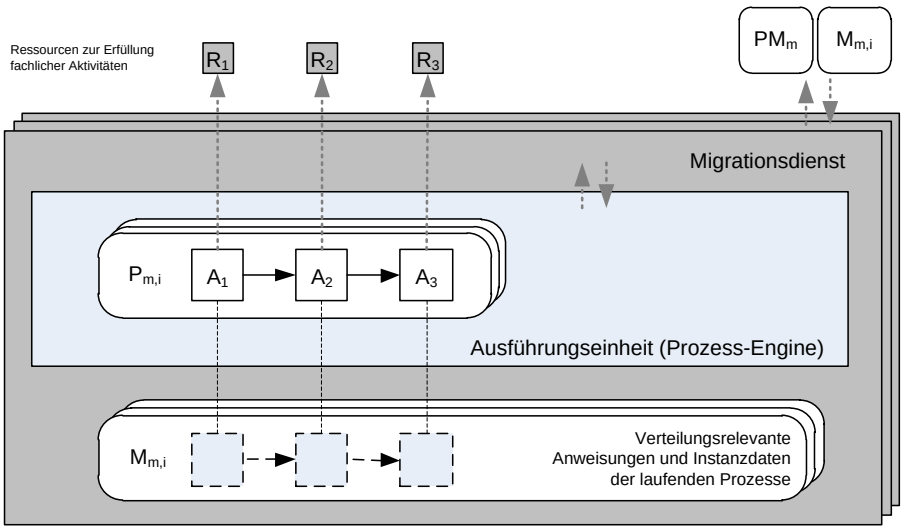


Abbildung 6.3: Alternative 2: Entkopplung der für die Verteilung notwendigen Daten und Anweisungen als Ausgangspunkt für eine nicht-invasive Migration am Beispiel des Prozessmodells PM_m mit den funktionalen Aktivitäten A_1, A_2, A_3 und den zur Prozessinstanz $P_{m,i}$ gehörigen Migrationsdaten $M_{m,i}$

nicht-verteilte Ausführung davon abgeleiteter Prozessinstanzen negativ beeinflussen bzw. verzögern kann. Abbildung 6.2 verdeutlicht beispielhaft, dass bereits für einen einfachen sequentiellen Prozess mit nur drei funktionalen Aktivitäten mindestens sechs neue technische Kontrollflusskonstrukte eingefügt werden müssen, um später eine beliebige Granularität bei der Verteilung des Prozesses zu ermöglichen. Neben einer unerwünschten Erhöhung der Komplexität aller potentiell zu verteilenden Prozesse wird dieser Ansatz auch der Forderung nach einer angemessenen Trennung von fachlichen und technischen Anweisungen nicht gerecht. Geht man zudem nicht davon aus, dass das auszuführende Prozessmodell bereits bei allen potentiell teilnehmenden Ausführungseinheiten bekannt ist, so stellt sich außerdem die Frage, wie der Transfer der Prozessbeschreibung von einer Ausführungseinheit zur nächsten organisiert werden soll, da sich eine passive Prozessbeschreibung nicht (wie z. B. ein mobiler Software-Agent) selbst migrieren kann. Im Vergleich mit einer physischen Partitionierung ergeben sich also zwar durch die logische Partitionierung Möglichkeiten zur Flexibilisierung, diese sind jedoch durch die Invasivität des Ansatzes begrenzt bzw. durch die genannten Nachteile begleitet [ZKML10, ZHKK10].

Die zweite Möglichkeit umfasst eine *nicht-invasive Migration* durch die Entkopplung der für die Verteilung notwendigen Daten und Anweisungen sowie deren Beschreibung auf einem geeigneten Abstraktionsniveau. Dieser Ansatz orientiert sich an der Ausführung klassischer nicht-verteilten Prozesse, bei denen in der Regel ebenfalls keine technischen Anweisungen zur organisatorischen Verwaltung oder zur Auswahl von Ressourcen in die Prozessbeschreibung eingewoben werden müssen. Für eine Verteilung der Prozessausführung wird bei der nicht-invasiven Migration bei Bedarf entsprechend aus dem aktu-

ellen Ausführungskontext eine separate Spezifikation von geeigneten *Migrationsdaten* abgeleitet und zusammen mit dem ursprünglichen Prozessmodell an eine unterstützende Middleware weitergegeben, so dass die laufende Prozessinstanz mit ihrem aktuellen Zustand auf einem ausgewählten Zielsystem installiert und dort an derselben Position im Kontrollfluss fortgeführt werden kann, wo die Ausführung des Vorgängersystems beendet wurde. Im Idealfall muss hierzu die fachliche Beschreibung des ausführbaren Prozessmodells nicht modifiziert werden und eine Verteilung ist prinzipiell auch für (unpräperierte) Prozesse möglich, deren Ausführung bereits begonnen hat [ZKML10, ZHKK10]. Eine grobe Darstellung dieses Ansatzes ist zum Vergleich beider Varianten in Abbildung 6.3 visualisiert. In beiden Fällen kann der jeweils dargestellte *Migrationsdienst* als Beispiel einer unterstützenden Middleware prinzipiell sowohl zentral als auch dezentral bzw. als lokale Erweiterung des Prozessmanagementsystems integriert werden. Im Gegensatz zur invasiven Migration, wo diese Middleware jedoch von Seiten der Ausführungsumgebung relativ unkontrollierbar aus den Prozessen selbst heraus aufgerufen und verwendet wird, wird im Fall der nicht-invasiven Migration der Prozess von der Middleware kontrolliert und verwaltet. Aus dieser Sichtweise folgt (im Gegensatz zum Ansatz mobiler Agenten) ein größeres Maß an Kontrolle über den Prozess und damit mehr Autonomie für die beteiligten Ausführungseinheiten mit ihren etwaigen individuellen lokalen Richtlinien.

Entwicklungsmethodik

Die in dieser Arbeit vorgeschlagene grundlegende Vorgehensweise für eine *nicht-invasive* Migration ist in Abbildung 6.4 dargestellt (vgl. [ZKML10, ZHKK10]). Ausgangspunkt der Betrachtung ist die Modellierung eines fachlichen Prozesses und seine entsprechende Abbildung in einer automatisiert ausführbaren Repräsentation, wie zum Beispiel in standardisierten Prozessbeschreibungssprachen wie XPDL, WS-BPEL oder BPMN (vgl. Abschnitt 5.2.4). Zur Prozessmodellierung können hierbei weiterhin die bekannten bzw. bestehenden Werkzeuge in unveränderter Form eingesetzt werden oder bereits bestehende Prozessmodelle oder -vorlagen zugrunde gelegt werden. Resultat dieses ersten Schritts ist in jedem Fall ein ausführbares Prozessmodell PM_m, welches nun – analog zum klassischen nicht-verteilten Prozessmanagement – direkt auf einer geeigneten Prozess-Engine installiert, instantiiert und auf diese Weise zentral ausgeführt werden kann.

Wird eine Verteilung des Prozesses bereits vor dessen Deployment in Betracht gezogen, so erlaubt der zweite Schritt optional die Definition von Rahmenbedingungen für die spätere Verteilung aller von diesem Prozessmodell abgeleiteten Prozessinstanzen. Hierfür kann basierend auf dem Prozessmodell PM_m die Erzeugung von prozessmodellspezifischen *Migrationsdaten* MM_m angestoßen werden, welche den Prozess „migrationsfähig“ machen und auf deren Basis die generellen Vorgaben des Prozessmodellierers integriert werden können. Werden solche Vorgaben spezifiziert, so wird das entsprechend re-

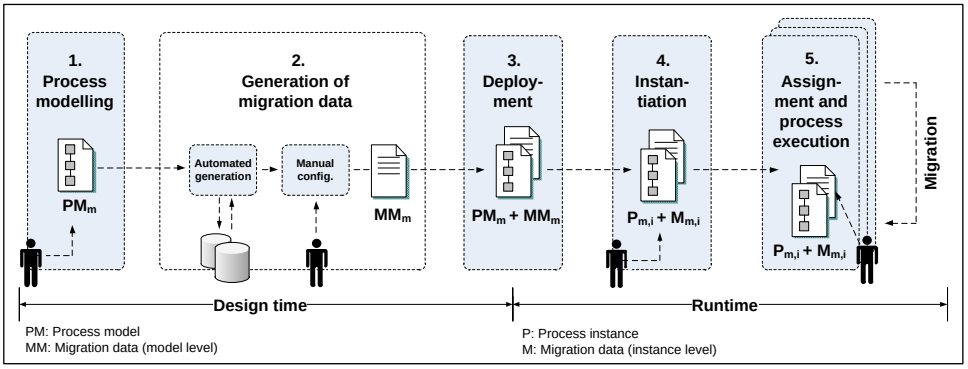


Abbildung 6.4: Entwicklungsmethodik dynamisch verteilt ausgeführter Prozesse (ZKML10, ZHKK10)

sultierende, mit Verteilungsanweisungen angereicherte Migrationsmodell im Rahmen des Deployments des Prozessmodells gemeinsam mit diesem auf einer ausgewählten Prozess-Engine gespeichert bzw. installiert (vgl. Schritt 3 in Abbildung 6.4).

Bei jeder Instantiierung des Prozesses durch eine Anwendung oder einen Benutzer kann nun die erzeugte Prozessinstanz $P_{m,i}$ durch die Eingabeparameter des Prozessinitiators individualisiert werden (Schritt 4). Ebenso können bei diesem Schritt bei Bedarf weitere individuelle Vorgaben des Prozessinitiators zum Migrationsmodell der Prozessinstanz $M_{m,i}$ hinzugefügt werden. Dies ist insbesondere dann vorteilhaft, wenn der Prozessinitiator die nicht-funktionalen Aspekte der aktuellen Prozessausführung gesondert beeinflussen darf. Im Kontext dienstorientierter Architekturen kann diese Anpassung zum Beispiel auf Basis eines Service-Level-Agreements geschehen. Wird zum Beispiel verhandelt, dass der Dienstnutzer einen höheren Preis für eine besondere Dienstgüte bezahlt, so müssen unter Umständen auch die zur Ausführung des Prozesses benötigten Kooperationspartner nach den verhandelten Kriterien ausgewählt werden. Durch die Auswertung des Service-Level-Agreements kann der Dienstanbieter dann stellvertretend für den Dienstnutzer die entsprechenden Vorgaben in die Migrationsdaten integrieren, um gezielt die Ausführung dieser bestimmten Prozessinstanz nach den Wünschen des Dienstnutzers anzupassen.

Der fünfte Schritt umfasst die Ausführung der Prozessinstanz $P_{m,i}$ nach den allgemeinen und/oder individuellen Vorgaben. Werden keine Vorgaben zur Verteilung spezifiziert oder eine Verteilung durch die Vorgaben untersagt, so kann der Prozess unbeeinflusst zentral von Anfang bis Ende auf einem einzigen Ausführungssystem bearbeitet werden (*nicht-verteilte Ausführung*). Ansonsten kann der Prozess auf Basis der Vorgaben oder der lokalen Richtlinien der Ausführungssysteme durch die Migration dynamisch partitioniert und an andere Ausführungseinheiten weitergegeben werden. Während der Ausführung des Prozesses können dabei weiterhin befugte Personen (z. B. Prozessteilnehmer oder Administratoren) auf die Verteilung der individuellen Prozessinstanz

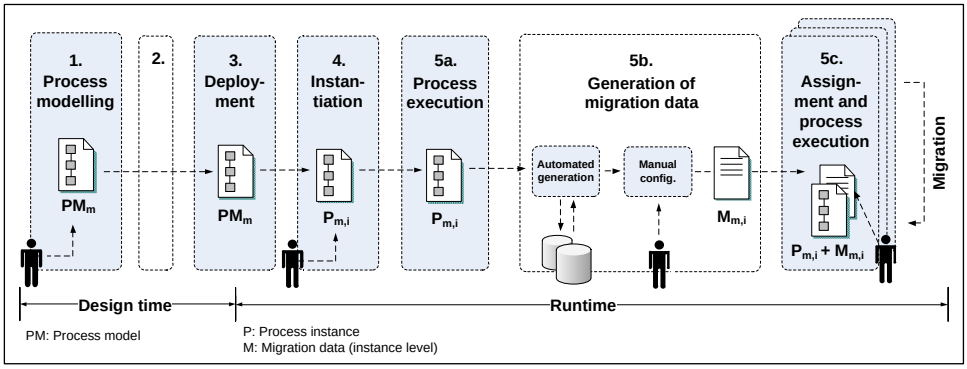


Abbildung 6.5: Spontane Verteilung bereits zentral ausgeführter Prozesse (ZKML10, ZHKK10)

einwirken, indem die benutzerdefinierten Vorgaben in den Migrationsdaten $M_{m,i}$ den aktuellen Bedürfnissen angepasst werden.

Die dargestellte Vorgehensweise richtet sich zunächst an Anwendungsfälle, für deren Unterstützung eine verteilte Ausführung des Prozesses zumindest prinzipiell in Erwägung gezogen wird, bevor der Prozess instantiiert und seine Ausführung begonnen wird. Soll auch eine unvorhergesehene Verteilung der Prozessausführung zur Laufzeit unterstützt werden, so ist es notwendig, die Erzeugung des Migrationsmodells $M_{m,i}$ und ggf. die Spezifikation von Vorgaben durch einen Prozessbeteiligten *während des laufenden Prozesses* zu ermöglichen. Abbildung 6.5 zeigt die entsprechend angepasste Vorgehensweise für den Fall, dass Schritt 2 ausgelassen wurde, d. h., dass vor der Instantiierung des Prozesses kein Migrationsmodell generiert bzw. spezifiziert wurden. Ein Anwendungsfall hierfür ist die spontane Weiterführung eines bereits zentral gestarteten Prozesses auf dem mobilen Gerät eines Prozessbeteiligten. Der Prozessbeteiligte (bzw. eine unterstützende Anwendung) muss in diesem Fall spezifizieren, welcher Teil des Prozesses auf welcher Ausführungseinheit ausgeführt werden soll. Da in dieser spontanen Variante der Prozessmigration zunächst einmal die zentrale Ausführung des Prozesses kontrolliert angehalten werden muss, müssen hierfür erweiterte Steuerungsfunktionalitäten der lokalen Prozess-Engine zugegriffen werden können (vgl. Abschnitt 6.3).

Die folgenden Abschnitte beschreiben die Konzeption der hier beschriebenen Teilaspekte eines solchen Vorgehens im Allgemeinen. Zunächst wird hierfür identifiziert, welche Informationen notwendig sind, um eine Prozessinstanz abstrakt so zu beschreiben, dass alle notwendigen Informationen aus dem aktuellen Ausführungskontext erfasst und dem folgenden Ausführungssystem für eine nahtlose Fortführung des Prozesses zur Verfügung gestellt werden können. Zudem wird darauf basierend vorgeschlagen, wie diese Daten mit den benutzerdefinierten Vorgaben als Rahmenbedingungen für die Verteilung so in Zusammenhang gebracht werden können, dass daraus zur Laufzeit des Prozesses kontrolliert Anpassungsbedarf abgeleitet werden kann.

6.2.2 Metamodell für die Migration von Prozessen

Eine flexible Partitionierung und Verteilung von Prozessen durch die Migration von Prozessinstanzen erfordert die Identifikation und Beschreibung aller relevanten Informationen und Anweisungen, um einen bestehenden Prozess zur Laufzeit „migrierbar“ zu machen. Dies umfasst in erster Linie die Erfassung des aktuellen Zustands des Prozesses in Form seiner *Instanzdaten*. Hierbei ist es wesentlich, dass ein Prozess bei seiner Migration immer von einem konsistenten Zustand auf dem Quellausführungssystem in denselben konsistenten Zustand auf dem Zielausführungssystem überführt werden kann. Um einen solchen konsistenten Zustand zu gewährleisten, wird in dieser Arbeit die Ausführung einer einzelnen (nicht weiter verfeinerbaren) Aktivität als *atomarer Schritt* betrachtet. Dies hat den Hintergrund, dass auch Interaktionen mit menschlichen Prozessteilnehmern und somit die Durchführung von manuellen Aktivitäten nicht ausgeschlossen werden sollen, deren interner Fortschritt sich in der Regel nicht in einer Datenbank abbilden lässt. Das gleiche Argument gilt für die Ausführung von gekapselten Softwareanwendungen (z. B. in Form von elektronischen Diensten), deren internes Verhalten für das Prozessmanagementsystem nicht ersichtlich ist und in der Regel auch nicht beeinflusst (z. B. temporär angehalten) werden kann.

Dieser Überlegung entsprechend wird als allgemeine Integritätsbedingung der Prozessmigration gefordert, dass laufende Prozessinstanzen nicht migriert werden dürfen, solange atomare Aktivitäten durch (von der lokalen Prozess-Engine aufgerufene) Ressourcen oder durch die Prozess-Engine selbst ausgeführt werden [KZL07b]. Eine bereits begonnene Ausführung einer Aktivität muss somit zunächst beendet (oder alternativ im Rahmen einer lokalen Transaktionsverwaltung abgebrochen und zurückgesetzt) werden, bevor der Prozess migriert werden kann. Zudem müssen die Datenwerte aller Prozessvariablen vorliegen und ebenfalls einen konsistenten Zustand aufweisen. Da hierbei angenommen wird, dass die innerhalb des Prozesses explizit spezifizierten Variablen nur durch die Ausführung von Aktivitäten gelesen bzw. modifiziert werden können (vgl. Abschnitt 2.5), ist diese Forderung ebenfalls erfüllt, solange die Datenwerte nicht während der Ausführung atomarer Aktivitäten erhoben werden – was bereits durch die zuvor genannte Bedingung verhindert wird. Falls durch erweiterte Konzepte (z. B. im Rahmen einer inhaltlichen Anpassung) von außerhalb des Prozesses auf Prozessvariablen zugegriffen wird, so muss an dieser Stelle (ebenfalls unter Kontrolle der lokalen Transaktionsverwaltung) auf eine Freigabe der betreffenden Daten gewartet werden. Schließlich müssen für die Durchführung einer Migration noch die exakten Positionen im Kontrollfluss des Prozesses gekennzeichnet werden können, an denen die Bearbeitung des Prozesses später wieder aufgenommen werden kann. Dies erfordert zusammengefasst eine Überprüfung aller Aktivitäten des Prozesses und eine eindeutige Identifikation ihrer aktuellen Zustände.

Das hier vorgeschlagene (Meta-)Modell für die Migration von Prozessen bildet mit der Annahme eines gemeinsamen *minimalen Prozessmetamodells* und

eines darauf angepassten *Zustandsmodells* für Prozesse und Aktivitäten die genannten Anforderungen hinsichtlich der Migration einer Prozessinstanz ab. Diese bilden zudem den Ausgangspunkt für eine Verknüpfung von *benutzerdefinierten Vorgaben und Rahmenbedingungen* für die Verteilung des Prozesses. Die resultierenden Entitäten und Beziehungen sind zusammenfassend in Abbildung 6.6 dargestellt und werden im Folgenden jeweils im Detail erläutert.

Minimales gemeinsames Prozessmetamodell

Als Abstraktionsebene des tatsächlich zu migrierenden Prozesses dient ein *minimales Prozessmetamodell*, welches aus der Analyse der grundlegenden Eigenschaften prozessorientierter Anwendungen im Allgemeinen abgeleitet wurde (vgl. Kapitel 2). Dabei wird der auszuführende Prozess (*Process*) durch eine endlichen Menge von Aktivitäten (*Activities*) und eine endliche Menge von Prozessvariablen (*Variables*) dargestellt. Aktivitäten können dabei entweder eine zu erfüllende fachliche Aufgabe oder eine Kontrollflussentscheidung repräsentieren (*Atomic Activity*) oder im Rahmen einer Blockstruktur als Behälter für die Gruppierung anderer Kontrollflusselemente verwendet werden (*Structured Activity*). Variablen sind Behälter für Werte, welche als Ein- oder Ausgabedaten dieser Aktivitäten verwendet oder zur Navigation des Kontrollflusses benötigt werden. Sie können entweder auf globaler Ebene für den ganzen Prozess definiert sein (*Global Variable*) oder nur in ihrer Sichtbarkeit auf die Ebene einzelner (strukturierter) Aktivitäten beschränkt sein (*Local Variable*). Alle Variablen können optional zur Entwicklungszeit mit einem vorgegebenen Wert initialisiert werden (*Initial Value*). Da für eine nichtinvasive Migration beide Entitäten aus den Migrationsdaten referenziert werden müssen, müssen sowohl Aktivitäten als auch Variablen innerhalb eines Prozesses jeweils einen eindeutigen Identifikator (*ID*) bzw. einen eindeutigen Variablennamen (*Name*) besitzen [ZKML10, ZHKK10].

Zustandsmodell der Prozessinstanz

Auf Basis dieser Abstraktion kann die Ausführung von Prozessen, deren Meta-Modell sich auf dieses minimale Prozessmetamodell abbilden lässt (z. B. unter anderem Prozessmodelle in XPDL, BPMN oder WS-BPEL, vgl. Abschnitte 5.2.4 und 8.3), durch Migrationsdaten erweitert werden, welche den aktuellen Ausführungszustand der betrachteten Prozessinstanz (*Process State*) und jeder ihrer Aktivitäten (*Activity State*) zu jedem Zeitpunkt der Prozessausführung widerspiegeln. Der Zustand des Prozesses kann dabei jeweils genau einen Wert aus dem Zustandsmodell migrierbarer Prozesse annehmen, welches auf Basis des etablierten Zustandsmodells von LEYMANN und ROLLER [LR00] auf die Migration von Prozessen angepasst wurde [KZL07b, Kun08] (vgl. Abschnitt 2.7.2). Abbildung 6.6 zeigt das erweiterte Zustandsmodell migrierbarer Prozesse (*Migratable Process State Model*) als Teil des Migrationsmodells im oberen rechten Bereich der Abbildung.

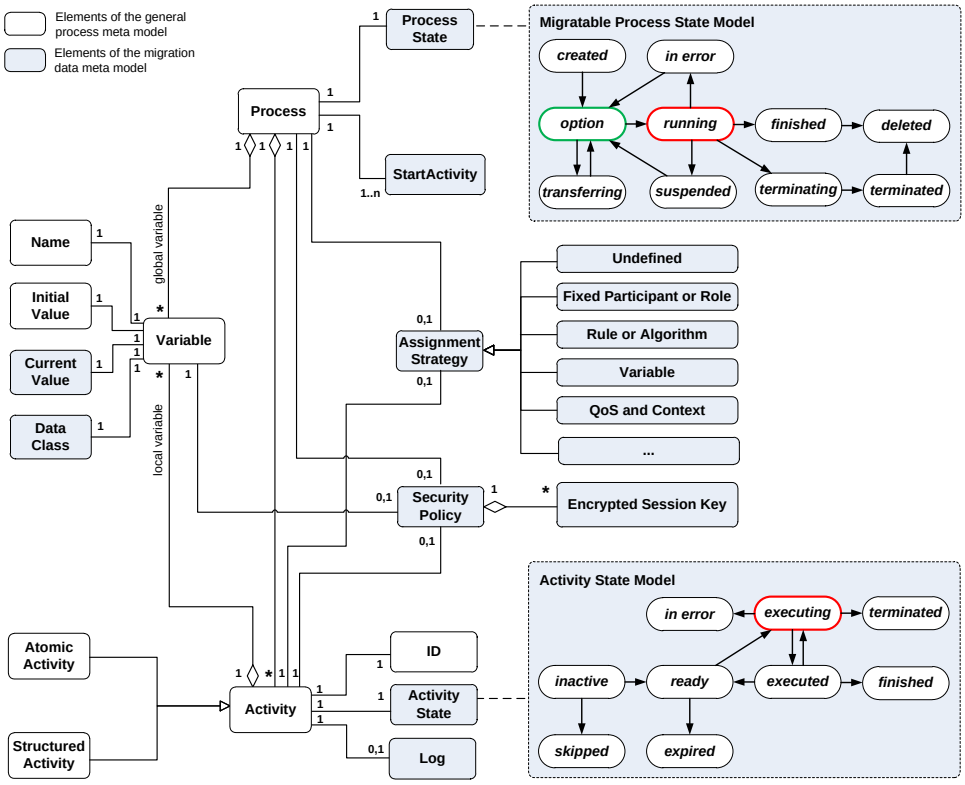


Abbildung 6.6: Metamodell für die Spezifikation von Migrationsdaten (ZKML10, ZHKK10)

Der neu eingefügte Zustand *Option* [KZL07b, Kun08] wird hierbei immer dann erreicht, wenn temporär keine atomaren Aktivitäten ausgeführt werden und dementsprechend ein konsistenter Ausführungszustand vorliegt, der die Migration des Prozesses erlaubt. In diesem Zustand kann jeweils entschieden werden, ob der Prozess weiter auf dem lokalen System ausgeführt oder migriert werden soll. In letzterem Fall wird der Prozess in den Zustand *Transferring* versetzt, welcher den Zustand der Migration beschreibt und dabei alle Maßnahmen umfasst, welche zur Deinstallation des Prozesses auf dem Quellausführungssystem, zur Suche und Auswahl geeigneter Zielausführungssysteme, zur Übertragung sowie zur Installation des Prozesses und zu dessen nahtloser Fortführung benötigt werden [KZL07b, Kun08].

Der Zustand *Option* kann nach dem Auftreten eines (technischen) Fehlers erreicht werden, z. B. wenn eine Ressource lokal nicht verfügbar ist (Zustand *InError*), oder wenn die Prozessausführung aktiv von außen angehalten wird (Zustand *Suspended*). In beiden Fällen wird auf das Beenden etwaiger noch laufender Aktivitäten gewartet. Im Gegensatz dazu befindet sich ein Prozess im Zustand *Running*, solange atomare Aktivitäten des Prozesses ausgeführt werden. Der Zustand jeder einzelnen Aktivität kann hierbei durch das Zustandsmodell einer Aktivität ausgedrückt werden, welches (hier in vereinfachter Form) ebenfalls auf dem entsprechenden Zustandsmodell von LEYMANN und ROLLER basiert [LR00, KZL06, KZL07b] (vgl. Abschnitt 2.7.2). Das Zustandsmodell einer Aktivität (*Activity State Model*) ist im unteren rechten Teil

von Abbildung 6.6 dargestellt. Als Verknüpfung beider Zustandsmodelle kann zusammenfassend ausgedrückt werden, dass der Prozess nur den Zustand *Option* annehmen kann, wenn sich temporär keine Aktivitäten im Zustand *Executing* befinden. Die Konsistenz des Prozesszustands zum Zeitpunkt der Migration ist somit im Rahmen dieses Modells gewährleistet.

Wie weiter oben erläutert wurde, befinden sich auch die Prozessvariablen zum Zeitpunkt einer (potentiellen) Migration jeweils in einem konsistenten Zustand. Die privaten Daten des Prozesses werden daher als aktuelle Werte der Variablen (*Current Value*) als Instanzdaten des Prozesses bei einer Migration übertragen. Die Angabe von Datentypen ist hier nicht notwendig, da diese bereits (i.d.R. unveränderlich) im regulären Prozessmodell enthalten bzw. implizit durch die Ein- und Ausgabedaten der Aktivitäten definiert sind. Für eine verteilte Ausführung von parallelen Prozesspfaden ist es jedoch vorteilhaft angeben zu können, wie das Synchronisationsverhalten der Variablen bei einem parallelen Zugriff von mehreren Ausführungseinheiten definiert sein soll. Dazu gibt die Einordnung der Variablen in eine *Datenklasse* (*Data Class*) an, ob eine Synchronisation des Variablenwerts unmittelbar vor bzw. nach dem Zugriff auf die Variable durchgeführt werden muss oder ob eine Verzögerung der Synchronisation (z. B. im Rahmen der Offline-Bearbeitung von Prozesspfaden) in Hinblick auf die Korrektheit der (parallelen) Ausführung hinnehmbar ist [ZKML10]. Eine detaillierte Betrachtung der Parallelausführung von migrierbaren Prozessen erfolgt in Abschnitt 6.2.5.

Ergänzend zu den Zuständen der einzelnen Aktivitäten und Variablen wird schließlich der aktuelle Bearbeitungszustand des Prozesses in den Migrationsdaten dokumentiert. Da der Prozess nur zwischen der Ausführung atomarer Aktivitäten migriert werden kann, wird bei einer Migration der Zeiger auf die als nächstes auszuführende Aktivität des Prozesses ausgewertet und diese als Startaktivität (*StartActivity*) vermerkt. Eine Startaktivität verweist dazu auf die eindeutige *ID* der zu referenzierenden Aktivität. Da im Fall einer Ausführung von parallelen Prozesspfaden mehrere Zeiger auf Folgeaktivitäten existieren, können auch mehrere Startaktivitäten definiert werden, an denen die Prozessausführung nach erfolgter Migration (sequentiell oder parallel) fortgesetzt werden kann [ZKML10, ZHKK10]. Die Definition von Startaktivitäten besitzt daher insbesondere im Kontext verteilt parallel ausgeführter Prozesspartitionen eine große Relevanz (vgl. Abschnitt 6.2.5).

Die bis hierhin erläuterten Migrationsdaten können prinzipiell automatisch generiert und mit den auf das minimale Prozessmetamodell abgebildeten Elementen bestehender Prozessmodelle verknüpft werden (vgl. Schritt 2 in Abbildung 6.4). Für eine solche *Startkonfiguration* wird die abstrakte Repräsentation des betrachteten Prozesses in den Zustand *Created* und alle Aktivitäten jeweils in den Zustand *Inactive* versetzt. Falls Variablen mit einem initialen Wert belegt wurden, werden diese als aktuelle Werte der Variablen vermerkt. Als Standardeinstellung können zudem alle Variablen als unmittelbar zu synchronisierend gekennzeichnet werden. Die erste Aktivität des Prozesses stellt schließlich automatisch die Startaktivität dar [ZKML10, ZHKK10].

Befindet sich der Prozess bereits in Ausführung (vgl. Schritt 5b in Abbildung 6.5) so müssen anstelle der genannten Startkonfiguration die entsprechend aktuellen Werte der Prozessausführung eingesetzt werden. Die Umsetzung der Migration zur Laufzeit des Prozesses ist Inhalt von Kapitel 7.5.

Benutzerdefinierte Vorgaben

In vielen Fällen sind Prozessmodellierer, Initiator des Prozesses oder Prozessteilnehmer daran interessiert, die Art und Weise der Verteilung eines Prozesses aktiv zu beeinflussen. Die beiden wichtigsten Aspekte einer Verteilungskonfiguration betreffen die *Granularität* und den *Zielort* der Verteilung (vgl. Abschnitt 4.3.2). Es stellt sich also die Frage, welcher Teil des Kontrollflusses des Prozesses unter welchen Bedingungen von welcher Ausführungseinheit verwaltet werden soll. Da die Antwort auf diese Frage von verschiedenen (anwendungsabhängigen) Aspekten beeinflusst sein kann, spezifiziert das hier vorgestellte Metamodell das erweiterbare Element *Assignment Strategy*, welches sowohl auf der Ebene des globalen Prozesses als auch auf der Ebene einzelner (strukturierter oder atomarer) Aktivitäten die Verteilungsstrategie des Benutzers abbilden kann (vgl. Abbildung 6.6).

Abhängig davon, ob der Prozessmodellierer, der Initiator des Prozesses oder ein Prozessteilnehmer auf die Verteilung des Prozesses einwirken möchten, kann der Inhalt des Elements *Assignment Strategy* und die konkret zu verfolgende Verteilungsstrategie zur Entwicklungszeit des Prozessmodells, bei dessen Instantiierung oder zur Laufzeit der Prozessinstanz festgelegt werden. Unter dem Zielaspekt der Flexibilisierung können daher auch bestehende Verteilungsstrategien – bei Vorliegen der entsprechenden Berechtigungen – im Laufe der Prozessausführung angepasst und erweitert werden. Die aktuelle Verteilungsstrategie überschreibt dabei die jeweils vorhergehende Verteilungsstrategie. Basierend auf den vorgestellten Fallbeispielen (vgl. Abschnitt 4.2) werden in dieser Arbeit fünf grundlegende Varianten für die Spezifikation benutzerdefinierter Vorgaben vorgeschlagen [ZKML10, ZHKK10]:

- **Undefinierte Zuordnung (*Undefined*):** Im einfachsten Fall und in der Standardeinstellung ist *keine Verteilungsstrategie* vorgegeben. Diese Option impliziert ein Höchstmaß an Flexibilität für die jeweils verantwortliche (lokale) Ausführungseinheit, da diese hierbei ohne Einschränkungen jederzeit über das gesamte Verteilungsspektrum des Prozesses entscheiden kann. Zum Beispiel kann hiermit die Prozessausführung jederzeit auf eine andere Prozess-Engine verlagert werden, um dort verfügbare Ressourcen zu nutzen oder die Auslastung der aktuellen Ausführungseinheit zu verringern. Auf der anderen Seite erfolgt die Ausführung des Prozesses aus der Sicht des Prozessmodellierers bzw. -initiators weitestgehend unkontrolliert. Eine undefinierte Zuordnung wird zum Beispiel im einfachsten Fall einer kontextbasierten Kooperation verwendet, wo durch eine ungerichtete Migration beliebige Ausführungseinheiten mit geeigneten Ressourcen für die als nächstes

auszuführende Aktivität des Prozesses aufgefunden werden können (vgl. Abschnitt 4.2.4).

- **Feste oder rollenbasierte Zuordnung (*Fixed Participant or Role*):** Hierbei ist es möglich, die verantwortliche Ausführungseinheit auf der Basis eines (bestehenden) rollenbasierten Organisationsmodells zu bestimmen. Ähnlich zur Zuweisung von (menschlichen) Ressourcen zu einzelnen Aufgaben eines Prozesses kann hierbei entweder eine konkrete Entität angegeben werden (z. B. eine bestimmte Person, Organisation oder Prozess-Engine) oder die Verantwortlichkeit kann einem frei wählbaren Individuum innerhalb einer bestimmten Gruppe solcher Entitäten übertragen werden (z. B. einer beliebigen Ausführungseinheit der Rolle „Kooperationspartner“). Die Flexibilität ist hierdurch in beiden Fällen relativ stark eingeschränkt, da eine Verteilung nur noch im Rahmen der Vorgaben vorgenommen werden darf. Eine feste oder rollenbasierte Zuordnung ist insbesondere in Hinblick auf Sicherheitsaspekte der verteilten Prozessausführung vorzunehmen. Im Extremfall kann der gesamte Prozess einer einzigen Ausführungseinheit zugewiesen werden, was einer vollkommen zentralen Ausführung entspricht.

- **Regel- oder algorithmenbasierte Zuordnung (*Rule or Algorithm*):** Soll der Zielort der Verteilung das Ergebnis einer Regelauswertung oder einer (komplexeren) Berechnung sein, so kann eine Regel oder ein Algorithmus angegeben werden, welcher die entsprechende technische Logik hierfür implementiert. In dieser Arbeit wird zwischen zwei Arten von Berechnungen unterschieden:

  - *Prozessbezogene Berechnungen* verwenden nur Informationen, welche ihnen im Ausführungskontext der betrachteten Prozessinstanz direkt zur Verfügung stehen, d. h. Informationen über das Prozessmodell, die Prozessdaten und Log-Informationen. Ein einfaches Beispiel ist die Formulierung der Forderung, dass Aktivität x von derselben Ausführungseinheit verwaltet werden soll, welche auch Aktivität y ausgeführt hat (vgl. [BD00]). In technischer Hinsicht sind prozessbezogene Berechnungen insbesondere für die Absicherung einer asynchronen Kommunikation auf Anwendungsebene relevant (vgl. Abbildung 2.24 in Abschnitt 2.7.3), um eine korrekte Zuordnung des Nachrichtenflusses zwischen den aufgerufenen Ressourcen und der lokal verantwortlichen Ausführungseinheit zu ermöglichen.

  - *Kontextbezogene Berechnungen* können auch allgemeine Informationen der Ausführungseinheiten zur Berechnung von geeigneten Zielorten der Verteilung einbeziehen und somit auch komplexere Berechnungen abbilden. Ein Beispiel hierfür ist die Auswahl von Ausführungseinheiten, welche aufgrund ihrer Ressourcen möglichst große Partitionen des Prozesses an einem Stück ausführen können,

um die Anzahl der notwendigen Migrationen insgesamt möglichst gering zu halten (vgl. [Gol09, MZL10, ZML11]).

Regeln oder Berechnungsanweisungen zur Auswahl von Ausführungseinheiten können prinzipiell als Anlage spezifiziert (d. h. mit dem Prozess migriert) oder entfernt referenziert werden (d. h. bereits auf dem Ausführungssystem vorhanden sein). Berechnungsanweisungen in der Anlage des Prozesses erlauben eine große Flexibilität bei der Definition neuer und für den einzelnen Anwendungsfall spezialisierter Zuordnungsverfahren. Dafür erfordern diese jedoch auch die Festlegung einer gemeinsamen Beschreibungssprache, welche die Definition von Regeln oder Algorithmen in maschinenunabhängiger Form und deren semantisch äquivalente Interpretation auf ggf. unterschiedlichen Plattformen ermöglicht. Die Referenz auf bestehende Berechnungsverfahren ist weniger aufwendig und erlaubt eine individuelle Implementierung von Algorithmen. Die Anwendbarkeit ist in diesem Fall jedoch auf die auf den lokalen Ausführungssystemen tatsächlich zur Verfügung stehenden Algorithmen begrenzt.

- **Variablenbasierte Zuordnung (*Variable*):** In einigen Fällen ist der Zielort der Verteilung Teil der fachlichen Anwendungslogik und ist bereits mehr oder weniger konkret in den Prozessdaten beschrieben. Die Verteilungsstrategie *Variable* gibt an, dass der Zielort der Verteilung für eine bestimmte Prozesspartition als Wert einer Prozessvariablen spezifiziert ist, welche zum Zeitpunkt der Verteilung ausgewertet werden soll. Dies ermöglicht es, den Zielort der Verteilung erst während der Prozessausführung unter fachlichen Kriterien bestimmen zu lassen. Ein Beispiel hierfür ist der Anwendungsfall *eErasmus* aus Abschnitt 4.2.2, wo die Gast-Universität erst auf Basis der Studienergebnisse des Studierenden ausgewählt wird. Die variablenbasierte Verteilungsstrategie stellt somit eine Vereinfachung der *prozessbezogenen Berechnungen* mit einem fachlichen Hintergrund dar. Als Spezialfall dieser Strategie können Verteilungsinformationen auch statisch als Initialwert einer Variablen im Prozessmodell spezifiziert oder bei der Instantiierung des Prozesses als Aufrufparameter übergeben werden. Die Flexibilität in Hinblick auf die Verteilung wird hierdurch natürlich entsprechend eingeschränkt.

- **Nicht-funktionale und kontextbasierte Zuordnung (*QoS and Context*):** Ebenso wie die variablenbasierte Zuordnung eine Vereinfachung der prozessbezogenen Auswahlalgorithmen darstellt, erlaubt die Zuordnung auf Basis von nicht-funktionalen Aspekten und Kontextparametern eine einfache Auswahl von Ausführungseinheiten auf Basis ihrer aktuellen Eigenschaften. Einfache Beispiele sind durch die Bevorzugung von Prozess-Engines mit der aktuell geringsten Auslastung oder die Auswahl von Ausführungssystemen an einem bestimmten geographischen Ort gegeben. Für komplexere Zusammenhänge und Kombinationen von Eigenschaften können Sprachen zur Beschreibung nicht-funktionaler Charak-

teristiken (*NFCs*) und nicht-funktionaler Anforderungen (*NFAs*) und entsprechende generische Matchmaking-Algorithmen zum Einsatz kommen. Ein Beispiel für eine solche Sprache und die entsprechende Auswahl von geeigneten Dienstanbietern für die Ausführung von Prozessen in dynamischen Umgebungen wurde in einer weiterführenden Arbeit [HSZ11] vorgestellt).

Ist auf der Ebene der einzelnen (atomaren) Aktivität keine Verteilungsstrategie angegeben (*undefinierte Zuordnung*), so gilt die Verteilungsstrategie des jeweils übergeordneten Gültigkeitsraums (d. h. auf der Ebene übergeordneter strukturierter Aktivitäten oder auf der Prozessebene). In den anderen Fällen überschreibt eine Verteilungsstrategie auf Aktivitätsebene eine etwaige übergeordnete Strategie auf Prozessebene. Dies erlaubt Kombinationen und Ausnahmeregelungen, wie zum Beispiel für den Fall, dass alle Ausführungseinheiten aufgrund des nicht-funktionalen Kriteriums K ausgewählt werden sollen, die Ausführung von Aktivität A jedoch in jedem Fall von der konkreten Organisation O durchgeführt werden soll.

Für weitere Verwaltungsaufgaben kann der Prozessmodellierer bzw. -initiator zudem festlegen, welche zusätzlichen Daten während der Ausführung des Prozesses gesammelt werden sollen, um eine Fehlerbehandlung und/oder eine spätere Nachvollziehbarkeit der Prozessausführung zu erlauben. Ein wichtiges Beispiel hierfür ist die Aufzeichnung von ausreichenden Informationen darüber, welche Ausführungseinheit zu welcher Zeit für welche Prozesspartition verantwortlich gewesen ist. Zudem können benutzerdefinierte Daten über die Ausführung gesammelt werden, z. B. an welchem geographischen Ort die Ausführung einer Aktivität stattgefunden hat (vgl. [ZL10] für Details). Die erforderlichen Informationen können bei Bedarf optional in einem *Log* auf Ebene der (atomaren) Aktivitäten gespeichert werden, welches in der Standardeinstellung von allen Ausführungseinheiten eingesehen oder von einer entfernten Verwaltungskomponente abgefragt werden kann (vgl. Abschnitt 6.3). Es kann somit sowohl zur Laufzeit des Prozesses als auch nach dessen Terminierung zur Optimierung der aktuellen bzw. nachfolgenden Prozessausführung herangezogen werden.

Schließlich ist außerdem die Festlegung benutzerdefinierter Sicherheitsmechanismen im Kontext migrierender Prozesse ein sehr wichtiger Aspekt. Das hier vorgestellte Migrationsmodell enthält daher die Möglichkeit zur Spezifikation von Sicherheitsrichtlinien (*Security Policies*), welche optional mit den Elementen des minimalen Prozessmetamodells (d. h. mit dem gesamten Prozess, einzelnen atomaren oder strukturierten Aktivitäten sowie einzelnen Prozessdaten) verknüpft werden können. Eine detaillierte Betrachtung der prozessbezogenen Sicherheitsrichtlinien sowie die Konzeption grundlegender Sicherheitsmechanismen erfolgt in Abschnitt 6.2.7.

6.2.3 Ressourcenzuordnung für migrierte Prozesse

Im Allgemeinen muss bei einer Code-Migration (vgl. Abschnitt 3.9.4) geprüft werden, ob die vom Programm benötigten Ressourcen bei einer Migration verschoben, kopiert, ersetzt oder durch einen systemweiten Verweis zugänglich gemacht werden können [FPV98, TS08]. Im Kontext prozessorientierter Anwendungen handelt es sich bei solchen Ressourcen aufgrund des höheren Abstraktionsgrades in der Regel um Personen, Maschinen oder Softwarekomponenten (bzw. Gruppen von Personen, Maschinen oder Softwarekomponenten), welche zur Ausführung der Aktivitäten des Prozesses vorgesehen sind (vgl. Abschnitt 2.1). Ob eine Neuzuordnung dieser Ressourcen bei einer Migration des Prozesses notwendig bzw. möglich ist, hängt dabei von der Art der Ressource, von der im Prozessmodell verwendeten Art ihrer Beschreibung und von der Kompatibilität der Organisationsmodelle der an der Prozessausführung beteiligten Ausführungsumgebungen ab.

Um eine direkte Ausführung eines Prozesses zu ermöglichen, muss bei einem technischen Prozessmodell der Verweis auf eine Ressource zumindest ein automatisiertes Auffinden bzw. eine automatisierte Auswahl einer geeigneten Ressource erlauben. Dies kann in Anlehnung an FUGGETTA et al. [FPV98] dabei entweder über einen global eindeutigen Identifikator der Ressource bzw. einer Angabe deren Zugriffsortes (z. B. über eine URI bzw. URL) geschehen (*starke Bindung*), oder durch die Verarbeitung einer maschinenlesbaren Beschreibung bestimmter Eigenschaften erfolgen, welche zur Erfüllung der jeweiligen Aktivität relevant sind (*schwache Bindung*). Beispiele für schwache Bindungen sind Rollenbeziehungen bei der Zuweisung menschlicher Prozessteilnehmer oder die Angabe von Ontologien und nicht funktionalen Eigenschaften zur Auswahl von elektronischen Diensten (vgl. Abschnitte 2.7.3 und 3.4). Die schwache Bindung erlaubt beim Vorliegen entsprechender Auswahlmechanismen eine höhere Flexibilität beim Zugriff auf Ressourcen. Eine Unterspezifikation in Form von gänzlich fehlenden Zuordnungen ist auf der Ebene direkt ausführbarer Prozesse jedoch nicht möglich, da dies eine (uneingeschränkte) Automatisierung der Prozessausführung verhindert.

Folglich kann prinzipiell eine von einem Prozess auf Anwendungsebene benötigte Ressource bei einer Migration des Prozesses entweder über die bereits im Prozessmodell spezifizierte Referenz erhalten bleiben oder nach erfolgter Migration neu gebunden werden (*Rebinding*). Eine weitere Möglichkeit besteht darin, die vom Prozess benötigte Ressource bei einer Migration des Prozesses zu verschieben oder zu kopieren, so dass sie in der Zielausführungsumgebung ebenfalls zur Verfügung steht. Im Folgenden werden diese drei Möglichkeiten und die Konsequenzen für die Migration von laufenden Prozessinstanzen aufgezeigt.

Ressourcen mit starker Bindung

Liegen in einem zu verteilenden Prozess eindeutige Referenzen zu Ressourcen vor, welche für die Ausführung einer bestimmten Aktivität aufgerufen wer-

den sollen, so behalten diese Referenzen nach einer Migration des Prozesses weiterhin Gültigkeit, wenn die Ressourcen in der Zielausführungsumgebung ebenfalls erreichbar sind. Vor allem Dienste in dienstorientierten Architekturen (vgl. Abschnitt 3.4), welche über das Internet global erreichbar sind (z. B. *Web Services*, vgl. Abschnitt 3.4.3) und in der Prozessbeschreibung durch eine URL eindeutig spezifiziert werden, können nach einer Migration des Prozesses durch die aktuelle Ausführungseinheit aufgerufen werden, sofern diese Zugang zum Internet und die Berechtigung zum Aufruf der Ressourcen besitzt. Die zu erreichende Flexibilität bezieht sich in diesem Fall jedoch nur auf einen möglichen Austausch der Ausführungseinheit und nicht auf einen Austausch der durch den Prozess aufgerufenen Ressourcen, da diese in dem betrachteten Fall fest vorgegeben sind.

Ist die Verfügbarkeit einer benötigten Ressource für die Zielausführungseinheit nur eingeschränkt möglich, ist im Einzelfall ein *Rebinding* möglich. Eine mögliche Vorgehenweise hierfür ist der bereits erläuterte Ansatz von MOSER, ROSENBERG und DUSTAR [MRD08] (vgl. Abschnitt 5.7.8). Da jedoch bei Ressourcen mit starker Bindung in der Regel nicht ersichtlich ist, ob aus fachlicher Sicht genau die Identität der spezifizierten Ressource für die Prozessausführung wesentlich ist oder eine Bindung nach Wert bzw. Typ ausreichend wäre, ist es notwendig, die referenzierte Ressource bei Bedarf als *substituierbar* zu kennzeichnen [MRD08]. Da sich insbesondere elektronische Dienste in der Regel durch eine hohe Wiederverwendbarkeit auszeichnen und auch für andere Konsumenten zur Verfügung stehen müssen, ist auch ein Verschieben einer benötigten Ressource bei einer Migration des Prozesses oft nicht sinnvoll. Ebenso ist ein Kopieren von Ressourcen oft mit Schwierigkeiten behaftet, da in diesem Fall neben der Kompatibilität der Prozessausführungsumgebung auch eine geeignete Ausführungsplattform für die transferierten Anwendungen mit ihren spezifischen Implementierungen bestehen muss. Das Verschieben oder Kopieren von Ressourcen ist zudem oft mit einem unverhältnismäßig hohen Aufwand verbunden (z. B. bei umfangreichen Ressourcen wie Datenbanken oder komplexen physikalischen Ressourcen wie Maschinen [TS08]). Die Kopierbarkeit von Ressourcen ist zudem in einigen Fällen gänzlich ausgeschlossen, z. B. bei menschlichen Prozessteilnehmern. Eine aus fachlicher Sicht korrekte Ausführung ist jedoch in jedem Fall gegeben, sofern genau die konkret angegebene Ressource aufgerufen wird. Bei einer eingeschränkten Verfügbarkeit einer Ressource mit starker Bindung ist es daher vorteilhaft, die Verteilung der Prozessausführung an die lokale Verfügbarkeit der benötigten Ressource anzupassen und den Prozess entsprechend an eine Ausführungseinheit zu migrieren, die Zugriff auf die aktuell benötigte (konkret spezifizierte) Ressource hat. Die Migration des Prozesses (bzw. ihre Einschränkung) ist somit unter Umständen die einzige praktikable Möglichkeit zum Zugriff auf eine (mobile) Ressourcen. In vielen Fällen werden also nicht die Ressourcen aufgrund der Migration des Prozesses verschoben, sondern die Verschiebung einer fest vorgegeben Ressource stellt den *Auslöser der Migration* dar.

Ressourcen mit schwacher Bindung

Ressourcen mit schwacher Bindung müssen in der Regel einem bestimmten Typ entsprechen, welcher bei der Nutzung einer Ressource dieses Typs gewährleistet, dass ein beabsichtigtes funktionales Ergebnis erzielt werden kann. Hierbei steht also die Funktionalität der Ressource im Vordergrund und nicht deren konkrete Identität. Die Autoren TANENBAUM und STEEN nennen als allgemeines Beispiel den Verweis auf lokale Geräte wie z. B. einen Drucker [TS08]. Durch schwache Bindung referenzierte Ressourcen können daher nach einer Migration des Prozesses durch lokal verfügbare Ressourcen mit gleichen oder vorteilhafteren Eigenschaften ersetzt werden. Voraussetzung für die automatische Ersetzbarkeit einer Ressource ist neben dieser gemeinsamen Semantik das Vorliegen einer festgelegten Syntax zum Aufruf der Ressource, zur Übergabe von Parametern und zur Entgegennahme von etwaigen Ergebnissen [KZL07a, ZKL09b].

Die Auswahl von geeigneten Ressourcen wird dabei zudem durch die von der Ausführungseinheit unterstützten Technologien zur Integration der Ressourcen durch entsprechende Adapter eingeschränkt [Kun08]. Ein Beispiel ist der unterschiedliche Zugriff auf Softwarefunktionalitäten, welche über eine CORBA- oder eine Web-Service-Schnittstelle angeboten werden (vgl. Abschnitt 3.4.3), da die Schnittstellenbeschreibungen jeweils unterschiedlich verarbeitet werden müssen. Wird ein entsprechend hoher Abstraktionsgrad der Ressourcenbindung in der Prozessbeschreibung unterstützt, so lassen sich durch die Migration eines Prozesses neue Ausführungskontexte erschließen, da aufgrund einer alternativen Menge der durch die Ausführungseinheit unterstützten Technologien zum Aufruf von Ressourcen auch das Spektrum erreichbarer Ressourcen vergrößert werden kann [KZL07a, KZTL08]. Ein Beispiel für die Spezifikation von Ressourcen auf einem derart hohen Abstraktionsniveau ist die Angabe von *abstrakten Dienstklassen* [KZL07a, Kun08] (vgl. Abschnitt 5.5.5). Die Verwendung abstrakter Dienstklassen am Beispiel von Web Services (*WSDL*) und CORBA (*IDL*) ist in Abbildung 6.7 dargestellt.

Für schwach gebundene Ressourcen wird durch die unterstützende Middleware ein (ggf. technologieunabhängiger) Mechanismus zum Auffinden von geeigneten Ressourcen sowie – bei Vorliegen mehrerer potentiell geeigneter Ressourcen – ein Mechanismus zu deren Auswahl benötigt. Bei einer Migration von Prozessen und einer späten Bindung von Ressourcen (d. h. einer Bindung erst zum Zeitpunkt der Aktivierung der betreffenden Aktivität oder kurz vor deren Ausführung) kann hierfür – analog zur nicht verteilten Ausführung des Prozesses – jeweils die lokale Komponente der verantwortlichen Ausführungsumgebung zur Identifikation und zum Aufruf aktuell geeigneter Ressourcen verwendet werden. Werden bei einer frühen Bindung Ressourcen zur Laufzeit des Prozesses reserviert, so gilt entsprechend die Vorgehensweise für stark gebundene Ressourcen. In jedem Fall ist hierbei jedoch im Rahmen der Vorgaben in den Migrationsdaten (vgl. Abschnitt 6.2.2) eine

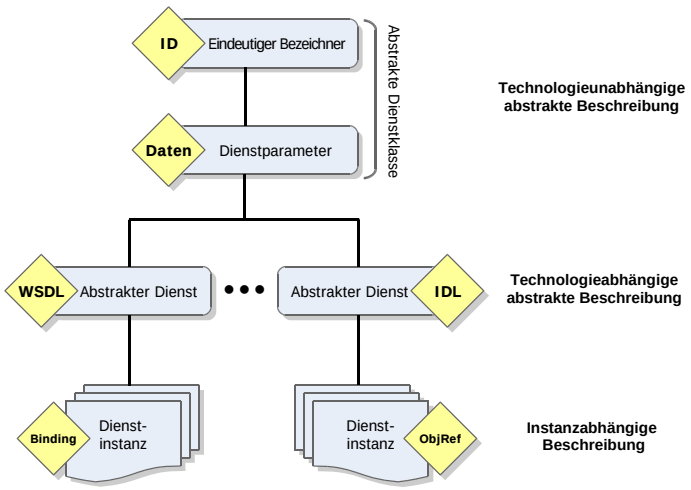


Abbildung 6.7: Beispiel: Verwendung abstrakter Dienstklassen (KZL07a, Kun08)

spätere Umleitung der auszuführenden Aktivität an eine andere Ressource möglich (vgl. hierzu auch Abschnitt 2.7.3).

Der automatisierte Austausch von schwach gebundenen Ressourcen über mehrere Ausführungseinheiten hinweg setzt in jedem Fall die Verwendung einer gemeinsamen Ontologie bzw. das Vorhandensein kompatibler Organisationsmodelle oder übergeordneter Rollenauflösungskomponenten voraus. Eine Migration laufender Prozessinstanzen mit Austausch schwach referenzierter Ressourcen ist daher vornehmlich für Systeme mit einer hinreichend gemeinsamen semantischen Interpretation der domänenspezifischen Anwendungsvorgänge geeignet.

Migration von Ressourcen

In einigen Fällen kann es sinnvoll sein, Ressourcen gemeinsam mit der laufenden Prozessinstanz in eine andere Ausführungsumgebung zu verschieben. Dies setzt voraus, dass die Ressourcen an ihrem Ursprungsort bei einer Migration des Prozesses nicht mehr anderweitig benötigt werden oder problemlos kopiert und transferiert werden können. Ein typisches Beispiel hierfür sind einfache prozessspezifische Programmfragmente wie zum Beispiel Skripte oder Anweisungen zur Darstellung von Benutzeroberflächen, welche beim Deployment eines Prozesses inbegriffen sind und in erster Linie der Ausführung dieses Prozesses dienen. Es besteht hierbei in der Regel eine starke wechselseitige Kopplung zwischen Prozessmodell und Programmfragment.

Die Migration von Prozessen mit weiteren Programmfragmenten, welche in einer anderen (Programmier-)Sprache verfasst sind als die Prozessbeschreibung selbst, macht eine kompatible Ausführungsumgebung auf dem Zielsystem erforderlich, welche diese zusätzlichen Programmfragmente verarbeiten kann. Während die Interpretation und Navigation des Prozesses nach fest vorgegebenen Regeln auf Basis der verwendeten Prozessbeschreibung erfolgt und

durch die Laufzeitumgebung der verarbeitenden Prozess-Engine in einer Art *Sandbox* abläuft, können die Ausführung weiterer Prozessfragmente beliebigen Inhalts und deren Auswirkungen auf das lokale System in der Regel nur schwer kontrolliert werden. Die Ausführung von Aktivitäten, welche derartige Ressourcen benötigen, sollte daher nach Möglichkeit auf der Ursprungsplattform des Prozesses durchgeführt werden oder auf Systeme mit einem ausreichenden Vertrauensverhältnis untereinander begrenzt sein. Eine Migration des Prozesses (inklusive der zu migrierenden Ressourcen) außerhalb dieser Aktivitäten ist jedoch prinzipiell möglich, sofern die migrierten Ressourcen erst bei der Ausführung der betreffenden Aktivität installiert werden müssen. In diesem Fall können diese mit der laufenden Prozessinstanz weitergereicht werden, bis sie konkret benötigt werden. Eine automatische Zuweisung der Prozessausführung zu einer geeigneten Ausführungseinheit kann dabei über die Angabe von Vorgaben in den Migrationsdaten (z. B. über eine feste oder rollenbasierte Zuordnung) geschehen (vgl. Abschnitt 6.2.2).

Auch Anweisungen zur Repräsentation von Benutzungsschnittstellen werden oft bereits beim Deployment des Prozesses integriert. Beispiele sind HTML-Formulare, welche zur Prozesslaufzeit durch eine Rendering-Komponente eingelesen und entsprechend dargestellt werden können. Im Gegensatz zu den zuvor genannten beliebigen Programmfragmenten besteht hierbei ein deutlich geringeres Sicherheitsrisiko für das lokale System. Es besteht jedoch das Problem, dass zur Entwicklungszeit festgelegte Beschreibungen zur Repräsentation von Benutzungsschnittstellen auf dem während der Ausführung des Prozesses verwendeten Ausgabegerät nicht adäquat angezeigt werden können (vgl. Abschnitt 5.8). Da die Ausführung von benutzerzentrischen Aufgaben in Zusammenhang mit der Mobilität von menschlichen Prozessteilnehmern jedoch ein wesentliches Ziel der Verteilung von prozessorientierten Anwendungen ist (vgl. Abschnitt 4.3.1), ist eine Einschränkung der Prozessmigration aus diesem Grund nicht ohne weiteres hinnehmbar. Eine erweiterte Lösung zur Integration von abstrakten Benutzungsschnittstellen für verteilt ausgeführte Prozesse wird daher gesondert in Abschnitt 6.4 vorgestellt.

6.2.4 Verteilung sequentiell ausgeführter Prozesse

Geht man zunächst von einem rein sequentiellen Kontrollfluss eines dynamisch durch Migration verteilten Prozesses aus, so erfolgt aus lokaler Sicht der einzelnen Ausführungseinheiten trotz der Verteilung stets eine (für die Dauer der Verantwortlichkeit) zentrale Ausführung des Prozesses. Dies bedeutet, dass zur Laufzeit des Prozesses bei einem rein sequentiellen Kontrollfluss jeweils genau eine Ausführungseinheit für die Navigation des Kontrollflusses, den Aufruf von Ressourcen zur Durchführung der definierten Aktivitäten sowie für die Aufzeichnung relevanter Logdaten verantwortlich ist. Da die Ausführung des Prozesses bis zum Zeitpunkt einer Migration äquivalent zu einer zentralen Ausführung des Prozesses verläuft, muss sie an dieser Stelle nicht gesondert betrachtet werden.

Die aktuell verantwortliche Ausführungseinheit (bzw. eine unterstützende Komponente) ist jedoch darüber hinaus im Rahmen der in dieser Arbeit vorgestellten Konzepte Entscheidungsträger für den Transfer der Prozessbeschreibung an andere Ausführungseinheiten. Hierfür muss auf Basis der benutzerdefinierten Rahmenbedingungen und der ggf. bestehenden lokalen Richtlinien der betreffenden Ausführungseinheit entschieden werden, an welcher Position im Kontrollfluss die lokale Prozessausführung angehalten und an welche andere Ausführungseinheit die Prozessbeschreibung zusammen mit den aktuellen Migrationsdaten versendet werden soll. Für Durchführung von Migrationen im Rahmen eines zur Laufzeit des Prozesses sequentiellen Kontrollflusses werden daher zunächst die in Abschnitt 2.4 aus den Arbeiten von VAN DER AALST und den weiterführenden Arbeiten anderer Autoren zusammengestellten abstrakten Kontrollflusskonstrukte untersucht. Das Vorliegen von entsprechenden benutzerdefinierten Vorgaben zur Durchführung einer Migration an der jeweiligen Position im Kontrollfluss wird dabei an dieser Stelle vorausgesetzt.

Ein Spezialfall stellt die Migration des Prozesses dar, ohne im Kontrollfluss des Prozesses vorangeschritten zu sein. Dies erlaubt zum Beispiel das Weiterreichen der Prozessinstanz, falls benötigte Ressourcen für die Ausführung anstehender Aktivitäten nicht zur Verfügung stehen oder die lokale Ausführungseinheit bereits ausgelastet ist. Auch bei technischen Problemen, z. B. dem Vorliegen einer nicht unterstützten Prozessbeschreibungssprache oder einer nicht unterstützten Version von Programmfragmenten kann die betroffene Komponente als Zwischenknoten fungieren und den Prozess an eine geeignetere Ausführungsumgebung weiterleiten.

Einfache Sequenzen

Bei einer einfachen *Sequenz* von Aktivitäten (vgl. Abschnitt 2.4.1) kann durch das Festschreiben der aktuell zu betrachtenden Aktivität (*Start Activity*) und der aktuellen Werte der Prozessvariablen eine Migration der Prozessinstanz vorgenommen werden, sobald ein konsistenter Zustand erreicht wird. Sind die beiden atomaren Aktivitäten A_1 und A_2 sequentiell angeordnet, so dass A_2 erst nach der Beendigung von A_1 gestartet werden darf, so ist dies nach den in Abschnitt 6.2.2 definierten Bedingungen genau dann der Fall, wenn die Ausführung von A_1 beendet wurde, die Ausführung von A_2 aber noch nicht begonnen wurde – also sich nach dem Zustandsmodell für Aktivitäten (*Activity State Model*) keine der beiden Aktivitäten im Zustand *Executing* befindet. Dabei macht es keinen Unterschied, ob die betrachtete Sequenz atomarer Aktivitäten direkt auf der globalen Ebene des Prozesses angeordnet ist oder sich innerhalb eines bestimmten lokalen Gültigkeitsbereichs befindet. Da sich strukturierte Aktivitäten zum Zeitpunkt der Migration in einem beliebigen Zustand befinden dürfen, ist lediglich der Zustand der atomaren Aktivitäten ausschlaggebend, um über eine Migration zu entscheiden [KZL06, KZL07b, ZK07].

Abbildung 6.8 zeigt einen verfeinerten Überblick über die prinzipiell möglichen Migrationszeitpunkte innerhalb einer sequentiellen Ausführung

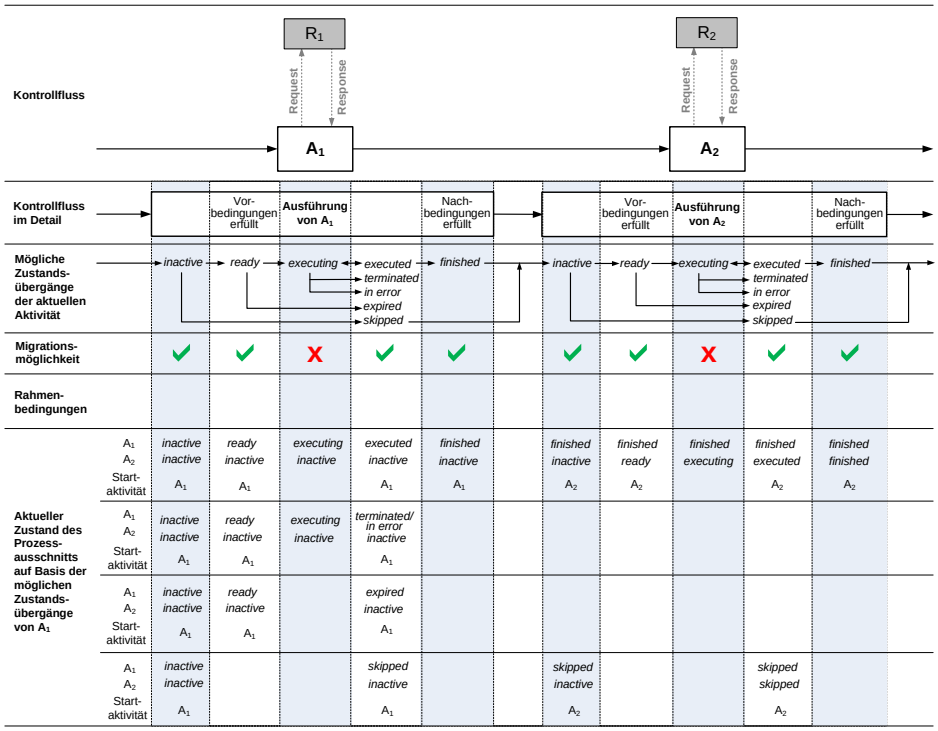


Abbildung 6.8: Migrierbarkeit innerhalb eines Kontrollflusses zweier sequentiell ausgeführter Aktivitäten A_1 und A_2 mit jeweils synchronem Aufruf der Ressourcen R_1 und R_2

der zuvor genannten Aktivitäten mit einem jeweils abgeschlossenen Ressourcenaufruf. Dabei ist insbesondere zu erkennen, dass der Prozess migriert werden kann, wenn relevante Vorbedingungen für die Ausführung einer Aktivität in der aktuellen lokalen Ausführungsumgebung nicht erfüllt sind (z. B. wenn die benötigte Ressource nicht zugegriffen werden kann). Der Prozess kann zudem migriert werden, wenn relevante Nachbedingungen der Aktivitätsausführung nicht erfüllt werden (z. B. wenn während der Ausführung der Aktivität ein Fehler aufgetreten ist oder nicht-funktionale Aspekte der Ausführung verletzt wurden). Die Ausführung der Aktivität kann nach diesem Modell also durch die Migration des Prozesses in einer anderen Ausführungsumgebung wiederholt werden. Die etwaige Rücksetzung von Daten oder Zuständen ist dabei – analog einer nicht-verteilten Ausführung – von der für die Ausführung der Aktivität verantwortlichen Prozess-Engine durchzuführen und wäre auch im nicht-verteilten Fall für eine wiederholte Ausführung der Aktivität notwendig.

Eine wichtige Einschränkung der Migrierbarkeit ergibt sich jedoch für zwei Aktivitäten A_1 und A_2, welche bei einer asynchronen Kommunikation mit einer Ressource durch das Senden einer Nachricht durch A_1 (*One-Way-Request*) und das Erwarten einer Antwortnachricht durch A_2 (*Solicit-Response*) in Zusammenhang stehen (vgl. Abschnitt 2.7.3 und [KL06]), wobei A_1 und A_2 nicht zwingend direkt aufeinander folgen müssen. Werden die Ausführungen von

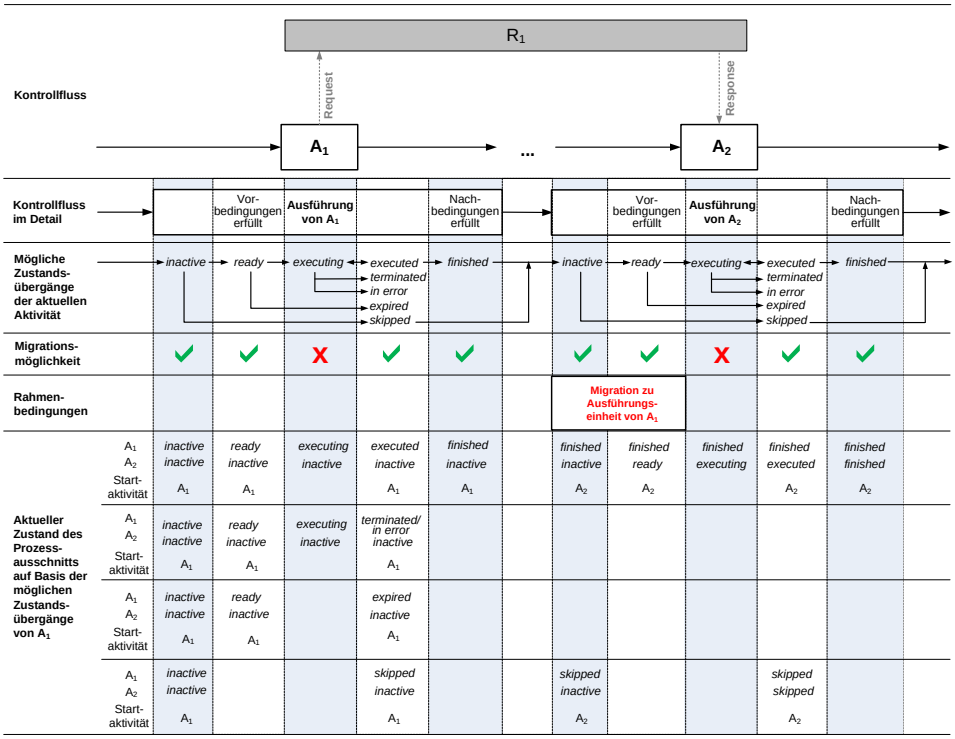


Abbildung 6.9: Migrierbarkeit innerhalb eines Kontrollflusses zweier sequentiell ausgeführter Aktivitäten A_1 und A_2 mit asynchronem Aufruf der Ressource R_1

A_1 und A_2 durch eine zwischenzeitliche Migration von zwei unterschiedlichen Ausführungseinheiten verwaltet, so wird die Zuordnung der Antwortnachricht für A_2 erschwert, da der Ressource die aktuelle Adresse für die Antwortnachricht nicht bekannt sein kann. Die Antwortnachricht muss daher entweder von der für A_1 verantwortlichen Ausführungsumgebung an die aktuelle Ausführungsumgebung weitergeleitet werden (was bei mehrfacher Migration auch eine entsprechend mehrfache Weiterleitung erforderlich machen kann) oder die Verteilung des Prozesses muss für die Ausführung von A_2 so eingeschränkt werden, dass die Ausführungen von A_1 und A_2 von derselben Ausführungseinheit verwaltet werden. Eine automatische Migration zu der dynamisch festgelegten Ausführungseinheit von A_1 kann im Rahmen der Vorgaben im Migrationsmodell durch eine prozessbezogene Regel bzw. Berechnungsanweisung ermöglicht werden. Diese Variante ist für die genannten Aktivitäten mit asynchronem Aufruf einer Ressource in Abbildung 6.8 visualisiert.

Kontrollflussbedingungen

Einen wichtigen Untersuchungsgegenstand in Hinblick auf die Migration von laufenden Prozessinstanzen nimmt die Behandlung von Kontrollflussbedingungen ein. Hierzu gehören im Wesentlichen *Transitionsbedingungen*, *Split*- und *Join*-Bedingungen sowie Schleifenbedingungen (vgl. Abschnitte 2.4.1 und 2.4.2). Wie in den zuvor genannten Abschnitten gezeigt wurde, können Be-

dingungen je nach Prozessbeschreibungssprache als jeweils eigene Art von Kontrollflusselement, als eigene (ggf. strukturierte) Aktivität oder als Teil einer (ggf. strukturierten) Aktivität abgebildet werden. Die mit der Auswertung von Bedingungen verbundenen Teilaufgaben des Einlesens von Variablenwerten sowie das Berechnen und Festlegen einer entsprechenden Reaktion in Bezug auf die Aktivierung nachfolgender Aktivitäten können in ihrer Gesamtheit daher als einzelner Prozessschritt aufgefasst werden, welcher analog zu den fachlichen Aktivitäten des Prozesses nicht durch eine zwischenzeitliche Migration des Prozesses zerteilt werden darf. Sieht man die Auswertung einer einzelnen Bedingung daher als eine durch die Prozess-Engine auszuführende Aufgabe, so kann man eine Kontrollflussbedingung als eine spezielle Art von Aktivität auffassen, welche durch einen ausschließlich lesenden Zugriff auf Prozess- oder Umgebungsdaten als Berechnungsergebnis entsprechende Zustandsänderungen der im Kontrollfluss direkt nachfolgenden Aktivitäten auslöst. Solange die Berechnung dieser Zustände andauert, befindet sich die betreffende „Auswertungsaktivität“ selbst im Zustand *Executing*. Der Prozess kann mit dieser Sichtweise somit sowohl *vor* als auch *nach* der Auswertung der Bedingung migriert werden, da der Zeiger auf die als nächstes auszuführende(n) Aktivität(en) bzw. auf die als nächstes auszuwertende Kontrollflussbedingung zum Zeitpunkt der Migration eindeutig definiert ist.

Ein weiterer Vorteil der Auffassung einer Kontrollflussbedingungen als eigene Aktivität besteht darin, dass im Rahmen des Migrationsmodells und der benutzerdefinierten Rahmenbedingungen für die Verteilung des Prozesses die Auswertung einer Bedingung gezielt einer geeigneten Ausführungseinheit zugeordnet werden kann. Diese Zuordnung kann unter Umständen erhebliche Auswirkungen auf die fachliche Prozessausführung haben. So würde zum Beispiel die Auswertung einer Zeitbedingung von zwei Ausführungseinheiten, welche sich aufgrund ihrer geographischen Position in unterschiedlichen Zeitzonen befinden, unter Umständen zu unterschiedlichen Auswertungsergebnissen und somit zu fachlich unterschiedlichen Kontrollflussentscheidungen führen. Verallgemeinert kann festgehalten werden, dass hinsichtlich einer Zuordnung von Kontrollflussbedingungen individuelle Auswertungen abgeleitet werden können, sobald die Bedingung nicht alleine auf der Auswertung von internen Prozessvariablen oder -konstanten basiert, sondern Umgebungsdaten integriert, welche von Ausführungseinheit zu Ausführungseinheit variieren können. Neben Zeitbedingungen kann dies alle relevanten Kontextdaten des Prozesses, wie zum Beispiel Wetter und Temperatur, die geographische Position oder sogar lokale Gesetze und Marktbedingungen umfassen (vgl. Abschnitt 3.5.2). Werden über die Anbindung eines lokalen Geschäftsregel-Managementsystems Entscheidungen getroffen (vgl. Abschnitt 3.6.2), so sind diese ebenfalls durch Regelwerke der lokalen Ausführungsumgebung determiniert. Die Möglichkeit, einzelne Kontrollflussbedingungen zu Ausführungseinheiten zuordnen, stellt damit einen wichtigen Flexibilitätsaspekt dar, denn die Flexibilität der Zuordnung kann dabei

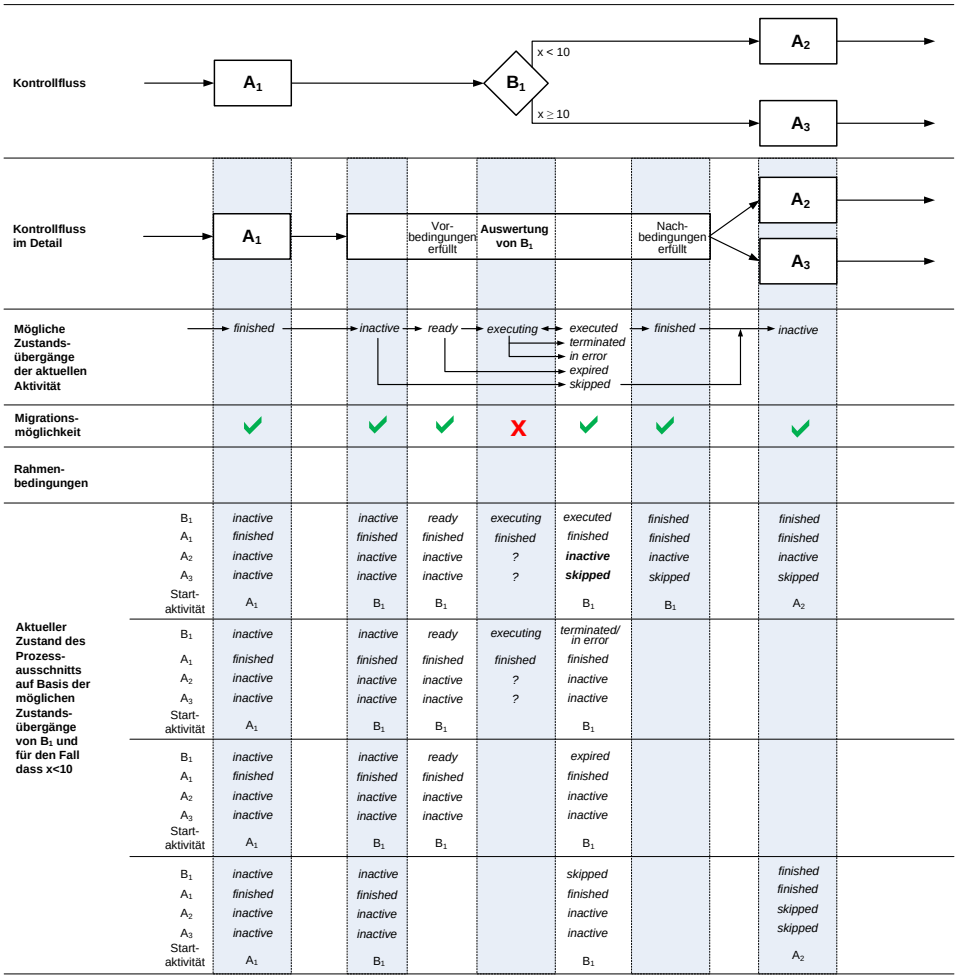


Abbildung 6.10: Migrierbarkeit innerhalb eines sequentiellen Kontrollflusses mit *Split*-Bedingung B_1 (Beispiel für $x < 10$)

über die Granularität der Verteilung in Bezug auf rein fachliche Aktivitäten hinaus erhöht werden.

Befindet sich eine Bedingung an einer Kontrollflussverzweigung (*Split*-Bedingung), so wird durch deren Auswertung der weitere Kontrollfluss des Prozesses festgelegt. Dabei kann die *Split*-Bedingung aus mehreren Einzelbedingungen zusammengesetzt sein, welche besagen, unter welchen Umständen ein ausgehender Pfad aktiviert werden darf. Um eine insgesamt konsistente Kontrollflussentscheidung treffen zu können, dürfen diese Einzelbedingungen jedoch nicht verteilt ausgewertet werden. Eine Split-Bedingung wird somit im Rahmen der Migration von Prozessen insgesamt als atomare Aktivität aufgefasst. Bei einem *XOR-Split* oder den Varianten des *OR-Splits*, bei welchen aufgrund der Auswertung der Kontrollflussbedingungen zur Laufzeit nur ein einziger ausgehender Pfad aktiviert wird, wird die Ausführung des Prozesses auf diesem Pfad sequentiell fortgesetzt. Die Zustände der Aktivitäten auf dem nicht aktivierten Pfad werden im Rahmen der *Dead Path Elimination* auf

den Zustand *Skipped* gesetzt. Da damit alle relevanten Informationen zur Auswertung einer etwaigen *Join*-Bedingung am Ende der verzweigten Ausführung vorliegen, kann prinzipiell auch diese als individuell zuzuordnende Aktivität betrachtet werden. Im Fall einer zur Laufzeit sequentiellen Ausführung kann der *Join* jedoch in der Regel vernachlässigt werden, da er ohne Synchronisation mehrerer eingehender Pfade nur eine inhaltslose Aktivität zur Weiterleitung des Kontrollflusses an die nächste fachliche Aktivität oder Kontrollflussbedingung darstellt.

Abbildung 6.10 zeigt die Migrierbarkeit eines verzweigenden Kontrollflusses mit einer *XOR-Split*-Bedingung an einem Beispiel. Der komplexere Fall des *AND-Splits* mit paralleler Ausführung von Aktivitäten und anschließender Synchronisation ist Inhalt von Abschnitt 6.2.5.

Schleifen

Strukturierte Schleifen besitzen in der Regel ebenfalls eine Bedingung, welche (abhängig von der Art der Schleife) festlegt, ob der Schleifenrumpf (erneut) durchlaufen werden soll (vgl. Anschnitt 2.4.2). Das Bedingungskonstrukt entspricht somit einem XOR-Split und kann im Rahmen der oben dargestellten Auffassung als eigene Aktivität einer geeigneten Ausführungseinheit zugewiesen werden, sofern alle für die Schleifenbedingung notwendigen Daten entweder als interne Prozessvariablen, als (Prozess-)Konstanten oder als Umgebungsdaten der Ausführungseinheit vorliegen. In diesem Fall ist eine Migration des Prozesses – unter den gegebenen Rahmenbedingungen der Migration – auch jederzeit während des Durchlaufens des Schleifenrumpfes möglich, da von jeder Ausführungsumgebung lokal entschieden werden kann, ob die Schleife erneut durchlaufen werden soll. Problematisch ist jedoch der Einsatz eines Durchlaufzählers, sofern dieser nicht explizit als Variable des Prozesses modelliert ist, sondern nur implizit von der Ausführungseinheit verwaltet wird. Ein Beispiel ist die einfache Anweisung, dass die Schleife x-mal durchlaufen werden soll. Hierbei muss bei einer Migration die Anzahl der bereits durchgeführten Schleifendurchläufe an die nächste Ausführungseinheit übergeben werden. Vermeidet man die Einführung expliziter Zählvariablen als Hilfskonstrukt für die Migration, so ist eine Rekonstruktion des aktuellen Ausführungszustandes der Schleife nur mit Hilfe einer Logdatei möglich [ZL10].

Unstrukturierte Schleifen (vgl. Anschnitt 2.4.2) sind nicht explizit als Schleifen ausgezeichnet und besitzen somit auch keine Semantik für implizite Durchlaufzähler. Sie stellen daher in Bezug auf eine zwischenzeitliche Migration kein Problem dar und können wie eine bedingte sequentielle Ausführung behandelt werden.

Blockaktivitäten und Subprozesse

Blockaktivitäten stellen Platzhalter für einen in sich abgeschlossenen strukturierten Kontrollfluss dar, welcher anstelle der Blockaktivität ausgeführt wer-

den soll (vgl. Abschnitt 2.4.3). Da es sich hierbei um eine strukturierte Aktivität im Rahmen des Migrationsmodells handelt (vgl. Abschnitt 6.2.2), kann eine Migration prinzipiell sowohl unmittelbar vor oder nach der Ausführung des Blocks vorgenommen werden, aber auch zwischen den atomaren Aktivitäten innerhalb des enthaltenen Kontrollflusses geschehen. Da die Beschreibung des Blocks stets Inhalt der migrierten Prozessbeschreibung ist, kann der Block unter Berücksichtigung der benutzerdefinierten Rahmenbedingungen von einer beliebigen (geeigneten) Ausführungseinheit fortgesetzt werden. Die Blockaktivität nimmt dabei den Zustand *Executing* an (vgl. Abbildung 6.11).

Anders verhält es sich bei der Integration von Subprozessen, da diese einen autonomen Prozess *außerhalb* der migrierten Prozessbeschreibung referenzieren können (vgl. Abschnitt 2.4.3). Vergleichbar mit einer atomaren Aktivität, welche den (lokalen oder entfernten) Zugriff einer bestimmten Ressource zu ihrer Ausführung erfordert, ist für den Aufruf eines Subprozesses ein Zugriffspunkt erforderlich, über den eine neue Instanz des Subprozesses erzeugt und gestartet werden kann. Für eine Migration werden hierbei für die Behandlung der unterschiedlichen Arten der Integration von Subprozessen zwei Fälle unterschieden:

- **Interner Subprozess mit enger Kopplung:** Der aufrufende Prozess und der Subprozess sind zusammen Teil eines gemeinsamen übergeordneten Pakets, welches im Rahmen der Prozessbeschreibung als Ganzes migriert wird. Die Prozessbeschreibung des Subprozesses liegt somit allen beteiligten Ausführungseinheiten zur Laufzeit vor, kann bei Bedarf gestartet und wie der aufrufende Prozess bei Bedarf angehalten und gemeinsam mit dem aufrufenden Prozess migriert werden.
  - Bei einer einfachen *Verkettung* des übergeordneten Prozesses und des (internen) Subprozesses wird die Ausführung des Subprozesses lediglich angestoßen (vgl. Abbildung 2.11b in Abschnitt 2.4.3). Sind aufrufender Prozess und Subprozess sequentiell zueinander angeordnet, so tritt die Ausführung des Subprozesses an die Stelle der Ausführung des aufrufenden Prozesses. Der Subprozess kann im Rahmen der benutzerdefinierten Vorgaben gemeinsam mit dem übergeordneten Paket und etwaigen historischen Daten des aufrufenden Prozesses weiter migriert werden. Sind aufrufender Prozess und Subprozess parallel zueinander angeordnet, können die in Abschnitt 6.2.5 beschriebenen Ansätze für eine verteilte parallele Ausführung angewendet werden. Der Subprozess kann somit auch durch eine andere Ausführungsumgebung ausgeführt werden wie der aufrufende Prozess und auch selbst wieder migriert werden.
  - Eine *synchrone Ausführung* des übergeordneten Prozesses und des (internen) Subprozesses entsprechen weitestgehend der oben genannten Blockaktivität, denn die Ausführung des Subprozesses tritt temporär an die Stelle der Ausführung des aufrufenden Prozesses (vgl. Abbildung 2.11c in Abschnitt 2.4.3). Der Subprozess kann daher

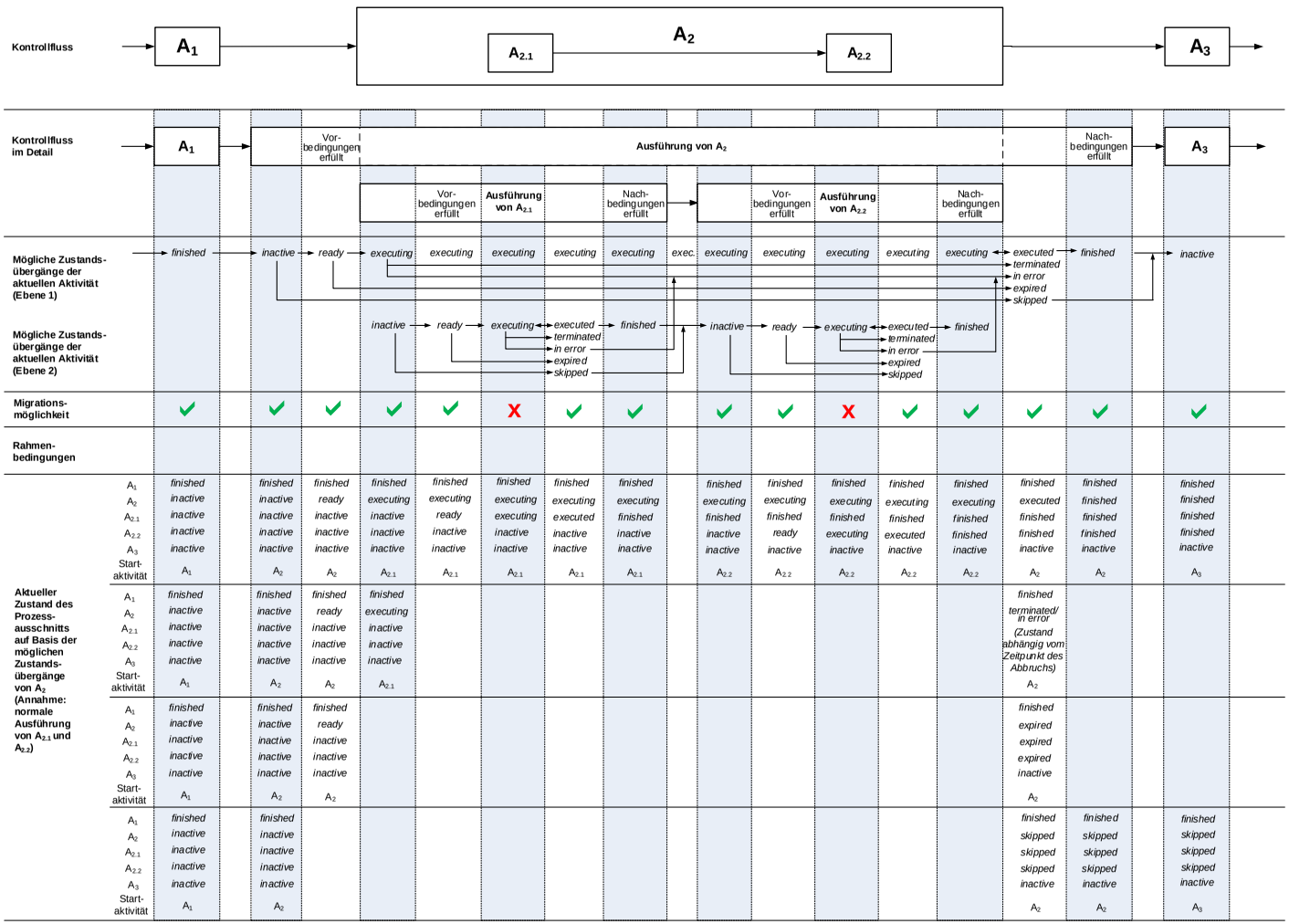


Abbildung 6.11: Migrierbarkeit innerhalb einer Blockaktivität A_2 mit den internen Aktivitäten $A_{2.1}$ und $A_{2.2}$

im Rahmen der benutzerdefinierten Vorgaben gemeinsam mit dem übergeordneten Paket und dem pausierten aufrufenden Prozess und dessen aktuellem Zustand weiter migriert werden. Im Gegensatz zu einer einfachen sequentiellen Verkettung müssen jedoch stets beide Prozesse auf der jeweils verantwortlichen Ausführungseinheit installiert werden, da der aufrufende Prozess nach Abschluss des Subprozesses fortgesetzt werden muss.

- ▷ Eine *asynchrone Ausführung* des übergeordneten Prozesses und des Subprozesses erfordert, dass beide Prozesse parallel zueinander ausgeführt werden (vgl. Abbildung 2.11d in Abschnitt 2.4.3). Es können daher weitestgehend die in Abschnitt 6.2.5 beschriebenen Ansätze für eine verteilte parallele Ausführung angewendet werden. Der Subprozess kann somit auch durch eine andere Ausführungsumgebung ausgeführt werden wie der aufrufende Prozess und auch selbst wieder migriert werden. Im Gegensatz zu der Ausführung paralleler Prozesspfade eines einzelnen Prozesses kann es jedoch in der Regel nicht zu Datenabhängigkeiten kommen, da jeder Prozess seine eigenen internen Prozessvariablen verwaltet (vgl. Abschnitt 6.2.5).

- ► **Externer Subprozess mit loser Kopplung:** Der aufrufende Prozess und der Subprozess sind *nicht* Teil eines gemeinsamen übergeordneten Pakets und werden nicht zusammen migriert. Der Aufruf des Subprozesses gleicht somit aus der Sicht der lokalen Ausführungsumgebung dem Aufruf einer beliebigen Ressource, welche ihrerseits einen Prozess ausführt (dienstorientierte Sichtweise). Da der für die Ausführung des aufrufenden Prozesses zuständigen Ausführungseinheit der Subprozess nicht zwingend bekannt sein muss, kann dieser auch nicht in eine etwaige Migration einbezogen werden. Unabhängig davon ist jedoch – bei Vorliegen der notwendigen Voraussetzungen – eine eigenständige Migration des Subprozesses während seiner Ausführung denkbar. In diesem Fall ist es für asynchron ausgeführte Prozesse notwendig, dass der aufrufende Prozess zur Übergabe eines etwaigen Berechnungsergebnisses später wieder aufgefunden werden kann. Dies erfordert ggf. die Festlegung eines geeigneten Synchronisationspunktes (vgl. Abschnitt 6.2.5). Für synchron ausgeführte Subprozesse ist dies nicht notwendig, da die Ausführung des aufrufenden Prozesses während der Bearbeitung des Subprozesses nicht weiter voranschreiten kann. Die den Subprozess aufrufende Aktivität befindet sich also während der gesamten Ausführung des Subprozesses im Zustand *Executing*. Da die Ausführung des Subprozesses für die lokale Ausführungseinheit transparent erfolgt, ist die aufrufende Aktivität für die Ausführungseinheit atomar und somit wird im Rahmen der vorgestellten Konzepte eine Migration des aufrufenden Prozesses verhindert, solange der Subprozess ausgeführt wird.

Fehlerbehandlungsmaßnahmen

Speziell als Fehlerbehandlungsmaßnahmen ausgewiesene Aktivitäten, welche im Verlauf eines sequentiellen Kontrollflusses aufgrund einer von der lokalen Ausführungsumgebung erkannten fachlichen oder technischen Ausnahmesituation aktiviert werden, können prinzipiell ganz oder teilweise an andere Ausführungseinheiten delegiert werden, indem sie auf die bereits oben beschriebenen Blockaktivitäten zurückgeführt werden (vgl. Abschnitt 2.4.5). Wird durch die aktuell verantwortliche Ausführungsumgebung ein Fehler innerhalb eines definierten Gültigkeitsbereichs festgestellt, so wird die entsprechend übergeordnete Blockaktivität nach dem Zustandsmodell für Aktivitäten in den Zustand *inError* versetzt (vgl. Abschnitt 6.2.2) und stattdessen mit der definierten Fehlerbehandlungsroutine fortgefahren. Eine Migration ist daher zu jedem erlaubten Zeitpunkt der entstehenden (sequentiellen) Ausführung möglich (vgl. Abbildung 6.12).

Einen in Bezug auf die Migration interessanteren Aspekt stellt die Gewährleistung von transaktionalem Verhalten zwischen mehreren Aktivitäten sowie die Umsetzung von Maßnahmen zur ggf. notwendigen *Backward-Recovery* dar. Sofern sequentielle Aktivitäten im Sinne einer Transaktion als zusammenhängend gekennzeichnet sind (vgl. z. B. [Hol07, Kun08, ZKL09a]) und die Ausführung der Transaktion aufgrund einer Ausnahmesituation fehlschlägt, ist eine Rücksetzung bzw. eine Kompensation der bereits abgeschlossenen Aktivitäten notwendig. Findet während der Ausführung transaktional zusammenhängender Aktivitäten eine Migration des Prozesses statt, so werden Änderungen an Prozessdaten oder externe Effekte über mehrere Ausführungseinheiten hinweg sichtbar. Eine Kompensation dieser Auswirkungen kann dabei ggf. von der aktuell verantwortlichen Ausführungseinheit durchgeführt werden oder muss an die lokale Ausführungseinheit delegiert werden, welche auch für die Ausführung der entsprechenden Aktivität verantwortlich gewesen ist. Da diese Zuordnung anwendungsspezifisch ist oder von den technischen bzw. organisatorischen Gegebenheiten der teilnehmenden Ausführungseinheiten abhängig sein kann, kann mit dem hier vorgestellten Migrationsmetamodell sowohl eine statische Zuweisung der Kompensationsaktivitäten zur Entwicklungszeit als auch eine dynamische Zuweisung zur Laufzeit unterstützt werden. Es können insbesondere die folgenden Varianten ermöglicht werden:

- **Keine vorgegebene Zuordnung:** Die Kompensation einer oder mehrerer Aktivitäten kann von einer beliebigen (geeigneten) Ausführungseinheit durchgeführt werden. Dies ist insbesondere der Fall, wenn es sich bei der Kompensation um den Aufruf von global erreichbaren Diensten mit in den Zustandsdaten des Prozesses vorliegenden Datenwerten handelt. Ein Beispiel ist die Stornierung einer zuvor aufgegebenen Bestellung bei einem Online-Shop durch Übergabe der Bestellnummer (als Variable des Prozesses) an eine Operation einer vorgegebenen Dienstschnittstelle. Hierbei ist nur die Wahl der richtigen

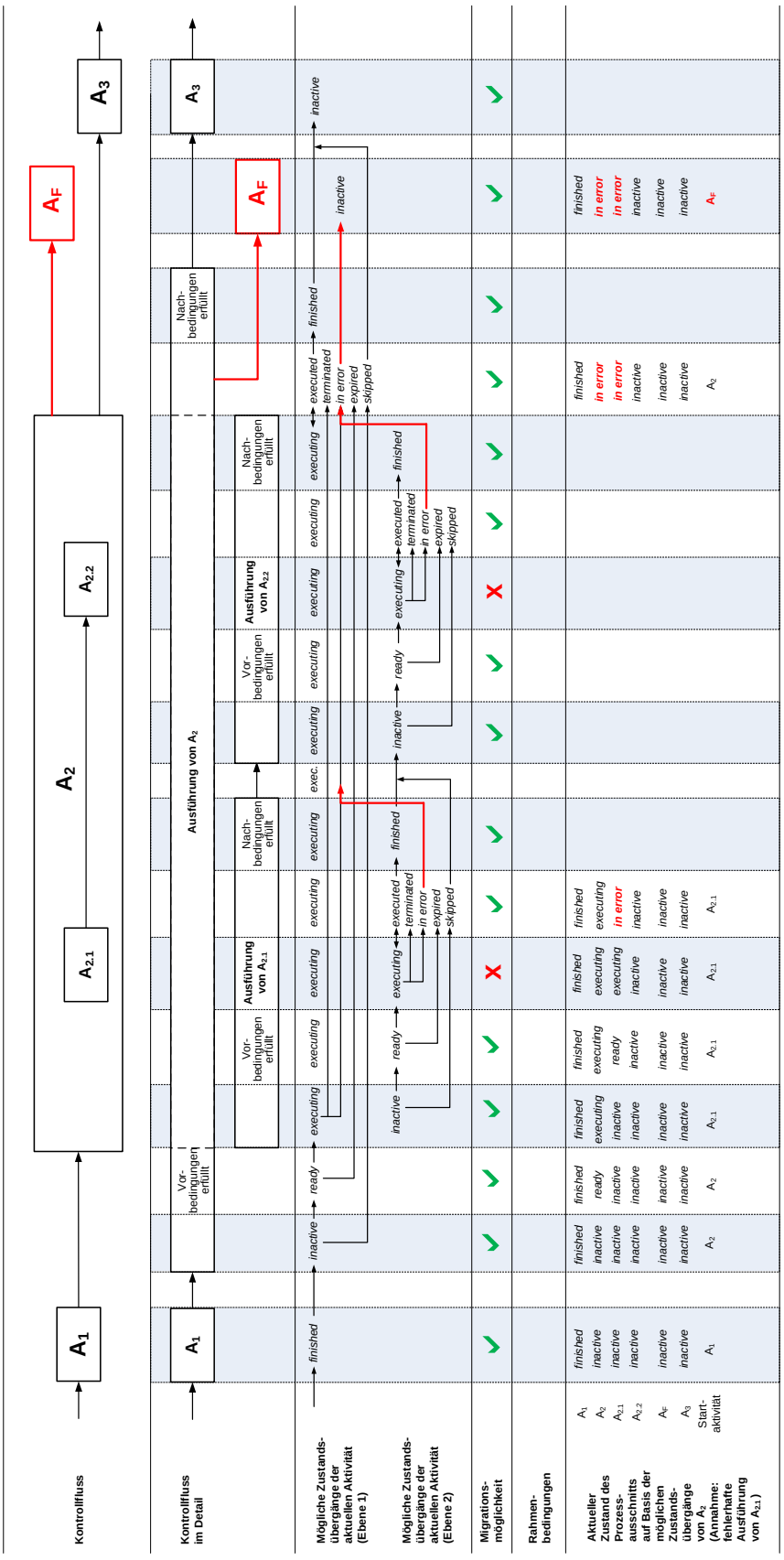


Abbildung 6.12: Migrierbarkeit innerhalb eines Kontrollflusses mit Fehlerbehandlungsmaßnahme A_F für die Aktivitäten im Gültigkeitsraum der Blockaktivität A_2 (Beispiel: Fehlerfall in $A_{2.1}$)

Ressource für die Rücksetzung relevant, nicht aber die Wahl der verantwortlichen Ausführungsumgebung. Die Flexibilität der Verteilung wird hierbei daher nicht eingeschränkt.

- **Feste Zuordnung zwischen Aktivitäten und Kompensationsaktivitäten:** Die Kompensation einer oder mehrerer Aktivitäten kann nur von genau derjenigen lokalen Ausführungseinheit durchgeführt werden, welche auch für die Ausführung der entsprechenden Ursprungsaktivität(en) verantwortlich gewesen ist. Dies ist zum Beispiel der Fall, wenn die für die Kompensation benötigten Ressourcen nur lokal verfügbar sind oder sich lokal individuell unterscheiden. Um eine statische Zuordnung im Voraus zu vermeiden und damit die Flexibilität nicht unnötig einzuschränken, kann zur Ausführungszeit einer zur Transaktion gehörenden Aktivität eine neue Zuordnung erzeugt werden, welche die Kompensationsaktivität der (dynamisch wählbaren) Ausführungseinheit der Ursprungsaktivität zuweist. Im Fall einer Rücksetzung wird somit automatisch eine Migration zu dieser Ausführungseinheit erzwungen. Ein Beispiel für diese Variante ist in Abbildung 6.13 dargestellt.

- **Kontextbasierte Zuordnung zwischen Aktivität und Kompensationsaktivität:** In einigen Fällen ist für eine Kompensation nicht eine konkrete Ausführungsumgebung relevant, sondern es muss ein bestimmter Kontext vorliegen, welcher auch der Ausführung der Ursprungsaktivitäten zugrunde lag. Ein Beispiel ist die Ausführung von Aktivitäten an einem bestimmten Ort durch das mobile Gerät eines Außendienstmitarbeiters. Die Kompensation dieser Aktivitäten darf in diesem Fall eventuell durchaus von einer anderen Ausführungseinheit (d. h. von einem anderen mobilen Endgerät) und/oder von einer anderen Ressource (d. h. von anderem anderen Mitarbeiter), nicht aber an einem anderen Ort ausgeführt werden. Eine statische Zuordnung ist daher nicht möglich, sondern es muss aufgrund der aktuellen Kontextdaten eine Neuzuordnung vorgenommen werden. Dazu muss in den Migrationsdaten explizit eine entsprechende Auswahlstrategie festgehalten werden, z. B. als kontextbasierte Zuordnung (vgl. Abschnitt 6.2.2).

- **Statische Zuordnung:** Die Kompensationsaktivität ist in diesem Fall (z. B. aufgrund von benutzerdefinierten Rahmenbedingungen) einer konkreten Ausführungseinheit fest zugeordnet. Im Fall einer Rücksetzung wird eine Migration zur vorgegebenen Ausführungseinheit initiiert.

Die Durchführung von Kompensationsaktivitäten kann im Rahmen der genannten Varianten gezielt gesteuert werden, erfordert jedoch aufgrund der potentiellen Anwendungsabhängigkeit der Verteilung unter Umständen die Spezifikation von benutzerdefinierten Rahmenbedingungen. Zusätzlich können die zu jedem Zeitpunkt zugewiesenen Ausführungseinheiten, die Zustände von Prozessdaten sowie das Auftreten von Ausnahmesituationen und fehlgeschlagenen Transaktionen in den Log-Informationen vermerkt werden, um hieraus

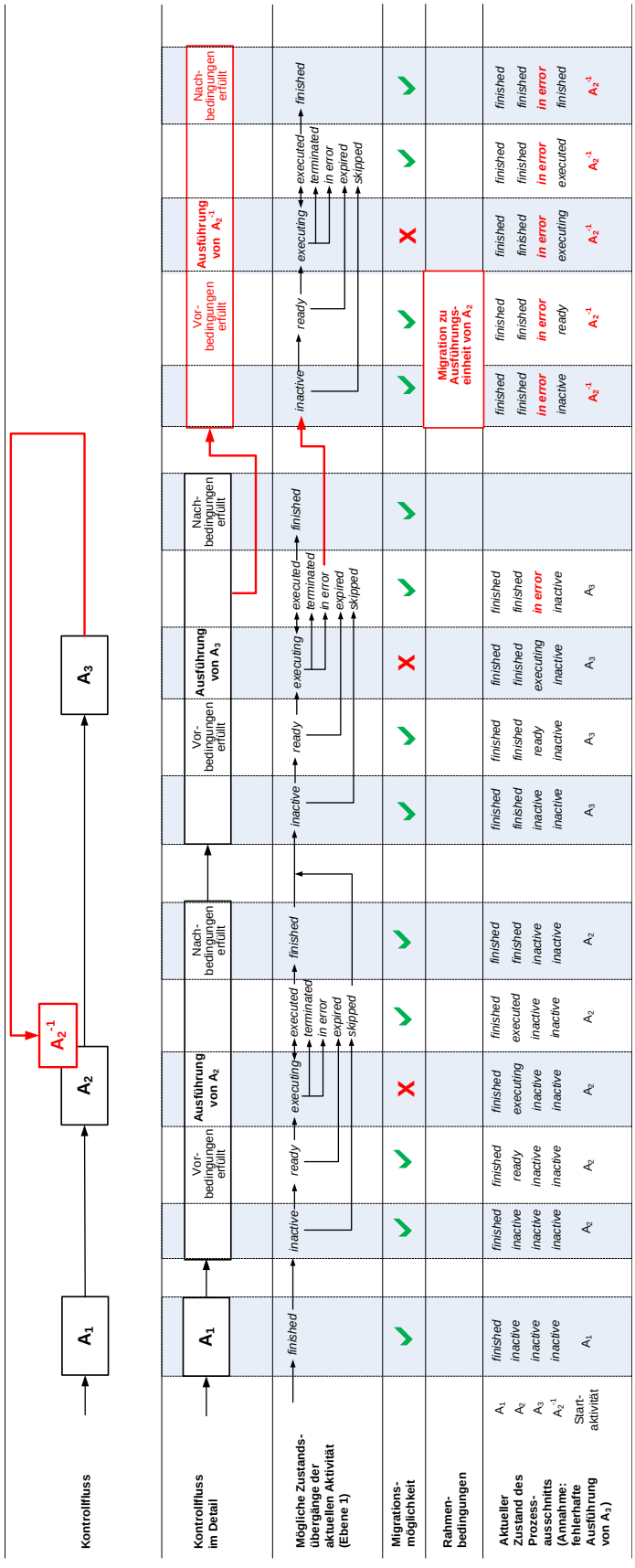


Abbildung 6.13: Migrierbarkeit innerhalb eines Kontrollflusses mit Kompensation A_2^{-1} von Aktivität A_2 (Beispiel: Fehlerfall in A_3): Feste Zuordnung zwischen Aktivität und Kompensationsaktivität (Beispiel)

bei Bedarf einen früheren Prozesszustand wiederherzustellen. Dies ist z. B. auch für den besonderen Fall notwendig, dass Informationen zur konkreten Kompensation erst bei der Durchführung einer Aktivität bekannt werden und bis zur etwaigen Kompensation zwischengespeichert werden müssen (vgl. hierzu auch [CR04]).

6.2.5 Verteilung und Synchronisation paralleler Prozesspfade

Wie im vorangegangenen Abschnitt gezeigt wurde, kann der Koordinationsaufwand für die Verteilung eines Prozesses im Fall einer sequentiellen Ausführung auf eine relativ einfache Delegation der Prozessausführung begrenzt werden. Solange ein laufender Prozess jeweils nur von genau einer Ausführungseinheit verwaltet wird, ist auch die Ausführung von parallelen Prozesspfaden innerhalb der Prozessbeschreibung unkritisch, da der Zustand der Prozessausführung bei jeder Migration stets wohldefiniert ist und Datenabhängigkeiten zwischen Prozessvariablen verschiedener paralleler Ausführungspfade durch die lokale Prozess-Engine bzw. der Transaktionsverwaltung der lokalen Datenbank behandelt werden können.

In vielen Fällen stellt aber gerade eine echte Parallelausführung im Sinne einer Ausführung von parallelen Prozesspartitionen durch verschiedene Ausführungseinheiten oder Organisationen ein wesentliches Ziel der Verteilung von prozessorientierten Anwendungen dar (vgl. Abschnitt 4.3.1). Im Rahmen der Migration von Prozessen lassen sich hierfür insbesondere zwei generelle Möglichkeiten identifizieren:

Zum einen kann der Prozess in seine parallelen Pfade partitioniert und die entstehenden physisch getrennten Partitionen können an die ausgewählten Ausführungseinheiten verteilt werden. Diese Lösung ist relativ unflexibel, da eine Migration des Prozesses dann nur noch innerhalb der einmal festgelegten Fragmente möglich ist. Zudem müssten die erforderlichen Synchronisationspunkte im Voraus festgelegt und zugeordnet werden, was eine weitere Verteilung des Prozesses zu seiner Laufzeit erschwert. Datenabhängigkeiten zwischen Prozessvariablen verschiedener paralleler Ausführungspfade sind auf diese Weise nur schwer zu erkennen, da den Ausführungseinheiten der einzelnen Partitionen Informationen über die Verwendung der Prozessvariablen in anderen Partitionen fehlen.

Zum anderen ist es für eine Verteilung paralleler Prozesspfade möglich, die gesamte Prozessbeschreibung zu replizieren und die entstehenden Replikate den zur Verfügung stehenden Ausführungseinheiten so zuzuordnen, dass jeder gewählte Teilnehmer für die Fortführung einer bestimmten Anzahl paralleler Pfade verantwortlich ist. Während der Ausführung der einzelnen Replikate können diese hierbei wiederum an andere Ausführungseinheiten migriert werden. Dieser Ansatz entspricht eher der hier gewählten Vorgehensweise einer logischen Partitionierung und vereinfacht die etwaige Synchronisation von Prozessvariablen, da jedem Teilnehmer prinzipiell alle Informationen über die Verwendung der Prozessvariablen auf anderen Pfaden zur Verfügung stehen

[ZKML10]. In beiden Fällen müssen jedoch Vorkehrungen für die geordnete Zusammenführung der parallelen Pfade getroffen werden, sofern dies im Prozessmodell gefordert wird. Für den allgemeinen Fall nach VAN DER AALST [vdATHKB03] betrifft dies die Kontrollflussstrukturen *AND-Join*, *OR-Join*, *Diskriminator* bzw. *N-out-of-M-Join* sowie die parallel ausgeführte Variante von *For-Each* (vgl. Abschnitte 2.4.1 und 2.4.2).

Im Folgenden wird die dynamische Verteilung paralleler Prozesspfade durch Replikation des Prozesses genauer erläutert. Im Anschluss wird darauf basierend die Identifikation und Behandlung von Datenabhängigkeiten zwischen gemeinsam auf mehreren parallelen Pfaden verwendeten Prozessvariablen beschrieben. Der Abschnitt schließt mit einem Vorschlag für ein geeignetes Synchronisationsverfahren, welches die speziellen Anforderungen dynamisch verteilt ausgeführter Prozesse – auch in Hinblick auf eine etwaige Offline-Bearbeitung von parallelen Prozesspartitionen – berücksichtigt.

Replikation des Prozesses

Um den Prozess für die Ausführung paralleler Pfade auf verschiedene Ausführungseinheiten zu verteilen, kann die für die aktuelle Ausführung verantwortliche Prozess-Engine (unter Beachtung etwaiger benutzerdefinierter Vorgaben) das Prozessmodell und die laufende Prozessinstanz replizieren und für die resultierenden Replikate geeignete Ausführungseinheiten auswählen. Es können im Rahmen der benutzerdefinierten Vorgaben insgesamt die folgenden Verteilungsstrategien bzw. Vorgehensweisen zur Ausführung der parallelen Prozesspartitionen berücksichtigt werden:

- **Keine verteilte Ausführung paralleler Pfade:** Die aktuell für die Prozessausführung verantwortliche Ausführungseinheit ist für alle aktuell auszuführenden parallelen Pfade selbst verantwortlich. Bei einer Migration der Prozessinstanz wird diese Verantwortlichkeit in ihrer Gesamtheit an eine einzelne Zielausführungseinheit übertragen. Es findet keine Replikation des Prozesses statt.

- **Zentrale Delegation einzelner paralleler Pfade:** Die aktuell für die Prozessausführung verantwortliche Ausführungseinheit wählt eine Menge von lokal auszuführenden Prozesspfaden und delegiert die übrigen parallelen Pfade an eine ausgewählte Menge von anderen Ausführungseinheiten. Für jede gewählte Ausführungseinheit wird dazu nacheinander ein aktuelles Replikat des Prozesses (d. h. der Prozessbeschreibung und der aktuellen Migrationsdaten) erzeugt und die erste auszuführende Aktivität des zugewiesenen Pfades (bzw. die Menge der ersten auszuführenden Aktivitäten bei der Zuweisung mehrerer Pfade) als Startaktivität (bzw. Startaktivitäten) des Replikats gesetzt. Die Delegation kann dabei erfolgen, sobald nach Eintritt in die parallele Partition ein *Option*-Zustand des Prozesses erreicht wurde (vgl. Abschnitt 6.2.2), d. h. sowohl direkt nach einer Kontrollflussverzweigung (*AND*- oder *OR*-

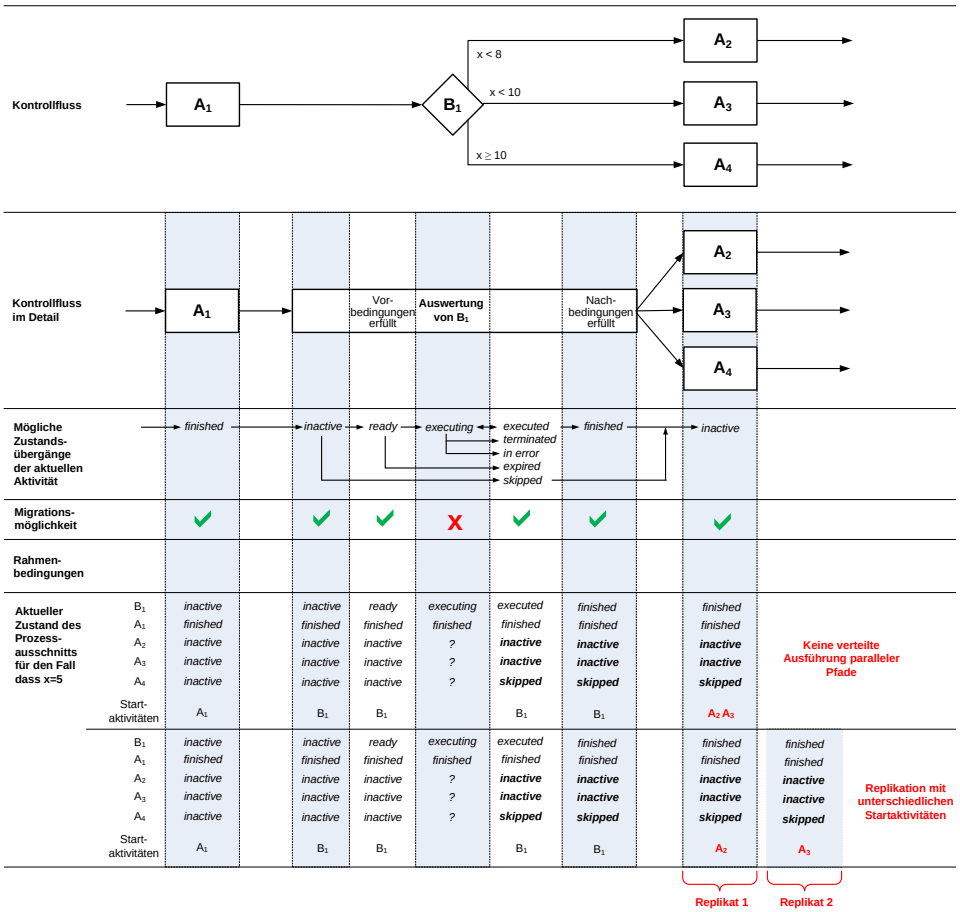


Abbildung 6.14: Migrierbarkeit des Kontrollflusses nach einer *Split*-Bedingung mit Möglichkeit zur parallelen Ausführung der Aktivitäten A_2 und A_3 (Beispiel für x=5)

Split) und damit zu Beginn der parallelen Prozesspartition oder nachdem bereits Aktivitäten auf den parallelen Pfaden ausgeführt wurden. Die beteiligten Ausführungseinheiten können im Rahmen der benutzerdefinierten Vorgaben wiederum ihr Replikat an andere Ausführungseinheiten migrieren oder im Fall einer geschachtelten parallelen Prozesspartition neue Replikate des Prozesses erzeugen und ihrerseits verteilen.

- **Vollständige Delegation aller parallelen Pfade an mehrere Ausführungseinheiten:** Die aktuell für die Prozessausführung verantwortliche Ausführungseinheit delegiert alle parallel auszuführenden Prozesspfade an mehrere andere Ausführungseinheiten und nimmt selbst nicht länger an der Prozessausführung teil. Es verbleibt also bei dieser Variante kein Replikat des Prozesses auf dem Quellausführungssystem. Ansonsten gilt die bei der Delegation einzelner paralleler Pfade beschriebene Vorgehensweise zur Replikation sowie die entsprechende Möglichkeit zur Weitergabe der Prozessausführung durch die Zielausführungssysteme.

- **Kaskadierende Delegation paralleler Pfade:** Die aktuell für die Prozessausführung verantwortliche Ausführungseinheit wählt eine Menge von lokal auszuführenden Prozesspfaden und erzeugt ein einziges Replikat für die Ausführung der nicht ausgewählten Prozesspfade mit den entsprechenden Startaktivitäten. Dieses Replikat wird an eine einzige ausgewählte Ausführungseinheit weitergegeben, welche ihrerseits Prozesspfade zur Ausführung auswählt und ein Replikat mit den übrigen Prozesspfaden weitergibt [KZL06, KZL07b]. Im Gegensatz zur zentralen Delegation erfolgt hierbei die Auswahl der auszuführenden Pfade nicht durch einen temporär zentralen Koordinator, sondern durch die Zielausführungssysteme selbst und ist somit weniger fremdbestimmt. Eine Kombination mit den zuvor genannten Strategien innerhalb der Kaskade ist natürlich weiterhin möglich.

Eine Replikation des Prozesses darf in jedem Fall erst erfolgen, wenn durch die aktuell verantwortliche Ausführungseinheit alle aktuellen *Split*-Bedingungen ausgewertet wurden und somit festgestellt wurde, welche Prozesspfade überhaupt ausgeführt werden sollen. Aktivitäten auf nicht auszuführenden Pfaden werden hierbei durch den Zustand *Skipped* gekennzeichnet, welcher im Rahmen der *Dead Path Elimination* anzeigt, dass bei einer späteren Synchronisation nicht auf das Eintreffen dieses Pfades gewartet werden muss (vgl. Abschnitt 2.7.2). Auch wenn alle beteiligten Ausführungssysteme bereits zur Entwicklungszeit vorgeben wurden, ist es vorteilhaft, die Replikate erst zum spätmöglichsten Zeitpunkt zu erzeugen, da jedes Replikat ein vollständiges Abbild der Prozessbeschreibung (Prozessmodell) und eine zum Zeitpunkt der Replikation aktuelle Version der Migrationsdaten (Prozessinstanz, Logdaten und benutzerdefinierte Vorgaben) darstellt. Die Replikate unterscheiden sich dabei mindestens durch die Vergabe unterschiedlicher Startaktivitäten und sind jeweils genau einer Ausführungsumgebung zugewiesen. Auf diese Weise ist immer genau eine Ausführungseinheit für eine festgelegte Menge von eindeutig bestimmten Prozesspfaden verantwortlich und kann entsprechend mit der Ausführung der zugewiesenen Pfade fortfahren [ZKML10]. Abbildung 6.14 zeigt die genannte Vorgehensweise an einem Beispiel.

Erkennung und Behandlung von Datenabhängigkeiten

Ein wesentlicher Aspekt bei der verteilten Ausführung paralleler Prozesspfade ist die Erkennung und Behandlung von Konflikten, welche durch den gemeinsamen Zugriff auf Prozessvariablen aus verschiedenen parallel ausgeführten Pfaden des Prozesses heraus entstehen. Analog zu Mehrbenutzeranomalien in klassischen Datenbanksystemen ohne Transaktionsverwaltung können unerwartete bzw. inkorrekte Ergebnisse verursacht werden, wenn global definierte Prozessvariablen in unterschiedlichen Prozesspartitionen mittels Lese-/Schreib-Zugriffen verwendet werden und während des Zugriffs nicht gesperrt bzw. vor und nach dem Zugriff nicht synchronisiert werden [ZKML10].

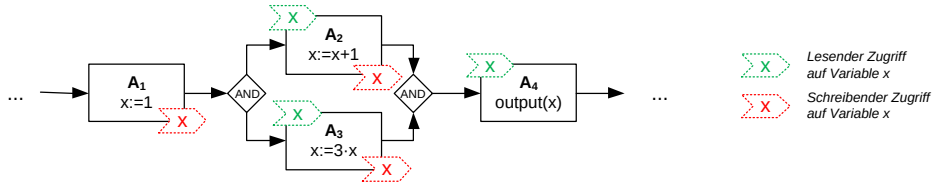


Abbildung 6.15: Beispiel für ein Prozessmodell mit Datenabhängigkeiten (nach (Ham09))

Ein einfaches Beispiel für eine derartige Situation ist in Abbildung 6.15 gezeigt (vgl. auch [Ham09]). Die Aktivitäten des dargestellten Prozesses greifen auf die globale Prozessvariable *x* zu, um diese zu initialisieren (A_1), einfache Berechnungen darauf durchzuführen (parallele Prozesspartition mit A_2 und A_3) und schließlich das Ergebnis dieser Berechnungen auszugeben (A_4). Betrachtet man eine Aktivität jeweils als atomaren Schritt, so ergeben sich mit $w_1 = A_1A_2A_3A_4$ und $w_2 = A_1A_3A_2A_4$ zwei mögliche serielle Ausführungsreihenfolgen. Bei der Folge w_1 wird in A_4 das Ergebnis $x = 6$ ausgegeben, während bei w_2 das Ergebnis $x = 4$ berechnet wird. Da die Ausführungsreihenfolge wegen der Parallelausführung nicht festgelegt ist, sind prinzipiell beide Ergebnisse zulässig.

Betrachtet man jedoch eine Aktivität i als Transaktion aus den beiden Operationen des Lesens von Eingabedaten (l_i) und des Schreibens von Ausgabedaten (s_i) (vgl. Abschnitt 2.5), so ist auch die Ausführungsreihenfolge $w_3 = l_1s_1l_2l_3s_3s_2l_4s_4$ mit dem Ergebnis $x = 2$ möglich. Hierbei überschreibt die Aktivität A_2 die Ausgabedaten von A_3 ohne den aktuellen Wert dieser Prozessvariablen bei ihren Eingabedaten berücksichtigt zu haben, so dass es hierbei – ohne Berücksichtigung der tatsächlichen Absichten des Prozessmodellierers – allein durch die Parallelausführung der Aktivitäten A_2 und A_3 zu einer fachlich nicht nachvollziehbaren Ausgabe in Aktivität A_4 kommt. Als Analogie zu typischen Mehrbenutzeranomalien bei der Durchführung von Datenbanktransaktionen kann man bei diesem Phänomen auch von einer *verlorengegangenen Änderung* (*Lost Update*) sprechen [Ham09].

Eine einfache Herangehensweise zur Vermeidung solcher unerwünschter Erscheinungen ist es, alle in einer Aktivität verwendeten Prozessvariablen (d. h. sowohl Eingabe- als auch Ausgabeparameter) über alle an der parallelen Ausführung beteiligten Ausführungseinheiten zu sperren und diesen nach Beendigung der Ausführung mit der Aufhebung der Sperre den aktuellen Wert der betreffenden Prozessvariablen mitzuteilen (vgl. z. B. *strikte Synchronisation* in [MWW+98]). Hierdurch kommt es zu einem hohen Kommunikationsaufwand, da nicht nur die Anforderung und Verwaltung von Sperren verteilt koordiniert werden muss, sondern auch die Werte aller Prozessvariablen unabhängig von ihrer Größe und ihrer späteren Verwendung ausgetauscht werden müssen. In vielen Fällen ist eine solche strikte Synchronisation unter Umständen auch gar nicht möglich, da (parallele) Prozesspartitionen offline (z. B. auf mobilen Geräten) ausgeführt werden sollen oder die beteiligten Ausführungseinheiten einander gar nicht bekannt sind.

Im Gegensatz zu Datenbanktransaktionen, bei denen Transaktionen auf beliebigen Daten ohne vorhersehbare Reihenfolge verwaltet werden müssen, kann der Synchronisationsaufwand bei parallel ausgeführten Prozesspfaden jedoch reduziert werden, da der Datenzugriff auf Basis des fest vorgegebenen Prozessmodells erfolgt und somit potentielle Datenabhängigkeiten bereits im Voraus abgesehen werden können [Ham09]. Der erste Schritt zur Vermeidung von unnötigem Synchronisationsaufwand ist daher die Identifikation von Datenabhängigkeiten, die während einer parallelen Ausführung des Prozesses Ergebnisse verursachen würden, welche einem vorgegebenen *Korrektheitskriterium* nicht genügen würden. In vielen Fällen ist dieses Korrektheitskriterium anwendungsabhängig, d. h. der Modellierer des Prozesses verfolgt mit dem explizit modellierten gemeinsamen Zugriff einer Variablen aus verschiedenen parallelen Pfaden eine gewisse Absicht (vgl. auch [CR04]). Ein anwendungsnahes Beispiel hierfür ist die Nutzung einer einfachen Zählvariablen, welche die Anzahl des Vorkommens eines gewissen Sachverhalts auf verschiedenen Prozesspfaden festhält, um diese später (ggf. erst nach der Zusammenführung der parallelen Pfade) durch weitere Aktivitäten zu nutzen. Eine Synchronisation dieser Zählvariablen ist in diesem Fall z. B. auch durch eine einfache Addition der Werte zum Zeitpunkt der Zusammenführung des Kontrollflusses zu erreichen. Bei anderen Variablen kann sich der Modellierer z. B. im Vornherein bewusst sein, dass keine relevanten Änderungen bei der Prozessausführung zu erwarten sind oder dass lediglich eine gewisse Aktualität der Daten eine Rolle spielt (z. B. bei einer Wettervorhersage), so dass unter Umständen auch gar keine oder zumindest keine ständige Synchronisation erforderlich ist (vgl. [Ham09] zu Details).

Aufgrund der vorgegebenen Struktur von Prozessen und der festgelegten Zusammensetzung jeder Aktivität aus höchstens einer Leseoperation des *Input-Containers* zu Beginn und einer Schreiboperation des *Output-Containers* als Abschluss der Aktivitätsausführung (vgl. Abschnitt 2.5) lassen sich potentielle Datenabhängigkeiten bereits aus der zugrunde liegenden Prozessbeschreibung ableiten [Ham09]. Die in Abschnitt 6.2.1 vorgestellte Methodik für dynamisch verteilt ausgeführte Prozesse kann daher zur Reduktion des Synchronisationsaufwands um eine präventive Prüfung des Prozessmodells erweitert werden, bei welcher identifiziert wird, ob der Prozess parallele Pfade aufweist und ob innerhalb solcher Pfade Lese-Schreib-Abhängigkeiten zwischen gemeinsam verwendeten Prozessvariablen existieren. Werden bei dieser ersten Prüfung Datenabhängigkeiten (*Data Conflicts*) erkannt, so werden diese in den Migrationsdaten mit einer *Datenklasse* (*Data Class*) gekennzeichnet, welche angibt, wie diese Variable bei einer Verwendung in parallelen Prozesspfaden zur Laufzeit synchronisiert werden soll (vgl. auch [CR04, YV02]). Um eine Automatisierbarkeit des Vorgangs zu erlauben, wird in dieser Arbeit vorgeschlagen, zunächst ein möglichst restriktives Korrektheitskriterium als Voreinstellung zu verwenden und damit die Korrektheit der Ausführung in jedem Fall sicherzustellen. Dieses Kriterium kann dann bei Bedarf durch den Prozessmodellierer in einem optionalen zweiten Schritt gelockert werden, indem die Daten-

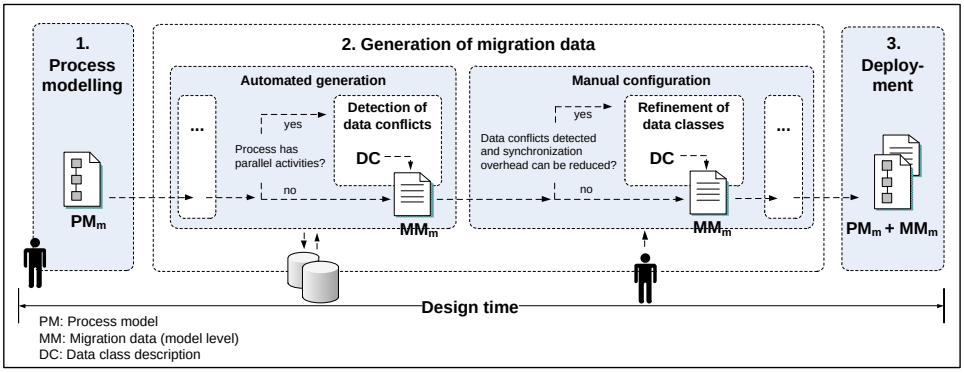


Abbildung 6.16: Vorgehensweise zur Reduktion des Synchronisationsaufwands für parallel ausgeführte Prozesspfade (Entwicklungszeit) (ZKML10)

klasse entsprechend modifiziert wird [ZKML10]. Abbildung 6.16 veranschaulicht diese Schritte im Kontext der übergeordneten Vorgehensweise.

Für die automatische Erkennung von Datenabhängigkeiten ist zunächst ein möglichst generisches Korrektheitskriterium erforderlich, welches notfalls auch ohne zusätzliche Konfiguration des Benutzers zu fachlich nachvollziehbaren Ergebnissen führt. Ein solches anwendungsunabhängiges Korrektheitskriterium stellt das aus dem Bereich der Datenbanktransaktionen bekannte Kriterium der *Serialisierbarkeit* dar, welches besagt, dass eine Ausführungsfolge genau dann *serialisierbar* ist, wenn das Ergebnis und alle erzeugten Ausgaben der Ausführungsfolge äquivalent zu denen einer möglichen seriellen Ausführung sind [EGLT76]. Da es sich bei verteilt ausgeführten parallelen Prozesspfaden um Replikate eines Prozesses handelt, muss zusätzlich gefordert werden, dass sich die nebenläufige Ausführung von Aktivitäten *auf Replikaten* äquivalent zu einer möglichen seriellen Ausführungsfolge auf einem zentralen Datenspeicher verhält [Ham09]. Nach BERNSTEIN und GOODMAN [BG83] spricht man hierbei im Kontext von Datenbanktransaktionen von *Eine-Kopie-Serialisierbarkeit* (*One-Copy Serializability*). Da geeignete Verfahren aus dem Bereich der verteilten Datenbanken bestehen, um Eine-Kopie-Serialisierbarkeit zu garantieren (vgl. [Ham09]), kann dieses Korrektheitskriterium als Standardeinstellung für dynamisch verteilte Prozesse mit Datenabhängigkeiten angewendet werden.

Um Eine-Kopie-Serialisierbarkeit bei einer verteilten Ausführung paralleler Prozesspfade zu gewährleisten, ohne eine strikte Synchronisation vornehmen zu müssen, müssen als nächstes alle Datenabhängigkeiten identifiziert werden, welche (unabhängig von der tatsächlichen Ausführung) zu nicht serialisierbaren Ausführungsfolgen führen können. Paralleles Schreiben derselben Variablen ohne vorheriges Lesen des Wertes stellt in Bezug auf die Serialisierbarkeitanforderung kein Problem dar, da der letztendlich resultierende Wert auch bei einer sequentiellen Ausführung von der Reihenfolge der Aktivitätsausführung abhängt (vgl. [Ham09]). Anders verhält es sich bei *Lese-Schreib-Abhängigkeiten*, welche auftreten, wenn eine Prozessvariable v von ei-

ner Aktivität A_1 gelesen wird, von einer Aktivität A_2 geschrieben wird und A_1 und A_2 im Prozessmodell parallel zueinander modelliert sind. Wenn während der parallelen Ausführung alle Änderungen an Variablenwerten nur auf dem lokalen Replikat erfolgen sollen, ist eine parallele Ausführung im Allgemeinen nur dann zu einer seriellen Ausführungsfolge äquivalent, wenn in einer seriellen Ausführung das Lesen der Variablen v durch A_1 vor dem Schreiben von v durch A_2 erfolgen kann. Andernfalls wäre ein Austausch des geschriebenen Wertes von v und damit eine Kommunikation zwischen den parallel ausgeführten Prozesspfaden erforderlich [Ham09].

Die Menge dieser durch eine Nebenläufigkeitskontrolle zu behandelnden Abhängigkeiten kann durch die Erstellung eines *Abhängigkeitsgraphen* festgestellt werden, welcher die für die Ausführung relevanten Aktivitäten des Prozesses als Knoten und sowohl die Transitionen des Prozessmodells als auch die durch die Lese-Schreib-Abhängigkeiten resultierenden Präzedenzen als gerichtete Kanten enthält (für eine formale Darstellung vgl. [Ham09]). Ist der aus einem gegebenen Prozessmodell gebildete Abhängigkeitsgraph zyklenfrei, so kann durch die topologische Sortierung der Aktivitäten mindestens eine zur Parallelausführung äquivalente Serialisierung abgeleitet werden, wodurch das vorgegebene Korrektheitskriterium erfüllt werden kann. Hingegen stellt ein Zyklus im Abhängigkeitsgraph einen *Abhängigkeitskonflikt* dar, da dieser einen Widerspruch in der Reihenfolge der Aktivitäten kennzeichnet und die Lese-Schreib-Abhängigkeit nicht durch die Wahl einer geeigneten Serialisierung aufgelöst werden kann. Besteht ein Abhängigkeitskonflikt innerhalb verteilt ausgeführter paralleler Prozesspfade, so ist zur Laufzeit eine Synchronisation der Replikate notwendig, um diesen Konflikt aufzulösen [Ham09]. Die einen Abhängigkeitskonflikt verursachenden Prozessvariablen werden daher in den Migrationsdaten mit der (restriktivsten) Datenklasse *Synchronized* gekennzeichnet. Abbildung 6.17 verdeutlicht die prinzipielle Vorgehensweise zur Erkennung von Abhängigkeitskonflikten und der Zuweisung der Datenklasse *Synchronized* graphisch. Ein geeignetes abstraktes mathematisches Modell aus Ausgangspunkt der Erkennung und Kennzeichnung von Abhängigkeitskonflikten, eine entsprechende formale Vorgehensweise sowie ein formaler Beweis der Korrektheit dieses Verfahrens auf der Basis von *Replicated Database Logs* (*RD Logs*) [BG83] sind der Arbeit von HAMANN [Ham09] zu entnehmen.

Der Prozessmodellierer kann nach Abschluss der automatischen Erkennung die Menge der zur Laufzeit unter Umständen zu synchronisierenden Variablenwerte weiter beschränken, indem semantisches Wissen über die Korrektheit der Prozessausführung eingebracht wird (s.o.). In Ergänzung zu den von *Cichocki* und RUSINKIEWICZ (vgl. Abschnitt 5.5.1 und [CR04]) vorgegebenen Datenklassen *Restricted* (strikte Synchronisation), *History Merging* (Wiederholung aller Datenoperationen auf Basis von Logdaten) und *Aggregate* (Aggregation von Daten z. B. durch Summe, Durchschnitt oder Konkatenation) wird in dieser Arbeit eine Datenklasse als abstraktes benutzerdefiniertes Konstrukt aufgefasst, welches prinzipiell beliebige Korrektheitskriterien repräsentieren

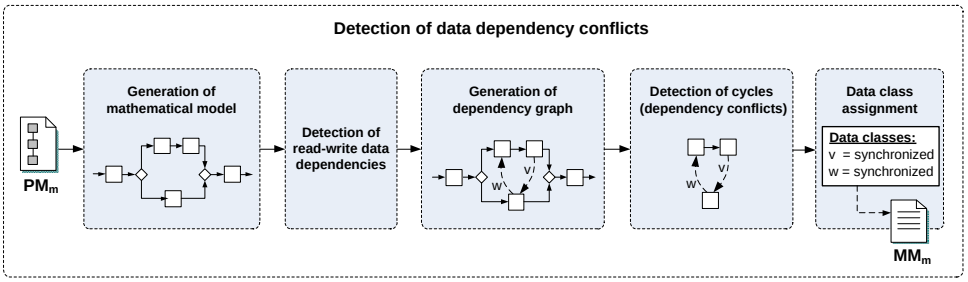


Abbildung 6.17: Vorgehensweise zur Erkennung und Kennzeichnung von Abhängigkeitskonflikten in parallelen Prozesspfaden

kann. Als weitere Beispiele für Datenklassen können z. B. die Klasse der nicht zu synchronisierenden Prozessvariablen (*Unsynchronized*) oder die Klasse der Prozessvariablen, die eine gewisse Aktualität aufweisen müssen (*Updated*) angeführt werden (vgl. [ZKML10]).

Die Erkennung und Kennzeichnung von Abhängigkeitskonflikten in parallelen Prozesspfaden kann jederzeit auch zur Laufzeit des Prozesses durchgeführt werden und somit die Anzahl der tatsächlichen Abhängigkeitskonflikte laufender Prozessinstanzen individuell konkretisieren. Werden zum Beispiel einige parallele Prozesspfade aufgrund von negativ evaluierten Kontrollflussbedingungen am verzweigenden *OR-Split* nicht ausgeführt, so müssen die entsprechenden Prozessvariablen auch nicht bei der Erstellung des Abhängigkeitsgraphen berücksichtigt werden. Die Anzahl der zu synchronisierenden Prozessvariablen wird somit zur Laufzeit weiter reduziert. Eine Anpassung der Datenklassen ist prinzipiell ebenfalls zur Laufzeit möglich, solange noch keine Replikate des Prozesses erzeugt und verteilt wurden. Da für diese Konfiguration jedoch hinreichend großes Wissen über die Semantik des Prozesses vorhanden sein sollte, welches in der Regel nur beim Modellierer des Prozesses vorausgesetzt werden kann, ist eine Lockerung des Korrektheitskriteriums durch den Prozessinitiator bzw. durch beliebige Prozessteilnehmer nur im Einzelfall anwendbar.

Synchronisation verteilt ausgeführter paralleler Prozesspfade

Abbildung 6.18 fasst die bis hierhin erläuterten Schritte für eine verteilte Ausführung paralleler Prozesspfade zur Laufzeit migrierter Prozessinstanzen zusammen. Dabei wird eine Prozessinstanz zunächst nach der in Abschnitt 6.2.4 dargestellten Vorgehensweise sequentiell ausgeführt und ggf. migriert, bis im Kontrollfluss eine Verzweigung erreicht wird und nach der Auswertung der entsprechenden Kontrollflussbedingung mehrere parallele Pfade verteilt ausgeführt werden sollen. Es folgt die oben dargestellte Replikation des Prozesses mit der Verteilung der Replikate und deren Ausführung. Auf Basis der Prozessbeschreibung und der identifizierten Datenabhängigkeiten zwischen mehreren parallelen Pfaden kann zur Laufzeit entschieden werden, ob eine Synchronisation der Replikate notwendig ist. Da eine implizite Synchronisa-

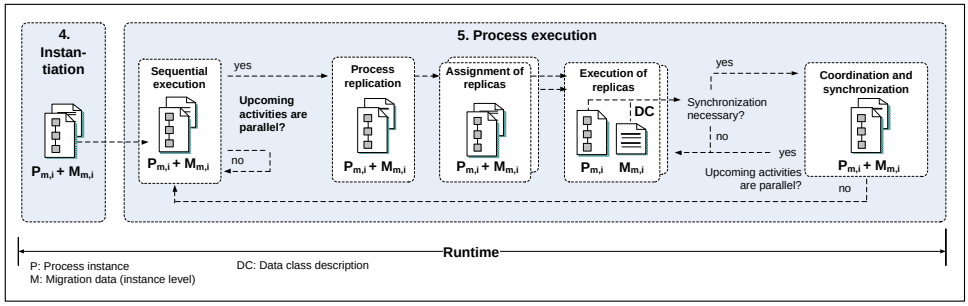


Abbildung 6.18: Parallele Ausführung und Synchronisation von migrierten Prozesspfaden (Laufzeit) (ZKML10)

tion von Prozessvariablen ebenso wie eine explizite Synchronisation des Kontrollflusses eine Kommunikation zwischen den an der Ausführung beteiligten Prozess-Engines erfordert, sind beide Arten der Synchronisation Inhalt dieses Unterabschnitts.

Die Zusammenführung der parallelen Pfade von verteilt ausgeführten Prozessen erfordert im Allgemeinen, dass die zur parallelen Ausführung erzeugten Replikate mit ihren Teilprozessausführungen auf einem (bestimmten) Ausführungssystem wieder zu einer einzigen Prozessinstanz vereinigt werden, so dass deren Ausführung wiederum durch ein einzelnes Prozessausführungssystem fortgesetzt werden kann. Im Fall eines *AND-Joins* muss das für die Synchronisation verantwortliche Ausführungssystem hierfür auf alle eingehenden Pfade warten. Für einen *OR-Join* ist darüber hinaus auch eine genaue Kenntnis der Anzahl der zuvor gestarteten parallelen Pfade erforderlich, damit die Prozessausführung nach dem Eingang der entsprechenden Kontrollflusspfade fortgesetzt werden kann. Der *Diskriminator* bzw. *N-out-of-M-Join* besitzt diese Information bereits a priori (vgl. Abschnitt 2.4.2). Der Diskriminator erfordert zwar keine Synchronisation des Kontrollflusses, da nach Eintreffen des ersten Pfades analog zu einer einfachen Sequenz von Aktivitäten mit der Ausführung der Folgeaktivitäten fortgefahren werden kann. Bei einem verteilt ausgeführten Prozess ist jedoch dennoch eine Abstimmung erforderlich, da in Erfahrung gebracht werden muss, ob es sich bei dem lokal ausgeführten Pfad um den ersten (bzw. N-ten) eingehenden Pfad oder um einen zu ignorierenden Pfad handelt. Bei einem *N-out-of-M-Join* mit $N > 1$ ist zusätzlich noch die Synchronisation der ersten N Pfade erforderlich.

Für die Synchronisation verteilt ausgeführter paralleler Pfade an einem Join-Punkt der genannten Arten muss also eine geeignete Ausführungseinheit als vorgegebener „Treffpunkt“ definiert werden, um den Kontrollfluss des Prozesses zusammenzuführen, die für die Parallelausführung erstellten Replikate auf eine gemeinsame Version zu reduzieren und hierbei die Werte der Prozessvariablen entsprechend der etwaigen Vorgaben zu synchronisieren. Die einfachste Lösung hierfür ist die Vorgabe eines konkreten Systems zur Durchführung dieser Maßnahmen durch die Spezifikation einer festen eindeutigen Zuordnung innerhalb der Migrationsdaten (Verteilungsstrategie *Fixed*

Participant, vgl. Abschnitt 6.2.2). Diese Zuordnung kann natürlich durchaus auch zur Laufzeit, d.h spätestens zum Zeitpunkt der Replikation, erfolgen. Zum Beispiel kann das für die Verteilung verantwortliche Ausführungssystem sich selbst als späteren Synchronisationspunkt aller parallelen Pfade und als Entscheider über die Fortführung des Kontrollflusses festlegen. In diesem (einfachsten) Fall migrieren alle beteiligten Ausführungseinheiten ihren Prozess aufgrund dieser in allen Replikaten der Migrationsdaten enthaltenen Vorgabe zu der angegebenen Ausführungseinheit. Da das erforderliche Wissen hinsichtlich der Anzahl der gestarteten Pfade als auch der Anzahl der zu verarbeitenden eingehenden Pfade in den Migrationsdaten bzw. in der Prozessbeschreibung enthalten ist, kann die Prozessausführung fortgeführt werden, sobald die Join-Bedingung erfüllt ist. Die ordnungsgemäße Durchführung der oben genannten Join-Varianten kann in diesem Fall daher ohne weiteren Koordinationsaufwand gewährleistet werden [ZKML10].

Eine größere Herausforderung ist die Synchronisation von parallelen Prozesspfaden für den Fall, dass zum Zeitpunkt der Replikation kein Ausführungssystem für die Zusammenführung des Kontrollflusses statisch festgelegt werden kann. Dies ist zum Beispiel im Kontext mobiler Ausführungssysteme der Fall, wo insbesondere bei lang andauernden Prozessen nicht immer abgesehen werden kann, ob ein bestimmtes Gerät zum Synchronisationszeitpunkt verfügbar sein wird. Wird vor der Verteilung der parallelen Pfade also keine Ausführungseinheit als festgelegter Synchronisationspunkt bestimmt und kommt es während der parallelen Ausführung zu weiteren Migrationen der einzelnen Replikate, so müssen die an der aktuellen Ausführung relevanter Aktivitäten beteiligten Ausführungseinheiten erstens dynamisch aufgefunden werden können und zweitens einen geeigneten Koordinationsmechanismus implementieren, um die notwendige Synchronisation dezentral durchzuführen. Dies gilt sowohl für die Synchronisation an nicht statisch zugewiesenen Zusammenführungen als auch für die zwischenzeitliche Synchronisation aufgrund von Abhängigkeitskonflikten.

Ein dynamisches Auffinden von Ausführungseinheiten, welche für die Ausführung einer bestimmten Aktivität verantwortlich sind, ist Aufgabe einer unterstützenden Kommunikationsinfrastruktur (vgl. z. B. [Böh09]). Für eine zur Laufzeit erfolgende verteilte Synchronisation ohne vorgegebenen Synchronisationspunkt wird aus der Sicht einer einzelnen Ausführungsumgebung eine Strategie zur Prüfung möglicher Verhaltensweisen vorgeschlagen. Als Vorbedingung hat die betrachtete lokale Ausführungsumgebung die Ausführung eines parallelen Pfades beendet und die aktuell für die anderen Pfade verantwortlichen Ausführungseinheiten identifiziert. Das nachfolgende Kontrollflusskonstrukt ist von der Art eines *AND-Join*, *OR-Join*, *Diskriminator* oder *N-out-of-M-Join*. Die Strategie umfasst die folgenden Prüfschritte (vgl. tlw. [Ham09]):

- **Teilnahme an einer bereits laufenden Synchronisation**: Die betrachtete lokale Ausführungsumgebung überprüft zunächst, ob eine an-

dere Ausführungseinheit bereits die Ausführung eines zu synchronisierenden parallelen Pfades beendet hat und auf die Migration von Replikaten wartet. In diesem Fall wird die Synchronisation von der bereits wartenden Ausführungsumgebung durchgeführt, indem das lokale Replikat dorthin migriert wird.

- **Auswahl eines Synchronisationspunkts:** Ist eine Teilnahme an einer laufenden Synchronisation nicht möglich, so wird durch die betrachtete lokale Ausführungsumgebung eine andere Ausführungsumgebung als Synchronisationspunkt gewählt. Hierfür gibt es drei Möglichkeiten, welche nacheinander durch die betrachtete lokale Ausführungsumgebung geprüft werden:
  - **Wahl einer anderen beteiligten Ausführungseinheit:** Existiert eine beliebige andere Ausführungsumgebung, welche ebenfalls einen zu synchronisierenden Pfad ausführt und unter Berücksichtigung der benutzerdefinierten Rahmenbedingungen als Synchronisationspunkt in Frage kommt, so wird diese als neuer Synchronisationspunkt gewählt. Sobald ein dort ausgeführter Pfad beendet wird, kann die Synchronisation begonnen werden. Die neu gewählte Ausführungseinheit wird somit zu einem auf sich selbst und auf andere wartenden Synchronisationspunkt.
  - **Wahl der eigenen Ausführungseinheit:** Sind alle anderen an der Ausführung der parallelen Pfade beteiligten Ausführungseinheiten nicht als Synchronisationspunkt geeignet, geht die betrachtete lokale Ausführungsumgebung selbst in den Wartezustand, um die Replikate der anderen Ausführungseinheiten nach Abschluss deren paralleler Ausführung zu synchronisieren.
  - **Wahl einer aktuell unbeteiligten Ausführungseinheit:** Ist unter Berücksichtigung der benutzerdefinierten Rahmenbedingungen keine der an der parallelen Ausführung beteiligten Ausführungseinheiten als Synchronisationspunkt geeignet, muss durch die betrachtete lokale Ausführungsumgebung eine bisher unbeteiligte Ausführungseinheit als Synchronisationspunkt gewählt werden. Das lokal abgeschlossene Replikat wird somit an die gewählte Ausführungseinheit migriert und diese beginnt auf weitere Replikate zu warten.

Sobald eine Ausführungsumgebung beginnt, auf die Replikate anderer Ausführungsumgebungen zu warten, werden die anderen Ausführungsumgebungen darüber mittels einer Nachricht informiert. Unter Umständen parallel gestartete Wartezustände müssen bei Eingang einer Wartenachricht abgebrochen werden [Ham09]. Da es bei dieser Vorgehensweise zu jedem Zeitpunkt nur einen Synchronisationspunkt gibt, kann die Ausführung des Prozesses fortgesetzt werden, sobald die zu erwartende Anzahl an Kontrollflusspfaden am Synchronisationspunkt eingetroffen ist. Das

Eintreffen weiterer Pfade kann vom Synchronisationspunkt ignoriert werden. Es entsteht somit für die verantwortliche Ausführungsumgebung nach der Fortführung des Kontrollflusses bzw. einer weitergehenden Migration des Prozesses kein weiterer Koordinationsaufwand.

Bei Ausnahmesituationen, welche dazu führen, dass die Ausführung eines parallelen Pfades abgebrochen werden muss, müssen – analog zu einer zentralen Ausführung – die dem Fehler nachfolgenden Aktivitäten bis zur nächsten Zusammenführung mit dem Zustand *skipped* gekennzeichnet werden. Abhängig von der Art der *Join*-Bedingung am Synchronisationspunkt kann somit entschieden werden, ob die Ausführung nach der Zusammenführung fortgesetzt werden kann oder insgesamt abgebrochen werden muss.

Unabhängig von einer zentralen oder verteilten Synchronisationsvorbereitung müssen die für die verteilte Parallelausführung erzeugten Replikate am Synchronisationspunkt zusammengeführt werden. Dies betrifft insbesondere die in den Migrationsdaten festgehaltenen Zustandsdaten der Aktivitäten, die Zustandsdaten der Variablen (d. h. deren Werte), sowie die jeweilige Aktualisierung der Log-Informationen. Hierzu wird das erste am Synchronisationspunkt eintreffende Replikat des Prozesses als Grundlage gewählt (*Basisreplikat*). Nach dem Eintreffen weiterer Replikate derselben Prozessinstanz werden bei der Synchronisation für alle in diesen Replikaten ausgeführten Aktivitäten bzw. veränderten Prozessvariablen die entsprechende Zustände bzw. Werte übernommen. Nicht veränderte Zustände und Werte ergeben sich automatisch aus dem Basisreplikat.

Die Werte von Prozessvariablen, welche nicht von Abhängigkeitskonflikten zu anderen parallelen Pfaden betroffen sind, können direkt in das Basisreplikat übernommen werden. Des Weiteren müssen jedoch bei der Zusammenführung des Kontrollflusses die zur Behandlung von Abhängigkeitskonflikten gewählten Synchronisationsstrategien vereinigt werden. Es entsteht hierbei ein globaler Abhängigkeitsgraph, welcher durch die Beseitigung der lokalen Zyklen ebenfalls zyklenfrei ist (Beweis in [Ham09]). Daher kann hieraus für die Synchronisation gemeinsam verwendeter Prozessvariablen eine serielle Ausführungsfolge berechnet werden [Ham09, ZHKK10].

Optimistische Konfliktauflösung

Obwohl der Synchronisationsaufwand durch die Erkennung von tatsächlichen Abhängigkeitskonflikten zur Laufzeit des Prozesses im Vergleich zu einer strikten Synchronisation bereits reduziert werden kann, ist eine zwischenzeitliche Synchronisation zwischen parallelen Prozesspfaden zur Gewährleistung der Eine-Kopie-Serialisierbarkeit unter Umständen nicht immer sofort möglich. Dies ist vor allem dann der Fall, wenn mobile Ausführungseinheiten an der Bearbeitung von parallelen Prozessschritten beteiligt sind und für einen gewissen Zeitraum offline arbeiten müssen. Durch Datenabhängigkeiten zu

anderen parallelen Pfaden kann es hierbei durch die temporäre Unerreichbarkeit eines mobilen Geräts zu Verzögerungen der Prozessausführung kommen, welche bei einer nicht-verteilten Ausführung nicht auftreten würden. Dies ist insbesondere bei der Ausführung von lang andauernden Aktivitäten der Fall [Ham09]. Um eine Lösung für die teilweise konfligierenden Anforderungen einer *korrekten Ausführung* (A_4) in Verbindung mit einer möglichen *Offline-Bearbeitung* von (parallelen) Prozessschritten (D_8) und einer *möglichst geringen Beeinflussung* der Prozessausführung (D_{11}) (vgl. Abschnitt 4.4) zu erarbeiten, wurde die Verwendung von optimistischen Verfahren zur Nebenläufigkeitskontrolle auf die Replikation von Prozessen zu deren paralleler Ausführung untersucht [Ham09, ZHKK10].

Optimistische Verfahren zur Nebenläufigkeitskontrolle sind aus dem Bereich der Datenbanktransaktionen bekannt (vgl. [KR81]). Im Gegensatz zu klassischen Sperrverfahren werden hierbei parallele Zugriffe nicht präventiv verhindert, sondern erst bei deren Auftreten behandelt. Basierend auf der Annahme, dass Abhängigkeitskonflikte nur relativ selten auftreten, können parallele Zugriffe auf gemeinsam genutzte Daten zunächst ohne Einschränkungen ausgeführt werden (*Lesephase*) [KR81]. Der Zugriff geschieht dabei auf einer lokalen Kopie der Daten, so dass andere Transaktionen (bzw. Aktivitäten) zunächst nicht beeinträchtigt werden. Nach Abschluss der lokalen Transaktion wird überprüft, ob in Hinblick auf die Ausführung anderer (paralleler) Transaktionen Serialisierbarkeit gewährleistet werden kann (*Validierungsphase*). Ist dies nicht der Fall, muss eine der konflikterzeugenden Transaktionen zurückgesetzt werden (ggf. durch Kompensation). Ist die Validierung erfolgreich, werden die Änderungen der lokalen Kopie persistent gespeichert (*Schreibphase*) [KR81]. In einer Verallgemeinerung dieses Verfahrens können auch mehrere hintereinander ausgeführte Transaktionen in zunächst unsynchronisierten Replikaten validiert werden [Dav84]. Dies erfordert in jedem Fall die Aufzeichnung aller lokalen Zugriffe und die generelle Möglichkeit zur Kompensation der ausgeführten Transaktionen bzw. Aktivitäten [Dav84].

Bei der hier vorgenommenen Anpassung des Verfahrens [Ham09, ZHKK10] können verteilt ausgeführte parallele Pfade mit Abhängigkeitskonflikten zunächst unsynchronisiert ausgeführt werden, indem auf der lokalen Ausführungseinheit zunächst optimistisch angenommen wird, dass

- der Kontrollfluss der anderen beteiligten Ausführungseinheiten den Abhängigkeitskonflikt noch nicht erreicht hat und
- die Benachrichtigung über die gewählte Art der Konfliktauflösung die betroffenen Ausführungseinheiten erreicht, bevor diese selbst mit einer lokalen Konfliktauflösung beginnen.

Auf Basis dieser Annahme kann lokal eine Konfliktauflösung vorgenommen werden, welche den Zyklus im lokalen Abhängigkeitsgraphen auflöst, die durchzuführende Synchronisation jedoch auf eine *andere* am Abhängigkeitskonflikt beteiligte Ausführungseinheit verschiebt. Dafür wird

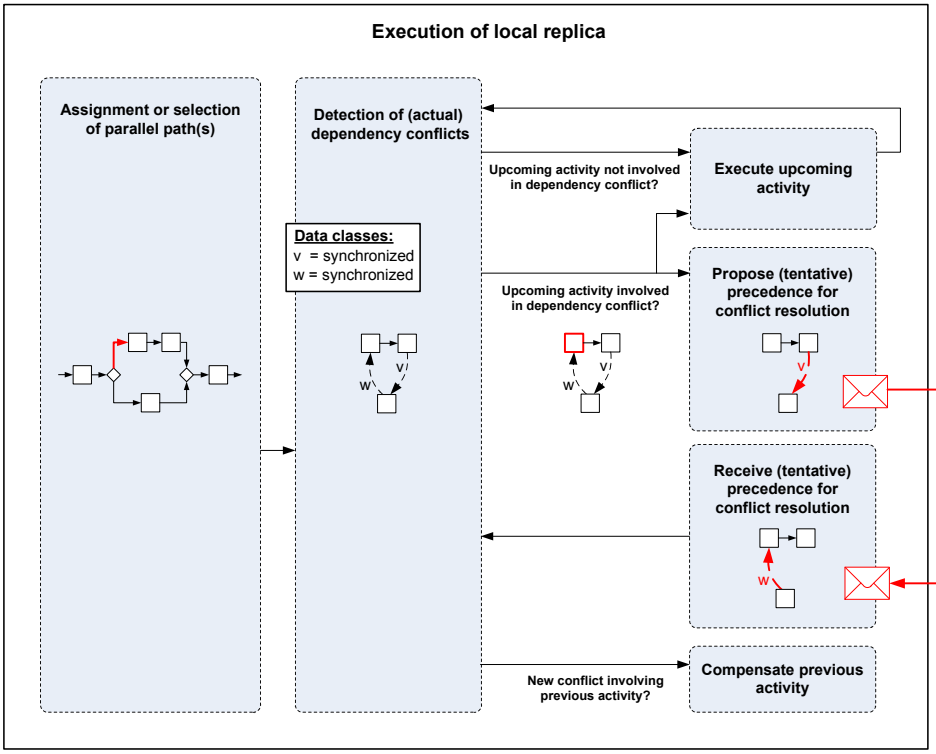


Abbildung 6.19: Parallele Ausführung aus der Sicht einer lokalen Ausführungseinheit mit optimistischer Konfliktauflösung zur Laufzeit

lokal eine Reihenfolge zwischen den ursprünglich parallelen Aktivitäten A_1 und A_2 verteilt ausgeführter Prozesspfade bestimmt, welche bewirkt, dass die von der lokal ausgeführten Aktivität A_1 geschriebenen Daten zunächst von der konfligierenden Aktivität A_2 auf dem parallelen Pfad gelesen werden müssen, bevor diese ausgeführt werden kann [Ham09]. Diese Synchronisationsanweisung entspricht einer neuen Präzedenz zwischen A_1 und A_2 und ist äquivalent zu einer expliziten Transition im Prozessmodell, welche besagt, dass A_2 nicht ausgeführt werden darf, bevor A_1 beendet wurde. Die neue Präzedenz kann somit direkt im lokalen Replikat des Prozessmodells verankert oder – zur Erhaltung der nicht-invasiven Strategie – im lokalen Replikat der Migrationsdaten vermerkt werden. Es folgt die Benachrichtigung der anderen am Abhängigkeitskonflikt beteiligten Ausführungseinheiten unter Angabe der zur Konfliktauflösung gewählten neuen Präzedenz. Insbesondere für die Benachrichtigung mobiler Ausführungseinheiten ist hierfür ein asynchrones Kommunikationsverfahren anzustreben.

Kann die Benachrichtigung mit der gewählten Konfliktauflösung von den am Abhängigkeitskonflikt beteiligten Ausführungseinheiten empfangen werden, bevor der jeweils von diesen verwaltete Kontrollfluss den Abhängigkeitskonflikt erreicht hat, so können die Ausführungseinheiten der zur Konfliktauflösung vorgeschlagenen Strategie direkt zustimmen und die neue Präzedenz in ihre lokalen Replikate integrieren. Haben jedoch bereits mehrere offline arbeitenden Ausführungseinheiten eine optimistische

Konfliktauflösung desselben Abhängigkeitskonflikts begonnen und lassen sich die individuell gewählten Strategien zur Konfliktauflösung hinterher nicht vereinigen, so muss eine Fehlerbehandlung mit Rücksetzung von Aktivitäten durchgeführt werden. Diese muss garantieren, dass die vorgeschlagenen Präzedenzen keinen (neuen) Zyklus bilden und die durch die Präzedenzen vorgegebenen Synchronisationen eingehalten werden [Ham09]. Somit müssen nicht alle aktuellen Ausführungen abgebrochen und die durchgeführten Aktivitäten kompensiert werden, sondern es kann bei Bedarf die Ausführung mit dem geringsten Wiederherstellungsaufwand ausgewählt werden. Der vollständige Algorithmus sowie das Kommunikationsprotokoll für die optimistische Konfliktauflösung können der Arbeit von HAMANN [Ham09] entnommen werden.

Wie bereits oben beschrieben wurde, ist eine optimistische Konfliktauflösung nur unter der Annahme sinnvoll, dass Konflikte relativ selten auftreten und dass bei einer fehlschlagenden Validierung eine Kompensation der Aktivitäten vorgenommen werden kann. Enthält ein paralleler Ausführungspfad nichtkompensierbare Aktivitäten, so sollte die Ausführung daher ausgesetzt werden, bis eine Kommunikation zwischen den beteiligten Ausführungseinheiten und damit die Einigung auf eine Strategie zur Konfliktauflösung zustande kommt. Jede notwendige Kompensation bedeutet zudem den Verzicht auf bereits erfolgreich verrichtete Arbeit und die Notwendigkeit zu deren Wiederholung, welche ebenfalls zu einer Verzögerung der Prozessausführung führt. Aus diesem Grund sollten möglichst keine verschachtelten optimistischen Konfliktauflösungen vorgenommen werden. Vor der optimistischen Ausführung eines weiteren Abhängigkeitskonflikts, einer weiteren Verzweigung des Kontrollflusses oder natürlich eines explizit im Prozessmodell verankerten Synchronisationspunkts sollte die Ausführung daher angehalten und erst nach einer Bestätigung der Konfliktauflösungsstrategie bzw. einer vollständigen Synchronisation fortgeführt werden [Ham09].

Abbildung 6.19 zeigt abschließend eine Zusammenfassung der Behandlung von Datenabhängigkeiten zur Laufzeit des Prozesses für Prozessvariablen, welche bei der Erkennung von Datenabhängigkeiten automatisch als *synchronized* gekennzeichnet wurden.

6.2.6 Verteilung ereignisbasierter Kontrollflussstrukturen

Wie in den Abschnitten 2.4.4 und 3.7 erläutert wurde, kann eine Flexibilisierung des Prozesses in fachlicher Hinsicht unter anderem durch die Modellierung von ereignisbedingten Reaktionsmöglichkeiten innerhalb des Prozesskontrollflusses erreicht werden. Bei einer zentralen Ausführung eines Prozesses ist es hierfür erforderlich, dass beim Deployment des Prozesses bzw. spätestens bei dessen Instantiierung die für diese Reaktionen relevanten Ereignisse durch die verantwortliche Ausführungseinheit abonniert werden. Für eine dynamische Migration von Prozessen stellt die Verwaltung und Verarbeitung von Ereignissen jedoch eine Herausforderung dar, da die Gefahr

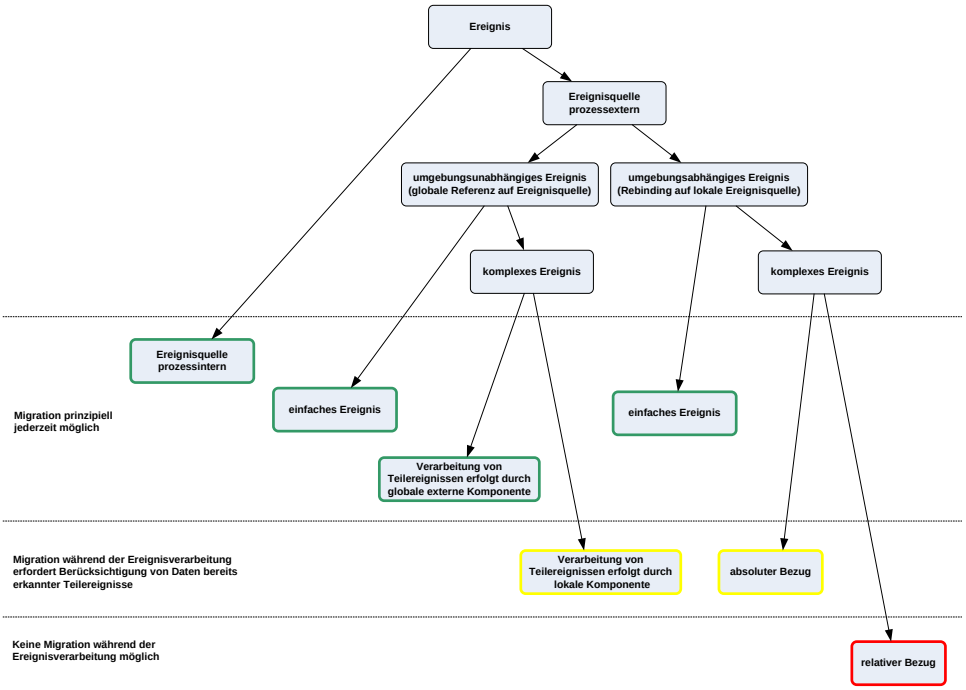


Abbildung 6.20: Klassifizierung von Ereignistypen zur Entscheidung über die Migrierbarkeit von ereignisbasierten Kontrollflussstrukturen (BDH+12)

besteht, dass Ereignisse während eines Transfers des Prozesses von keiner Ausführungseinheit aufgefangen oder während der verteilten Ausführung nicht der aktuellen Ausführungsumgebung zugeordnet werden können.

Um eine geeignete Behandlung von ereignisbasierten Kontrollflusskonstrukten abzuleiten, ist zunächst eine genauere Betrachtung unterschiedlicher Ereignistypen und ihre Klassifizierung in Hinblick auf eine Migration des Prozesses notwendig (vgl. Abbildung 6.20). Dazu soll zunächst zwischen *prozessinternen* und *prozessexternen Ereignissen* unterschieden werden. Prozessinterne Ereignisse haben ihren Ursprung in der Prozessinstanz selbst und können daher nur auftreten, solange Aktivitäten im Prozess ausgeführt werden. Da während der Ausführung von Aktivitäten keine Migration erfolgen kann, können prozessinterne Ereignisse auch nur bei der jeweils verantwortlichen Ausführungsumgebung auftreten und müssen dort auch direkt verarbeitet werden. Beispiele für solche Ereignisse sind Fehler während der Ausführung von Aktivitäten (z. B. durch einen Verbindungsabbruch zu einer aufgerufenen Software-Anwendung) oder durch den Prozess selbst ausgelöste Ereignisse wie das Senden einer Nachricht. Prozessinterne Ereignisse sind daher in Bezug auf eine Migration des Prozesses unproblematisch.

Prozessexterne Ereignisse sind Ereignisse, welche direkt oder indirekt durch die Ausführungsumgebung einer Prozessinstanz ausgelöst werden. In Bezug auf die Migration kann hierbei zwischen *umgebungsunabhängigen* und *umgebungsabhängigen Ereignissen* differenziert werden. Bei umgebungsunabhängigen Ereignissen ist eine eindeutige Quelle des Ereignisses mit glo-

baler Referenz vorgegeben. Der Wechsel der Ausführungsumgebung hat daher keinen Einfluss auf das Eintreten oder Ausbleiben von Ereignissen dieser Art. Ein Beispiel ist das Abonnement von Ereignissen eines zentralen Finanzdienstleister, welcher das Steigen von Börsenkursen über einen bestimmten Wert bekannt gibt. Bei umgebungsabhängigen Ereignissen kann die konkrete Quelle des Ereignisses von Ausführungseinheit zu Ausführungseinheit variieren und nach einer Migration des Prozesses neu gebunden werden. Der Wechsel der Ausführungsumgebung kann daher Einfluss auf das Eintreten oder Ausbleiben von Ereignissen dieser Art haben. Beispiele sind absolute Zeitereignisse ohne Angabe einer Zeitzone (z. B. zwölf Uhr mittags), physikalische Ereignisse (z. B. Temperatur größer 30° Celsius) oder Benutzerereignisse (z. B. Eingabe zum Abbruch des Prozesses). An den genannten Beispielen ist bereits ersichtlich, dass das Eintreten solcher Ereignisse vom Kontext der aktuellen Ausführungsumgebung abhängig sein kann oder sogar soll. Das potentielle Eintreten oder Ausbleiben eines bestimmten Ereignisses kann damit möglicherweise sogar den Hintergrund einer Migration des Prozesses darstellen. Ein einfaches Beispiel hierfür ist die Verlagerung der Prozessausführung in eine andere Region, um die dort vorherrschenden Wetterbedingungen für bestimmte Aktivitäten auszunutzen.

Solange es sich bei den Ereignissen um *einfache Ereignisse* oder durch eine andere (globale) Komponente bereits *vorverarbeitete Ereignisse* handelt, ist eine Migration des Prozesses prinzipiell zu jedem der erlaubten Zeitpunkte möglich. Werden jedoch lokal aus einem Strom von Ereignissen komplexe Ereignisse nach vorgegebenen Mustern identifiziert (vgl. Abschnitt 3.7.5), so ist es für eine (in Bezug auf die Ereignisverarbeitung verlustfreie) Migration von Prozessen notwendig, die gesammelten Informationen über die lokal bereits erkannten Teilereignisse ebenfalls an die Zielausführungseinheit weiter zu reichen, um die Ereignisverarbeitung nach der Migration des Prozesses am Zielort fortsetzen zu können.

Zudem können komplexe Ereignisse wiederum auf der Basis absoluter oder relativer Änderungen ausgelöst werden. In Bezug auf eine Migration von Prozessen ist dieser Unterschied vor allem bei den zuvor beschriebenen umgebungsabhängigen Ereignissen kritisch. So kann eine komplexe Ereignisverarbeitung aus mehreren historischen Werten und dem Vergleich zu aktuellen Daten nach einer Migration unter Umständen zu nicht mehr nachvollziehbaren Ereignissen führen. Analog zu dem oben genannten Beispiel könnte somit allein durch eine Migration des Prozesses in einen andere Region das Ereignis „Temperaturanstieg um 50%“ ausgelöst werden, ohne dass sich weder bei der Quell- noch bei der Zielausführungeinheit die Temperatur überhaupt verändert hat. Um solche Effekte zu vermeiden ist es notwendig, die Modellierung solcher Ereignisse in ereignisbasierten Kontrollflussstrukturen vor einer Migration zu überprüfen und diese ggf. durch benutzerdefinierte Rahmenbedingungen sinnvoll zu steuern oder einzuschränken.

Wie bereits bei den zuvor beschriebenen Kontrollflussbedingungen ist es dazu vorteilhaft, sich ein im Kontrollfluss des Prozesses modelliertes Ereignis

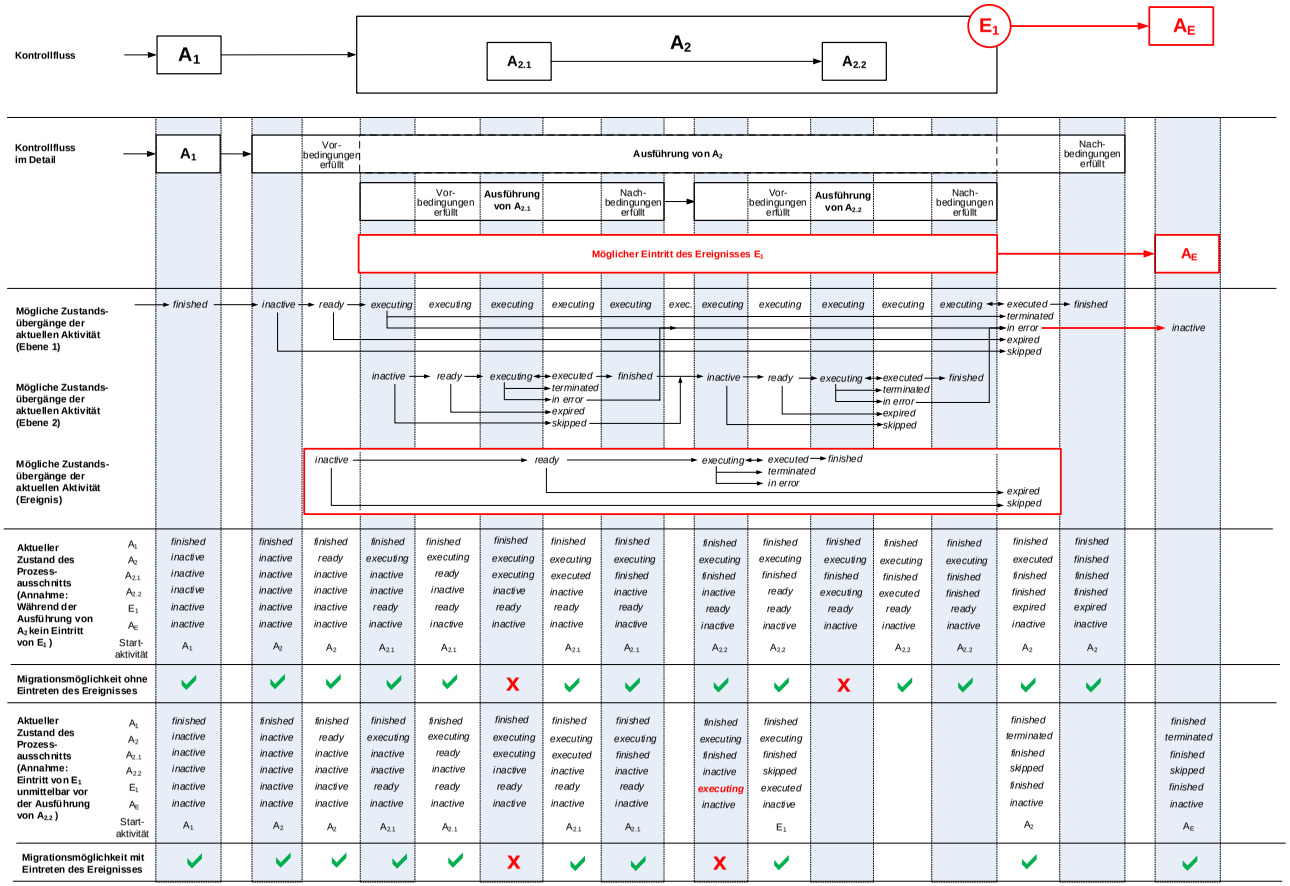


Abbildung 6.21: Migrierbarkeit innerhalb eines ereignisbasierten Kontrollflusses mit Ereignis E_1 und Ereignisbehandlung A_E für die Aktivitäten im Gültigkeitsraum der Blockaktivität A_2 (Beispiel: Eintritt von Ereignis E_1 während der Ausführung von $A_{2.1}$)

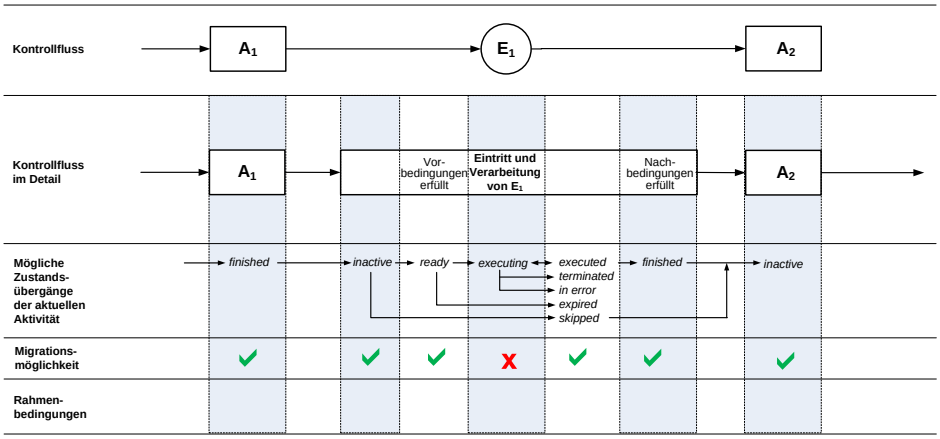


Abbildung 6.22: Migrierbarkeit innerhalb eines ereignisbasierten Kontrollflusses mit Ereignis E_1

als eine besondere Art von Aktivität vorzustellen, welche durch die Prozess-Engine selbst ausgeführt wird. Dabei handelt es sich bei dieser Betrachtungsweise um eine Aktivität, welche auf das Eintreten eines Ereignisses wartet und dieses im Erfolgsfall verarbeitet. Im Fall eines einfachen Ereignisses handelt es sich bei dieser Verarbeitung dabei nur um das einfache Weiterführen des Kontrollflusses. Im Fall eines komplexen Ereignisses muss ggf. auf das Eintreten weiterer Teilereignisse gewartet werden. Die Aktivität befindet sich also im Zustand *executing*, während das Eintreten des relevanten Ereignisses von der Ausführungsumgebung wahrgenommen wird und ggf. auf weitere Teilereignisse gewartet werden muss. Nach dem (vollständigen) Eintreten des Ereignisses und ggf. der Bestimmung nachfolgend auszuführender Aktivitäten ist die „Ausführung“ des Ereignisses beendet. Somit kann der Prozess prinzipiell vor oder nach dem Eintreten eines Ereignisses migriert werden, wobei die Verarbeitung komplexer Ereignisse nicht unterbrochen werden kann. Die Ereignisverarbeitung kann auf diese Weise auch gezielt über die in den Migrationsdaten enthaltenen benutzerdefinierten Rahmenbedingungen einer bestimmten Ausführungsumgebung zugewiesen werden.

Handelt es sich um ein *global definiertes*, d. h. ein vom restlichen Kontrollfluss des Prozesses losgelöstes Ereignis (vgl. Abschnitt 2.4.4), so kann die Ausführung des Prozesses direkt nach dem Eintreten des spezifizierten Ereignisses mit dem angegebenen Kontrollfluss fortgesetzt bzw. ersetzt werden. Bei *lokal definierten*, d. h. direkt in den Kontrollfluss eingebetteten Ereignissen (vgl. Abschnitt 2.4.4) muss zusätzlich die Navigation des Prozesses bis zum entsprechenden Kontrollflusselement fortgeschritten sein, bevor das als Aktivität betrachtete Ereignis aktiviert werden kann (vgl. Abbildung 6.22). Sollen auch persistente Ereignisse berücksichtigt werden, so ist hierfür bei der Migration ein entsprechender Eintrag in der Logdatei vorzunehmen, falls ein Ereignis bereits eingetreten ist, bevor der Kontrollfluss das lokal definierte Ereignis erreicht hat.

Prozessexterne Ereignisse erfordern zudem eine besondere Behandlung in Bezug auf das Abonnement von Ereignissen, da Ereignisse prinzipiell auftreten können, während der Prozess migriert wird. Hierbei darf die Migration des Prozesses also entweder erst erfolgen, wenn die Zielausführungseinheit sich für alle erforderlichen Ereignisse registriert hat, oder die Quellausführungseinheit muss solange für alle Ereignisse registriert bleiben, bis die Migration vollständig abgeschlossen ist, um die Ereignisse im Bedarfsfall „nachzusenden“. In beiden Fällen ist ein erweitertes Koordinationsprotokoll erforderlich, um zu gewährleisten, dass Ereignisbehandlungsroutinen bei Eintritt eines Ereignisses nur genau einmal ausgeführt werden.

6.2.7 Sicherheitsmechanismen für migrierte Prozesse

Ein weiterer wichtiger Aspekt im Kontext der Verteilung der Prozessausführung durch die Migration laufender Prozessinstanzen ist die von vielen Autoren bemängelte Durchsetzung relevanter Sicherheitseigenschaften (vgl. Abschnitt 5.5). Nach den Autoren TANENBAUM und STEEN [TS08] sind im Kontext der Code-Migration mit dem Schutz des mobilen Codes und dem Schutz der ausführenden Computersysteme im Allgemeinen zwei übergeordnete Schutzziele zu verfolgen, welche auf Basis der in Abschnitt 4.4 erarbeiteten Anforderungen (insb. Anforderung A_9) auf die Migration von Prozessinstanzen auf Anwendungsebene übertragen werden können. Dabei muss zunächst aus der Sicht des Prozessinitiators sichergestellt werden, dass die Ausführung, die Vertraulichkeit und die Integrität des Prozesses sowie der bei der Migration mitgeführten Daten und Anweisungen nicht durch böswillige Ausführungseinheiten gefährdet werden. Zum anderen muss sichergestellt werden, dass migrierte Prozesse fremder Ausführungseinheiten nicht unberechtigt auf die lokalen Ressourcen der eigenen Ausführungseinheit zugreifen können. Es werden daher im Folgenden für beide Teilgebiete geeignete Sicherheitsmaßnahmen in Bezug auf Vertraulichkeit, Authentifizierung und Autorisierung untersucht bzw. erarbeitet.

Schutz des zu migrierenden Prozesses

Wie auch für mobile Agenten ist ein allumfassender Schutz für migrierte Prozesse im Allgemeinen nicht umsetzbar, da nicht vollständig sichergestellt werden kann, dass eine Prozesspartition von einer fremden Ausführungseinheit korrekt interpretiert, genau ein Mal vollständig und im Sinne des Prozessinitiators ausgeführt und anschließend an die vorgegebenen Ausführungseinheiten weitergegeben wird (vgl. [FGS96, TS08]). Die genannten Probleme treten jedoch im Kontext verteilter prozessorientierter Anwendungen nicht nur bei der Migration einzelner Prozessinstanzen auf, sondern stellen auch für die anderen genannten Verteilungsmodelle (vgl. Abschnitt 4.3.3) und somit auch für bereits zur Entwicklungszeit verteilte Prozesspartitionen potentielle Sicherheitsrisiken dar [vRZ07]. Es werden daher im Folgenden nur diejenigen Sicherheitsanforderungen betrachtet werden, welche durch

eine Migration von Prozessinstanzen *zusätzlich* erforderlich sind. Im Wesentlichen betrifft dies die Vertraulichkeit und die Integrität von denjenigen Prozesspartitionen, welche der aktuellen Ausführungseinheit nicht zur Ausführung zugeordnet sind und nur aufgrund der Migration der Gefahr von unerlaubten Zugriffen ausgesetzt sind.

Zunächst ist es hierbei wesentlich, dass der hier betrachteten Flexibilität eine Migration des gesamten Prozesses an beliebige (geeignete) Ausführungseinheiten zugrunde liegt. Wenn es aus diesem Grund vermieden werden soll, den Prozess physikalisch zu zerteilen, so ist eine bedarfsgerechte Aufteilung des Prozesses zur Umsetzung des *Need-to-know*-Prinzips nicht mehr durchführbar. Durch die Migration des gesamten Prozesses werden jedoch dessen gesamten Informationen potentiell allen beteiligten (ggf. externen) Ausführungseinheiten bekannt und unterliegen dem Risiko von beabsichtigten oder unbeabsichtigten Veränderungen. Zur Durchsetzung von grundlegenden Sicherheitseigenschaften wie Vertraulichkeit und Integrität ist es daher nicht mehr durchgängig möglich, weiterhin auf Veränderungen der ursprünglichen Prozessbeschreibung zu verzichten. Ein einheitlicher Schutz des Prozesses durch kryptographische Maßnahmen birgt jedoch hinsichtlich der Möglichkeiten zur dynamischen Verteilung große Einschränkungen. Es muss daher zunächst geklärt werden, welche zu schützenden Objekte innerhalb des Prozesses und der weiteren Migrationsdaten existieren und welchen Rahmenbedingungen der Zugriff auf diese Objekte unterliegt.

In Hinblick auf Vertraulichkeit und Integrität des Prozesses können aufbauend auf den in Kapitel 2 beschriebenen Grundlagen insbesondere die folgenden zu schützenden Objekte identifiziert werden [ZKML10, ZHKK10]:

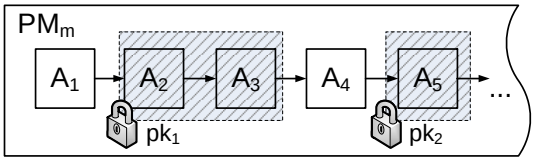
- **Werte von Prozessvariablen:** Jede Prozessinstanz kann vertrauliche Informationen in Form von Variablenwerten beinhalten, welche nur von bestimmten Ausführungseinheiten eingesehen werden dürfen (z. B. Kreditkarteninformationen). Findet im Rahmen der dynamische Verteilung des Prozesses eine Migration der Prozessinstanz auf andere Ausführungseinheiten statt, so darf daher jeweils nur die für die jeweilige Aufgabe zugewiesene Ausführungseinheit auch die dazugehörigen Eingabedaten lesen. Eine Änderung von Variablenwerten sollte zudem generell nur von der Ausführungseinheit vorgenommen werden dürfen, welche eine Aktivität ausführt, die das Schreiben der betreffenden Variablen erfordert. Dabei ist es jedoch durchaus möglich, dass verschiedene Ausführungseinheiten bei der Bearbeitung der ihnen zugewiesenen Prozesspartition nacheinander oder auf verschiedenen Replikaten des Prozesses eine Variable lesen oder verändern müssen.

- **Statische Kontrollflussinformationen:** In einer Prozessbeschreibung können vertrauliche Kontrollflussinformationen vorliegen, welche bestimmten Ausführungseinheiten nicht bekannt werden sollen. Ein Beispiel ist die Nutzung der Ergebnisdaten einer Aktivität als Eingabeparameter von Nachfolgeaktivitäten. So kann es z. B. unerwünscht sein,

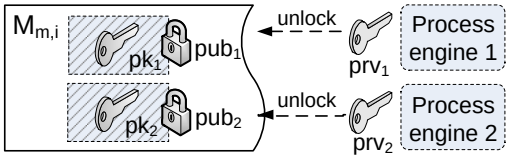
dass ein Endkunde Kenntnis über die Weiterleitung seines Auftrags an einen Subunternehmer erlangt. Allgemein kann dabei sowohl das fachliche Wissen über Art und Inhalt der Aktivitäten als auch der sachliche Zusammenhang zwischen den Aktivitäten schützenswert sein. Für nicht berechtigte Ausführungseinheiten muss daher der Zugriff auf diese Informationen eingeschränkt werden. Im Gegensatz zu Variablenwerten ist bei Prozessen mit fest vorgegebener Struktur jedoch in der Regel keine Bearbeitung von statischen Kontrollflussinformationen erforderlich. Zur Laufzeit prinzipiell unveränderliche Objekte wie Variablennamen, Variablenzuordnungen, Aktivitätsnamen und -beschreibungen, Referenzen zu fest vorgegebenen Ressourcen, Transitionen sowie Bedingungskonstrukte und Ereignisdefinitionen sollten daher generell vor beabsichtigten oder unbeabsichtigten Modifikationen geschützt werden.

- **Verweise auf Ausführungseinheiten:** Ein besonderer Aspekt der Prozessmigration nach dem in Abschnitt 6.2.2 dargestellten Konzept ist die Einschränkung der Verteilung auf fest vorgegebene Ausführungseinheiten. Die hierfür auszuwertenden Rahmenbedingungen sind in den Migrationsdaten beschrieben und steuern auf diese Weise die Verteilung des Prozesses. Diese Angaben dürfen nicht unbemerkt modifiziert werden, da hierdurch der vorgesehene verteilte Ausführungsverlauf des Prozesses gestört werden kann. Eine vollständige Geheimhaltung dieser Informationen birgt jedoch die Gefahr, dass der Prozess nicht an die angegebene Zielausführungseinheit weitergeleitet werden kann, falls der aktuellen Ausführungseinheit die Identität des nachfolgenden Teilnehmers nicht bekannt werden soll. Ein Beispiel für eine solche Situation ist gegeben, wenn die nacheinander beteiligten Ausführungseinheiten verschiedenen konkurrierenden Organisationen angehören. In diesem Fall muss bei der Migration zunächst ein Umweg über einen vertrauenswürdigen Teilnehmer eingelegt werden, was jedoch zusätzlich in den Migrationsdaten modelliert werden muss.

Um die Anforderungen hinsichtlich Vertraulichkeit und Integrität durchzusetzen, muss der Zugriff auf die zu schützenden Objekte so eingeschränkt werden, dass nur den in den Migrationsdaten fest oder rollenbasiert zugeordneten Ausführungseinheiten (vgl. Abschnitt 6.2.2) der Inhalt von vertraulichen Informationen bekannt werden kann und dass unautorisierte Änderungen an den Prozessdaten nicht unbemerkt vorgenommen werden können. Die Zuordnung von zu schützenden Objekten des Prozesses und den berechtigten Ausführungseinheiten sollte dabei möglichst feingranular vorgenommen werden können, um eine möglichst hohe Flexibilität für die Verteilung des Prozesses zu ermöglichen [ZKML10, ZHKK10]. Der hier gewählte Ansatz umfasst daher eine benutzerdefinierte „Zensur“ vertraulicher Prozessinformationen durch die Anwendung kryptographischer Verfahren auf der Ebene der einzelnen Kontrollflusskonstrukte. Somit können mit dem in Abschnitt 6.2.2 vorgestellten Migrationsmodell Sicherheitsrichtlinien (*Security Policies*) sowohl



(a) Verschlüsselung einzelner Prozesspartitionen auf Aktivitätsebene



(b) Distribution des Sitzungsschlüssels

Abbildung 6.23: Veranschaulichende Darstellung der Abbildung von Sicherheitsrichtlinien (Beispiel) (ZKML10)

auf der Ebene des Prozesses als auch auf der Ebene einzelner Aktivitäten und Prozessvariablen vorgegeben werden und – bei Bedarf – globale Sicherheitsrichtlinien der Prozessebene auf Aktivitäts- oder Variablenebene detailliert oder überschrieben werden. Die grundlegende Idee eines solchen Ansatzes ist beispielhaft in Abbildung 6.23a dargestellt.

Der Einsatz von kryptographischen Verfahren zum Schutz der Prozessbeschreibung und ihrer Instanzdaten kann auf Basis bestehender Sicherheitsmechanismen geschehen, erfordert jedoch die Existenz einer sicheren Kommunikationsinfrastruktur, wie zum Beispiel einer *PKI* (*Public Key Infrastructure*) (vgl. [Sch07a]). Wie oben erläutert wurde, muss dabei jedoch berücksichtigt werden, dass sowohl Daten als auch Aktivitäten in verschiedenen (ggf. übergeordneten) Aktivitäten (wieder)verwendet oder referenziert werden können und somit potentiell durch verschiedene Ausführungseinheiten zugegriffen werden müssen. Es ist daher zu restriktiv, die Werte von Prozessvariablen oder die Beschreibung des Kontrollflusses von Prozesspartitionen direkt mit dem öffentlichen Schlüssel pub_i einer einzelnen autorisierten Zielausführungseinheit E_i zu verschlüsseln [ZKML10, ZHKK10].

Basierend auf der Schlüsselverteilung für *Broadcast-Encryption*-Verfahren (vgl. [LNP02]), welche das Senden von verschlüsselten Informationen an eine privilegierte Menge von Empfängern erlauben, können auch für migrierte Prozesse die für die Entschlüsselung notwendigen Schlüssel zusammen mit dem geschützten Inhalt gesendet werden, ohne dass eine weitere Kommunikation zur Authentifizierung oder Autorisierung stattfinden muss. Dabei werden in dem hier verwendeten Ansatz die zu schützenden Prozesselemente zunächst durch ein symmetrisches Verschlüsselungsverfahren mit verschiedenen individuellen Sitzungsschlüsseln (*Session Keys*) verschlüsselt. Das Beispiel in Abbildung 6.23a zeigt die Verschlüsselung der Prozesspartition A_2 bis A_3 mit dem Sitzungsschlüssel pk_1 und die Verschlüsselung der Aktivität A_5 mit dem Sitzungsschlüssel pk_2. Der Sitzungsschlüssel wird nun wiederum selbst (bei Be-

darf in mehrfacher Ausfertigung) mit dem öffentlichen Schlüssel pub_i der autorisierten Zielausführungseinheit(en) E_i verschlüsselt. Entsprechend enthalten die Migrationsdaten innerhalb der Sicherheitsrichtline für ein geschütztes Prozesselement (vgl. *Security Policy* in Abbildung 6.6) somit eine Anzahl verschlüsselter symmetrischer Schlüssel (z. B. die Schlüssel pk_1 und pk_2 in Abbildung 6.23b), welche jeweils mit den privaten Schlüsseln prv_i der autorisierten Zielausführungseinheit E_i zugegriffen werden können [ZKML10, ZHKK10].

Es können somit verschiedene berechtigte Ausführungseinheiten auf das durch den gleichen Sitzungsschlüssel geschützte Prozesselement lesend zugreifen, ohne dass eine weitere Interaktion zwischen dem Prozessinitiator und den Ausführungseinheiten notwendig ist (vgl. [BCF01]). Zudem ist bezüglich der zu sendenden Datenmenge die redundante Bereitstellung mehrerer verschlüsselter Sitzungsschlüssel im Vergleich zu einer mehrfachen Verschlüsselung des gesamten Inhalts vorteilhaft. Beide Eigenschaften machen den Ansatz daher insbesondere auch für einen Einsatz im Umfeld mobil ausgeführter Prozesse ohne dauerhafte Netzwerkanbindung und geringen Datenübertragungsraten anwendbar [ZL10].

Neben der Vertraulichkeit von Prozessinformationen stellt vor allem die Integrität des Prozesses und seiner Daten ein wesentliches Schutzziel dar. Hierbei erweist sich der Ansatz der nicht-intrusiven Prozessmigration als vorteilhaft, da die Prozessbeschreibung selbst (bei vorliegender fixer Struktur des Prozesses) während der Ausführung nicht mehr geändert wird. Diese kann somit bei Bedarf durch das Erzeugen eines konventionellen Prüfteils (*Message Authentication Code*) und einer digitalen Signatur in ihrer Gesamtheit durch den Prozessinitiator vor einer unbemerkten Manipulation geschützt werden. Im Fall einer vorliegenden PKI kann daher jede an der Prozessausführung teilnehmende Ausführungseinheit anhand des öffentlichen Schlüssels des Prozessinitiators die Authentizität der Prozessbeschreibung nachprüfen.

Erlaubte Modifikationen der Prozessinstanz betreffen vornehmlich die Migrationsdaten, insbesondere die Werte von Variablen (*Current Value*), die aktuellen Werte der Prozess- und Aktivitätszustände (*Process State* und *Activity State*) sowie die Auswahlstrategien zur Zuordnung von Ausführungseinheiten (*Selection Type*) (vgl. Abschnitt 6.6). Sind diese Daten von einer Ausführungseinheit aktualisiert worden, so ist für jedes geänderte Datum vor der Durchführung einer Migration ein neuer Prüfteil zu bilden und unter Verwendung eines Zeitstempels digital zu signieren [ZKML10, ZHKK10]. Etwaige Logdateien können durch die Realisierung sogenannter *nur anfügbarer Logs* (*Append-Only Logs*) geschützt werden (vgl. [KT01b]). Es kann somit während der Prozessausführung eine ausreichende Nachvollziehbarkeit der Änderungen gewährleistet werden.

Als letzter Betrachtungsgegenstand stellt der Prozess selbst in seiner Gesamtheit einen schützenswerten Vorgang dar, der nicht beabsichtigt oder unbeabsichtigt während der Ausführung gelöscht, verzögert oder von einer Ausführungseinheit „gefangen“ gehalten werden darf. Die ordnungsgemäße Ausführung nach den Vorgaben des Prozessinitiators kann dabei nur durch

zusätzliche Überwachungsmaßnahmen kontrolliert werden, wobei das Ausmaß der Überwachung und der hierdurch entstehende Aufwand in der Regel mit dem Nutzen der Verteilung des Prozesses in Konflikt stehen (vgl. [vRZ07]). Ein Überblick möglicher Überwachungsmaßnahmen für verteilt ausgeführte Prozesse im Allgemeinen wird in Abschnitt 6.3 vorgestellt.

Schutz der lokalen Ausführungsumgebungen

Neben dem Schutz des Prozesses stellen bei einer Migration laufender Prozessinstanzen auch die lokalen Ausführungseinheiten und ihre Ressourcen wichtige zu schützende Objekte dar. Im Gegensatz zu mobilen Agenten oder Prozessen auf Betriebssystemebene ist es bei den hier betrachteten Prozessen auf Anwendungsebene jedoch in der Regel nicht möglich, beliebigen Programmcode zur Ausführung zu bringen und somit zum Beispiel unberechtigt auf das Dateisystem einer Ausführungsumgebung zuzugreifen. Es können lediglich die für die lokale Prozess-Engine explizit bereitgestellten Schnittstellen zu Anwendungssystemen genutzt werden, welche durch die Spezifikation von lokalen Sicherheitsrichtlinien für verschiedene Prozesse noch zusätzlich eingeschränkt werden können. So existieren in der Regel bereits Schutzmechanismen für (eigene) nicht-verteilte Prozesse, welche den Zugriff nicht-autorisierter Prozessinitiatoren auf sensible Daten oder teure Ressourcen überprüfen und ggf. verhindern können. Eine Identifikation von Prozessen und deren Zuordnung zu Prozessinitiatoren kann dabei zum Beispiel über digitale Signaturen erfolgen und an eine entsprechende *lokale Sicherheitsrichtlinie* geknüpft werden.

Ein Beispiel einer auch für mobile Systeme geeigneten Infrastruktur zur Spezifikation von lokalen Sicherheitsrichtlinien ist der Arbeit von KOTTKE [Kot10] zu entnehmen. Hierbei ist es zum Beispiel möglich, durch eine feingranularen Vergabe von Zugriffsberechtigungen eine anbieterseitige Einteilung von Diensten in private und öffentliche Ressourcen oder eine direkte Zuordnung von Dienstnutzern zu den angebotenen Funktionalitäten vorzunehmen. Zugriffsrechte auf einen geschützten Dienst können dabei prinzipiell über das Konzept der *Rechtedelegation* auch mit dem Prozess und den in den Migrationsdaten enthaltenen Zugriffsinformationen migriert werden und somit den Initiator des Prozesses beim Zugriff auf einen Dienst als autorisierten Nutzer identifizieren (vgl. [Kot10, ZHKK10]). Falls eine Ressource aufgrund von Zugriffsbeschränkungen nicht zugegriffen werden darf, kann durch den hier vorgestellten Verteilungsansatz unter Umständen ein Rebinding auf eine öffentliche Ressource oder eine erneute Migration des Prozesses zu einer Ausführungseinheit mit Zugriff auf eine öffentlichen Ressource erfolgen.

Ein letzter Aspekt ist der kontrollierte Zugang von Prozessen zur Prozess-Engine selbst, d. h. zum Deployment und zur Navigation des Kontrollflusses. Selbst wenn in der lokalen Ausführungsumgebung für die Ausführung eines Prozesses keine (fachlichen) Ressourcen zur Verfügung stehen, da diese durch entsprechende Sicherheitsmechanismen geschützt sind, so besteht den-

noch die Gefahr, dass fremde Prozesse die Rechen- und Speicherkapazitäten der Prozess-Engine ungewollt in Anspruch nehmen oder im Rahmen von *Denial-of-Service*-Attacken für die Ausführung eigener Prozesse unverfügbar machen. Eine einheitliche Behandlung der Prozess-Engine als zu schützende Ressource kann erreicht werden, indem die Funktionalität der (verteilten) Prozessausführung als Dienst im Sinne einer dienstorientierten Architektur angesehen und über eine öffentliche Dienstschnittstelle angeboten wird, welche – ebenso wie die funktionalen Dienste – durch entsprechende Zugriffsbeschränkungen geschützt ist. Eine detaillierte Betrachtung des erweiterten dienstorientierten Paradigmas für verteilt ausgeführte Prozesse erfolgt im folgenden Abschnitt.

6.2.8 Das PMaaS-Paradigma für verteilt ausgeführte Prozesse

In Abschnitt 4.2.6 wurde die Bereitstellung von Diensten zur Prozessautomatisierung im Rahmen des Modells *Process-Management-as-a-Service* (*PMaaS*) als Anwendungsfall für eine dynamische Verteilung von Prozessinstanzen diskutiert. Die Kapselung der Funktionalität zur Ausführung von Prozessen erlaubt jedoch prinzipiell nicht nur die Bereitstellung einer vorliegenden Hard- und Software für die (kostenpflichtige) Ausführung einzelner (nicht-verteilter) Prozesse, sondern kann durch eine entsprechende Erweiterung auch als Basisarchitektur für eine dynamisch verteilte Ausführung durch Migration einzelner Prozessinstanzen genutzt werden [ZL10]. Das resultierende Modell eines verteilten *Process-Management-as-a-Service* wird in diesem Abschnitt erläutert und gegenüber anderen dienstorientierten Ansätzen zur Prozessautomatisierung und -verteilung abgegrenzt. Abbildung 6.24 visualisiert die Evolution der grundlegenden Paradigmen in Richtung einer solchen verteilten Ausführung.

Basierend auf der ursprünglichen Sichtweise einer dienstorientierten Architektur stellen Prozesse als Dienstkompositionen selber (komplexe) Dienste dar, welche analog zu nicht-prozessorientierten (atomaren) Diensten über eine Dienstschnittstelle aufgerufen werden können (vgl. Abbildung 6.24a). Die Aufrufparameter des Dienstes stellen dabei die Eingabeparameter für die Prozessausführung einer einzelnen Prozessinstanz nach dem durch den Dienst gekapselten Prozessmodell dar (vgl. Abschnitt 3.4). Die Verteilung des Prozesses führt daher zu einer physikalischen Partitionierung des Prozessmodells mit jeweils eigenen Dienstschnittstellen, die aus den resultierenden Prozessfragmenten aufgerufen werden können (vgl. Abbildung 6.24b). Im Fall des *Process-Management-as-a-Service* (*PMaaS*) wird die Funktionalität der Prozessausführung als Dienst angeboten. Eine als Dienst angebotene Ausführungsseinheit stellt dabei eine Ressource dar, welche über eine zu den oben genannten komplexen oder atomaren Diensten technisch äquivalente Schnittstelle aufgerufen werden kann. Dabei stellt das auf der angebotenen Ausführungseinheit auszuführende Prozessmodell den Aufrufparameter des Dienstes dar (vgl. Abbildung 6.24c). Die entsprechende ausführbare Prozessbe-

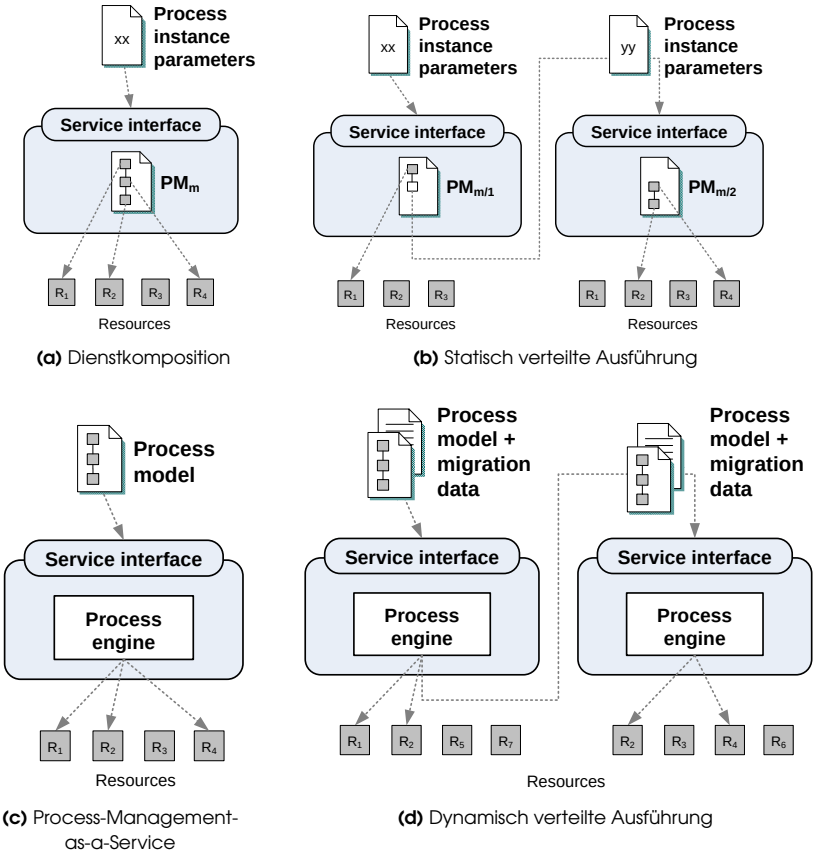


Abbildung 6.24: Dienstorientierte Ansätze zum (verteilten) Prozessmanagement

schreibung ist daher in einem festgelegten Format an den Dienst zu senden, um den Prozess dort zu installieren und von diesem Zeitpunkt an für beliebige Instantiierungen bereitzuhalten (vgl. Abschnitt 4.2.6). Im Gegensatz dazu adressiert das in dieser Arbeit vorgeschlagene *PMaaS*-Modell für eine *dynamisch verteilte Prozessausführung* die Übergabe einzelner (laufender) Prozessinstanzen mit ihren aktuellen Zustandsdaten. Eine Ausführungseinheit wird dabei als Dienst gekapselt, welcher analog zum *Process-Management-as-a-Service* das Prozessmodell als Ausführungsanweisung für die Prozessinstanz und zusätzlich die Instanz der dazugehörigen Migrationsdaten nach dem in Abschnitt 6.2.2 eingeführten Migrationsmodell als Aufrufparameter entgegennimmt (vgl. Abbildung 6.24d). Tabelle 6.1 zeigt eine zusammenfassende Klassifikation der verschiedenen Dienstobjekte.

Die Entkopplung der fachlichen Prozessbeschreibung und der Verteilungsinformationen kann analog zur separaten Verwaltung von fachlichen Regeln aufgefasst werden, welche eine flexible Erweiterung und Anpassung von Entscheidungsvorgängen erlauben, um den Kontrollfluss des Prozesses zu steuern (vgl. Abschnitt 3.6). Bei der dynamischen Verteilung der Prozessausführung kann das (unveränderte) Prozessmodell zur Ausführung des funktionalen Prozesses genutzt werden, während die separaten Migrationsdaten eine benutzer-

	Dienstkomposition		Software-as-a-Service	
	Komplexer Dienst	**Statisch verteilte Ausführung**	**BPM-as-a-Service**	**Dynamisch verteilte Ausführung**
Gekapseltes Objekt:	Prozessmodell	(Teil-)Prozessmodell	Prozess-Engine	Prozess-Engine
Funktion der Schnittstelle:	Instantiierung von Prozessmodellen	Instantiierung von (Teil-)Prozessmodellen	Deployment von Prozessmodellen	Ausführung von Prozessinstanzen
Aufruf-parameter:	Eingabeparameter für Prozessinstanz	Eingabeparameter für (Teil-)Prozessinstanz	Prozessmodell	Prozessmodell und Migrationsdaten
Abstraktions-ebene:	Prozessmodell	(Teil-)Prozessmodell	Prozessmodell	Prozessinstanz

Tabelle 6.1: Vergleich dienstorientierter Ansätze zum (verteilten) Prozessmanagement

definierte Verteilung des Prozesses steuern. Die fachliche Funktionalität und die Verteilung des Prozesses sind somit angemessen voneinander entkoppelt.

Die Migration einer laufenden Prozessinstanz von einem *PMaaS*-Anbieter zum nächsten erfordert jedoch (im Gegensatz zu einer statischen Verteilung der Prozessausführung) die Konzeption von unterstützenden Softwarekomponenten, welche die Funktionalität der gekapselten Prozess-Engine um die Möglichkeit zur Migration des Prozesses erweitern. Dies umfasst Komponenten zur Interpretation und Auswertung von Migrationsdaten, zum Auffinden, Auswählen und Aufrufen von geeigneten Ausführungseinheiten sowie zum Anhalten und zur Wiederaufnahme laufender Prozessinstanzen. Eine prototypische Implementierung solcher Komponenten ist Inhalt von Kapitel 7.5.

Da bei der Kapselung einer Ausführungseinheit die Parameter (d. h. Prozessmodell und Instanzdaten) jeweils einen komplexen Datentyp besitzen, ist es für den Austausch von Prozessmodellen und Instanzdaten nach dem vorgestellten Migrationsmodell wesentlich, dass den teilnehmenden Parteien ein hinreichend gemeinsames Verständnis über das zur Laufzeit ausgetauschte Prozessmodell als Basis für eine äquivalente Interpretation und Verarbeitung zugrunde liegt. Hierfür ist die Verwendung einer bereits eingesetzten, standardisierten Prozessbeschreibungssprache von Vorteil. Eine Anwendung des hier vorgestellten Konzepts auf bestehende Prozessbeschreibungssprachen, deren Ausführung durch individuelle Prozessmanagementsysteme für sowohl stationäre und mobile Umgebungen sowie der Flexibilitätsgewinn gegenüber zur Entwicklungszeit physikalisch partitionierten Prozessen werden daher in Kapitel 8 eingehender untersucht.

6.2.9 Zusammenfassung und Einordnung

Um zur Erhöhung der Flexibilität bei der Verteilung von individuellen und ggf. bereits gestarteten Prozessinstanzen beizutragen, wurde in diesem Abschnitt ein Ansatz zur nicht-invasiven Migration von laufenden Prozessinstanzen vorgestellt. Ausgehend von einer vollständigen Freiheit für die Migration eines Prozesses wurden hierbei sinnvolle Einschränkungen der Flexibilität zur Gewährleistung der Korrektheit der Prozessausführung und zur Berücksichtigung fachlicher oder qualitativer Vorgaben des Prozessmodellie-

rers, Prozessinitiators oder Prozessteilnehmers erarbeitet. Ergebnisse sind daher zum einen ein abstraktes Modell für die Migration von Prozessinstanzen, welche den Anforderungen eines einfachen Prozessmetamodells genügen, und die Anwendung des Migrationsmodells auf mögliche Kontrollflussstrukturen prozessorientierter Anwendungen. Zum anderen wurde eine Einordnung des Konzepts der Migration laufender Prozessinstanzen in bestehende Entwicklungsmethodiken dienstorientierter Anwendungen vorgestellt und zur Realisierung einer dynamisch verteilten Ausführung eine Erweiterung des *Process-Management-as-a-Service*-Modells propagiert.

Der Gedanke einer als Dienst gekapselten Prozess-Engine wird im Folgenden weiter thematisiert und für die Überwachung und Steuerung von Prozessen – unabhängig vom zugrunde liegenden Verteilungsmodell – nutzbar gemacht. Die daraus resultierenden Möglichkeiten zur Abfrage von verteilungsrelevanten Informationen und zur Steuerung einer entfernt ausgeführten Prozessinstanz können wiederum dynamisch in die Entscheidung über eine Verteilung einfließen und somit die Migration einzelner Prozessinstanzen wesentlich beeinflussen. Der Ansatz zur nicht-invasiven Migration von laufenden Prozessinstanzen kann somit als flexible Basisstruktur angesehen werden, um Anpassungen an der Verteilung eines Prozesses aufgrund von wahrgenommenen Umgebungsänderungen zu erlauben. Die im Folgenden diskutierte Bereitstellung und Nutzung einer Prozess-Engine als (verwaltbare) Ressource ermöglicht darüber hinaus die Erhebung und Bereitstellung der für die Migrationsentscheidung relevanten Informationen.

6.3 Überwachung und Steuerung verteilt ausgeführter Prozesse

Unabhängig vom zugrunde liegenden Verteilungsmodell besteht bei einer verteilten Prozessausführung ein großer Bedarf, entfernt ausgeführte Prozesspartitionen möglichst zur Laufzeit zu überwachen und – bei Auftreten einer als Ausnahme identifizierten Situation – geeignet auf die Prozessausführung einwirken zu können (vgl. Abschnitte 3.5, 3.7 und 3.8). Analog zur Integration von Verteilungsinformationen wurde in Abschnitt 5.7 festgestellt, dass eine feste Integration von Überwachungs- und Steuerungsmechanismen in das Prozessmodell die Flexibilität beim Zugriff auf individuell relevante Informationen und ein individuelles Eingreifen in die Prozessausführung stark begrenzt. Es wird daher in diesem Abschnitt ein nicht-invasives Konzept vorgeschlagen, welches auf Basis der anbieterseitigen Bereitstellung relevanter Managementfunktionalitäten sowohl eine kontinuierliche Beobachtung relevanter Geschehnisse zeitnah zu ihrem Auftreten als auch eine spontane Abfrage und Anpassung der Menge von individuell und situativ relevanten Informationen erlaubt.

Kern des hier vorgeschlagenen Konzepts ist die Abbildung eines *Prozessmanagementsystems als verwaltbare Ressource* (vgl. Abschnitt 3.8 und [vLLM+08]), so dass die für eine verteilte Prozessausführung allgemein relevanten Informationen auf Basis eines einfachen gemeinsamen Verständnisses

in einheitlicher Art und Weise bereitgestellt und zugegriffen werden können. Dementsprechend erarbeitet dieser Abschnitt ein abstraktes (Referenz-)Modell eines Prozessmanagementsystems, dessen Entitäten und deren Eigenschaften durch eine dienstbasierte Management-Komponente als Erweiterung bestehender Ausführungssysteme in individuell verhandelbarer Art und Weise von den teilnehmenden Kooperationspartnern zugegriffen werden können. Ziel ist es dabei, die Überwachungs- und Steuerungsmechanismen weitestgehend von der fachlichen Ausführung der Prozesse zu entkoppeln und somit fortwährend in Art und Umfang anpassbar und erweiterbar zu halten, damit auch spontane Reaktionen und der Umgang mit erst zur Laufzeit auftretenden Anforderungen ermöglicht werden können.

Nach einer kurzen Beschreibung der wesentlichen Prinzipien und Entwicklungsgrundsätze eines solchen Ansatzes (vgl. Abschnitt 6.3.1) wird zunächst auf Basis der in Kapitel 2 und 4 erarbeiteten Erkenntnisse ein geeignetes Referenzmodell für Prozessmanagementsysteme zur Überwachung und Steuerung verteilt ausgeführter Prozesse vorgestellt (vgl. Abschnitt 6.3.2). Darauf aufbauend wird eine dienstbasierte Schnittstelle vorgeschlagen, welche die Kapselung eines Prozessmanagementsystems nach dem oben genannten Referenzmodell als *verwaltbare Ressource* und den geordneten Zugriff auf die angebotenen Managementfunktionalitäten erlaubt (vgl. Abschnitt 6.3.3). Um den Kommunikationsaufwand für Überwachungs- und Steuerungsmaßnahmen möglichst gering zu halten und auch das Offline-Management verteilt ausgeführter Prozesse nach individuellen Vorgaben zu unterstützen, werden schließlich ein prozessbasierter Ansatz (vgl. Abschnitt 6.3.4) und ein Ansatz zur regel- und ereignisbasierten Überwachung und Steuerung der Prozessausführung vorgestellt (vgl. Abschnitt 6.3.5). Als Ergebnis kann die Überwachung und Steuerung von verteilt ausgeführten Prozessen sowohl über eine dauerhafte Netzwerkverbindung echtzeitnah zur Ausführung des Prozesses als auch automatisiert für Prozesse außerhalb der direkten Kommunikationsreichweite (z. B. im Kontext mobiler Ausführungseinheiten) vorgenommen werden. Eine kurze Zusammenfassung und Einordnung der erarbeiteten Teilkonzepte erfolgt in Abschnitt 6.3.6.

6.3.1 Prinzipien und Entwicklungsgrundsätze

In diesem Abschnitt stehen – unabhängig vom zugrunde liegenden Verteilungsmodell – die Überwachung und Steuerung von inhaltlich festgelegten Prozesspartitionen im Vordergrund, welche nicht im direkten Einflussbereich des Prozessinitiators ausgeführt werden. Zur Vereinfachung werden diese Prozesspartitionen im Folgenden als *entfernt ausgeführte Prozesspartitionen* bezeichnet. Die eine Prozesspartition auslagernde Partei wird analog zum *Process-Management-as-a-Service*-Gedanken (vgl. Abschnitt 6.2.8) als *Auftraggeber* der entfernten Prozessausführung bezeichnet. Den Anbieter des Dienstes stellt in diesem Fall die *prozessausführende Partei* dar.

Werden Prozesse ganz oder teilweise durch die Ausführungssysteme externer Kooperationspartner verwaltet, so stehen dem Auftraggeber bzw. dem Prozessinitiator der Prozessausführung bislang oft nur begrenzte Möglichkeiten zur Überwachung und Steuerung des Prozesses zur Verfügung (vgl. Abschnitt 5.7). Dabei kann ohne zusätzlichen Entwicklungsaufwand in der Regel nur das von außen beobachtbare Verhalten der Prozessausführung (wie etwa der Start einer Prozessinstanz, die Rückgabe von Ergebnissen oder die Gesamtdauer der Prozessausführung) als Grundlage für die Überwachung solcher Prozesse herangezogen werden. In der Regel kann weder das interne Verhalten des Prozesses beobachtet werden, noch kann auf die in Ausführung befindlichen Prozesse von außen steuernd eingewirkt werden.

In Zusammenhang mit der Kapselung von Softwarefunktionalitäten in Form von elektronischen Diensten (vgl. Abschnitt 3.4) hat dieses Verbergen von internen Details den Vorteil, dass die Implementierung eines Dienstes jederzeit flexibel – und für den Dienstkonsumenten transparent – durch den Dienstanbieter angepasst werden kann. Wird jedoch ein bestehender Prozess im Sinne dieser Arbeit verteilt ausgeführt, so ist die Struktur des Prozesses dem Auftraggeber der Prozessausführung bekannt und den teilnehmenden Ausführungseinheiten in Form des Prozessmodells fest vorgeschrieben. Es besteht daher eine hinreichende Basis zum Austausch von Informationen über die Ausführung von einzelnen Prozessinstanzen und zu deren individueller Steuerung. Im Folgenden werden die wichtigsten Rahmenbedingungen für eine Bereitstellung von Informations- und Steuerungsfunktionalitäten zusammengefasst und entsprechende Entwurfsentscheidungen für den Lösungsansatz dieser Arbeit begründet.

Vorüberlegung

Im Gegensatz zu den oben genannten elektronischen Diensten, die – für den Dienstnutzer transparent – entweder eine atomare Softwarefunktionalität oder einen strukturierten Prozess auf Anwendungsebene implementieren, stehen bei der Überwachung entfernt ausgeführter Prozesspartitionen insbesondere die ansonsten verborgenen internen Details der Prozessausführung im Vordergrund. Beispiele hierfür sind die Bearbeitungszeiten einzelner Aktivitäten, die Identität der ausführenden Ressourcen oder die (zwischenzeitlichen) Werte von Prozessvariablen. Die in Abschnitt 4.4 genannten Anforderungen an Möglichkeiten zur kontinuierlichen Überwachung (Anforderung $D_{4.1}$) macht die Verwendung einer ereignisbasierten Infrastruktur vorteilhaft (*Push*-Modell). Die Anforderung nach einer individuellen Abfrage von Informationen im Bedarfsfall (Anforderung $D_{4.2}$) erfordert die (zusätzliche) Unterstützung eines *Pull*-Modells. Für die Beeinflussung entfernt ausgeführter Prozesse werden zudem geeignete Steuerungsfunktionalitäten benötigt (Anforderung $D_{6.1}$). Der Bedarf an den genannten Funktionalitäten kann dabei von Prozessinstanz zu Prozessinstanz variieren und unterliegt somit selbst einer hohen Dynamik [vRZ07].

In dieser Arbeit wird davon ausgegangen, dass die benötigten Informationsfunktionalitäten einem interessierten Beobachter bei einer *nicht-verteilten* Ausführung desselben Prozesses über eine entsprechende Monitoring-Schnittstelle der lokalen Prozess-Engine prinzipiell bereitgestellt werden können. Analog dazu existieren in der Regel softwaretechnisch realisierte Steuerungsmechanismen für Prozess-Engines, welche es einem lokalen Administrator erlauben, auf die Ausführung eines Prozesses einzuwirken, z. B. diesen zwischenzeitlich anzuhalten, zu verändern oder gänzlich abzubrechen. Bei einer verteilten Ausführung eines Prozesses hindert jedoch eine vollständige Kapselung solcher *Managementfunktionalitäten* den Auftraggeber der Prozessausführung an der Auswertung der für eine Evaluierung und ggf. Optimierung „seines“ Prozesses notwendigen Informationen. Eine uneingeschränkte Bereitstellung dieser Aspekte stellt jedoch auf der anderen Seite einen starken Eingriff in die Autonomie der ausführenden Organisation bzw. Organisationseinheit dar, welche unter Umständen private Aspekte der Prozessausführung (wie zum Beispiel monetäre Kosten für die Bezahlung von Ressourcen) vor einem Zugriff schützen möchte. Die individuellen Interessen des Beobachters stehen damit prinzipiell mit den individuellen Interessen der prozessausführenden Partei in Konflikt [vRZ07, Zap07].

Im Allgemeinen lassen sich vor diesem Hintergrund verschiedene Stufen der Überwachung und Steuerung entfernt ausgeführter Prozesspartitionen identifizieren. Dabei kann der notwendige Aufwand zur Durchführung der Überwachung bzw. Steuerung mit zunehmender Stufe als wachsend angesehen werden [vRZ07]:

- **1. Keine Überwachung:** Der Auftraggeber der entfernt ausgeführten Prozesspartition verwendet die Ausführungseinheit des externen Kooperationspartners wie eine Black-Box. Er übergibt das auszuführende Prozessmodell bzw. die auszuführende Prozessinstanz und vertraut auf eine ordnungsgemäße Ausführung. Die Verantwortung für die Erkennung und Behandlung von Fehlern oder anderen Ausnahmesituationen wird damit vollständig an die entfernte Ausführungseinheit abgegeben. Es entsteht kein zusätzlicher Aufwand für die teilnehmenden Parteien.

- **2. Bestätigung:** Der Auftraggeber der entfernt ausgeführten Prozesspartition lässt sich die (erfolgreiche) entfernte Ausführung des Prozesses bestätigen. Dies ist insbesondere von Vorteil, wenn der Kontrollfluss einer entfernt ausgeführten Prozesspartition nicht zur aufrufenden Partei zurückkehrt bzw. keine Rückgabewerte erwartet werden. Obwohl hierbei keine direkte Beobachtung oder Kontrolle der Prozessausführung vorgenommen werden kann, können die erhaltenen Bestätigungen verwendet werden, um die Verantwortlichkeit einer prozessausführenden Partei für die Ausführung einer bestimmten Prozesspartition festzuhalten und etwaige im (weiteren) Prozessverlauf festgestellte Fehler einem Verursacher zuzuordnen. Es entsteht hierbei Aufwand für das Senden bzw. Empfangen und Verarbeiten der Bestätigungen.

- **3. Überwachung durch fachliche Kontrollflusskonstrukte:** Die fachliche Prozessbeschreibung kann bereits Kontrollflusskonstrukte enthalten, welche das Verhalten des Prozesses im Fall einer spezifizierten Ausnahme vorschreiben. Beispiele sind Zeitbeschränkungen oder das Überprüfen von Rückgabewerten auf einen erlaubten Wertebereich. Wird die betrachtete Prozesspartition entfernt ausgeführt, so ist die aktuell verantwortliche Ausführungseinheit an den jeweils vorgegebenen Prozessverlauf gebunden. Fachliche Kontrollflusskonstrukte zur Überwachung und Steuerung müssen in der Regel zur Entwicklungszeit des Prozessmodells integriert werden. Es entsteht daher zusätzlicher Aufwand zur Entwicklungszeit des Prozesses und ebenfalls zur Laufzeit, in der die zusätzlichen Kontrollflusskonstrukte ausgeführt werden müssen. Es besteht zudem die Gefahr, dass fachliche und technische Anweisungen zum Management des (verteilten) Prozesses innerhalb der Prozessbeschreibung vermengt werden.

- **4. Richtlinien und Vereinbarungen:** Nicht-funktionale Aspekte der Prozessausführung können durch die zusätzliche Vereinbarung von Dienstgütekriterien außerhalb der funktionalen Prozessbeschreibung spezifiziert werden. Dabei können ebenfalls Maßnahmen für den Fall der Nichteinhaltung dieser Dienstgütekriterien definiert werden. Die Überwachung der Dienstgüte zur Laufzeit des Prozesses und die Durchführung ggf. notwendiger Anpassungsmaßnahmen obliegt jedoch der prozessausführenden Partei. Es entsteht zudem Aufwand für eine der Prozessausführung vorgelagerte Verhandlung der Vereinbarung.

- **5. Entfernte Überwachung und Steuerung:** Hierbei werden durch die prozessausführende Partei vor, während oder nach der Laufzeit einzelner Prozessinstanzen detaillierte Informationen über die Prozessausführung und/oder das betrachtete Prozessmanagementsystem versendet oder zur Abholung bereitgestellt. Der Austausch von Informationen kann entweder direkt zwischen einem Beobachter der Prozessausführung und der aktuell verantwortlichen Ausführungseinheit oder über eine ansonsten unbeteiligte dritte Partei erfolgen. Bei einer anbieterseitigen Bereitstellung von Informationen kann jedoch ein maliziöses Verhalten der prozessausführenden Partei in Form der absichtlichen Bereitstellung von Falschinformationen nicht vollständig ausgeschlossen werden. Es entsteht Aufwand für eine einmalige Einrichtung von Schnittstellen zur Entwicklungszeit des Prozessmanagementsystems und für den Austausch von Informationen sowie für deren Verarbeitung zur Laufzeit.

- **6. Unmittelbare Kontrolle:** In diesem Fall wird die Ausführung eines Prozesses direkt am Ausführungsort durch den Auftraggeber der Prozessausführung oder einen unabhängigen Dritten überwacht bzw. beeinflusst. Die softwaretechnische Umsetzung eines solchen *Vier-Augen-Prinzips* für die Ausführung bedeutet für die teilnehmenden Organisa-

tionen einen hohen Aufwand und stellt einen starken Eingriff in die Autonomie der prozessausführenden Partei dar.

Die Untersuchung dieser sechs Stufen [vRZ07, Zap07] zeigt, dass eine flexible Lösung für die Überwachung entfernt ausgeführter Prozesspartitionen erreicht werden kann, indem die genannte Stufe 5 (*Entfernte Überwachung und Steuerung*) so anpassbar gestaltet wird, dass sich auch die Überwachungsmechanismen der Stufen 1 bis 4 damit hinreichend abbilden lassen. Basierend auf dieser Überlegung kann eine Erfüllung der Anforderungen nach Dezentralität und Selbstbestimmung erfolgen, wenn die genauen Modalitäten der Überwachung und Steuerung der entfernten Prozessausführung durch den Auftraggeber und die prozessausführende Partei direkt verhandelbar sind [Zap07]. Interne Details der Prozessausführung können hierbei freiwillig von der prozessausführenden Partei zur Verfügung gestellt werden. Der Grad der Bereitstellung von Ereignissen und Abfrageschnittstellen zur Überwachung sowie die Bereitstellung von Steuerungsfunktionalitäten stellt dabei wiederum ein nicht-funktionales Kriterium zur Wahl einer geeigneten Ausführungseinheit dar (vgl. auch [WKK+10]). In einem kommerziellen Szenario mit Wettbewerb zwischen verschiedenen Ausführungseinheiten ist es daher für eine prozessausführende Partei vorteilhaft, ihren potentiellen „Kunden“ eine möglichst hohen Grad an Überwachungs- und Steuerungsmöglichkeiten bereitzustellen.

Leitbild

Eine Analogie zu dieser Betrachtungsweise stellt die in der Logistik oft zusätzlich zur funktionalen Dienstleistung des Pakettransports angebotene Dienstleistung der *Paketverfolgung*[1] dar [Zap07] (vgl. Abbildung 6.25). Hierbei stellt der Transport von Paketen einen fest strukturierten (Teil-)Prozess dar, der nach den Vorgaben eines Prozessinitiators durch die Angabe von Start- und Zielort des Transports individualisiert wird. Die entstehende Prozessinstanz eines konkreten Pakettransports kann durch den Prozessinitiator und beliebige weitere Prozessteilnehmer (im Wesentlichen Versender und Empfänger des Pakets) durch die Kenntnis eines eindeutigen Identifikators (*Sendungsnummer*) im Zeitraum der Prozessausführung überwacht werden. Dazu werden dem Beobachter die Schritte des Prozesses (z. B. Einlieferung des Pakets, Transport an Zwischenstationen und Auslieferung an den Empfänger) mit ihrem jeweiligen Status und den Bearbeitungszeiten angezeigt. Ein autorisierter Beobachter dieses Prozesses kann sich somit einen Überblick über den Fortschritt der Prozessausführung und (über zusätzliche Erfahrungswerte) die voraussichtliche Fertigstellung des Vorgangs verschaffen. Durch die Bereitstellung zusätzlicher Informationen, wie z. B. dem Namen und der Anschrift des Empfängers, können zudem mögliche Fehler der Prozessausführung erkannt und behandelt werden. Der Status des Pakettransports kann dabei

[1] Darstellung nach der Sendungsverfolgung der *Deutschen Post DHL*, Variante *ProView*, Stand vom 1. November 2011.

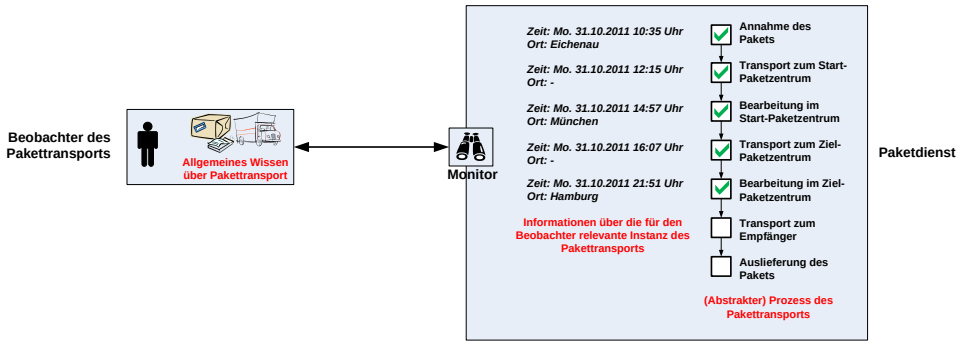


Abbildung 6.25: Konzept der Paketverfolgung als Überwachung des Vorgangs *Pakettransport*

ereignisbasiert erfolgen (z. B. durch das Senden einer E-Mail oder SMS bei Statusänderungen) oder über eine webbasierte Schnittstelle bei Bedarf jederzeit abgefragt werden. Ein weitergehendes Szenario ermöglicht sogar die Beeinflussung der Prozessinstanz durch die Änderung von Prozessvariablen. So kann z. B. nachträglich ein benutzerdefinierter Zeitpunkt für die Endauslieferung spezifiziert werden. Die Daten über abgeschlossene Pakettransporte stehen dem autorisierten Beobachter über einen längeren Zeitraum zur Verfügung. Dies erlaubt einen Aufbau von Erfahrungswerten und darüber die Möglichkeit zur Einschätzung aktueller und zukünftiger Pakettransporte.

Ziel des hier vorgestellten Ansatzes ist es, die Idee der Paketverfolgung auf entfernt ausgeführte Prozesse im Allgemeinen zu übertragen und für eine Automatisierung von Anpassungsmaßnahmen nutzbar zu machen. Bei der Paketverfolgung erfolgt eine Abstraktion von individuellen, für den Beobachter unwesentlichen Details des Pakettransports und somit eine Reduktion auf ein abstraktes Modell des Anwendungsfalls. Obwohl der tatsächliche Vorgang des Pakettransports komplexer ist als die dem Benutzer dargestellte Repräsentation dieses Anwendungsfalls, so ist das gewählte Abstraktionsniveau in den meisten Fällen ausreichend, um die Anforderungen an eine entfernte Überwachung des Pakettransports zu erfüllen und die entfernte Ausführung somit hinreichend nachvollziehbar zu gestalten. Der Paketverfolgung liegt somit eine Art reduzierter Prozessbeschreibung des Pakettransports zugrunde.

Abbildung 6.26 zeigt analog dazu das Konzept einer (dezentralen) Überwachung durch die Bereitstellung von Funktionalitäten zum Zugriff auf entfernte Prozessmanagementsysteme. Zum einen kann dadurch das aktuell prozessausführende System nach individuell verhandelbaren Vorgaben überwacht und ggf. gesteuert werden. Zum anderen können die öffentlichen Eigenschaften weiterer (potentiell) prozessausführender Systeme beobachtet und als Grundlage einer Entscheidung über die weitere Verteilung des Prozesses zur Laufzeit integriert werden. Durch die Betrachtung einer Ausführungseinheit als Ressource (bzw. als Dienst) kann das Problem der Auswahl von Ausführungseinheiten schließlich auf die Auswahl von Diensten in dynamischen Umgebungen zurückgeführt werden.

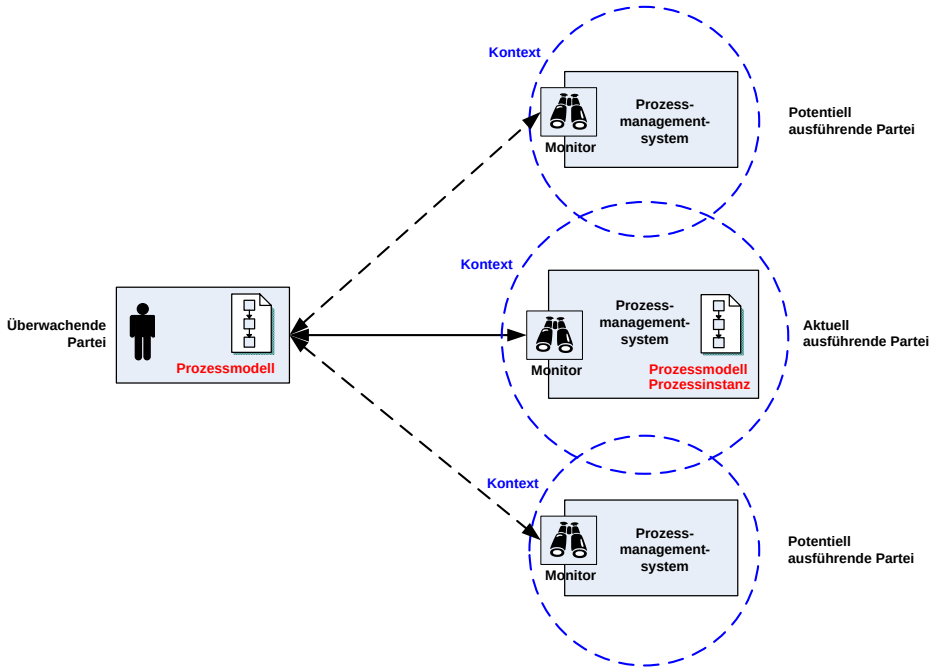


Abbildung 6.26: Dezentrale Überwachung durch Bereitstellung von Informationen durch (potentielle) Kooperationspartner

Im Rahmen der entfernten Prozessausführung im Allgemeinen ist eine Abstraktion des jeweiligen Anwendungsfalls nicht notwendig, da diese bereits durch das vorliegende Prozessmodell gegeben ist. Das einer Überwachung und Steuerung zugrunde liegende Abstraktionsniveau ist also durch das Abstraktionsniveau der Prozessbeschreibung determiniert und der Anwendungsfall dem Auftraggeber einer (verteilten) Prozessausführung somit hinreichend bekannt. Es ist daher nur zu identifizieren, wie abstrakte Informationen zu den Prozessen im Allgemeinen ausgetauscht werden können und anhand welcher Informationen eine entfernte Prozessausführung unabhängig vom jeweiligen Anwendungsfall oder in Kombination mit dem separaten Wissen über den jeweiligen Anwendungsfall bewertet werden kann. Dabei steht jedoch nicht nur der Prozess selbst, sondern auch sein aktueller und zukünftiger *Ausführungskontext* im Zentrum der notwendigen Betrachtungen (vgl. Abschnitt 3.5). Es werden daher geeignete Referenzmodelle für Prozessmodelle, Prozessinstanzen und Ausführungseinheiten (Prozess-Engines) benötigt, um Informationen über die Prozessausführung und deren Umwelt bereitstellen und abfragen zu können [ZBH+10]. Ein auf diese Anforderungen ausgerichtetes Referenzmodell wird in Abschnitt 6.3.2 vorgestellt.

Entwicklungsmethodik

Basierend auf diesen Überlegungen kann die Vorgehensweise zur Überwachung und Steuerung entfernt ausgeführter Prozesspartitionen wie in den Abbildungen 6.27 und 6.28 zusammengefasst werden: Für eine zu-

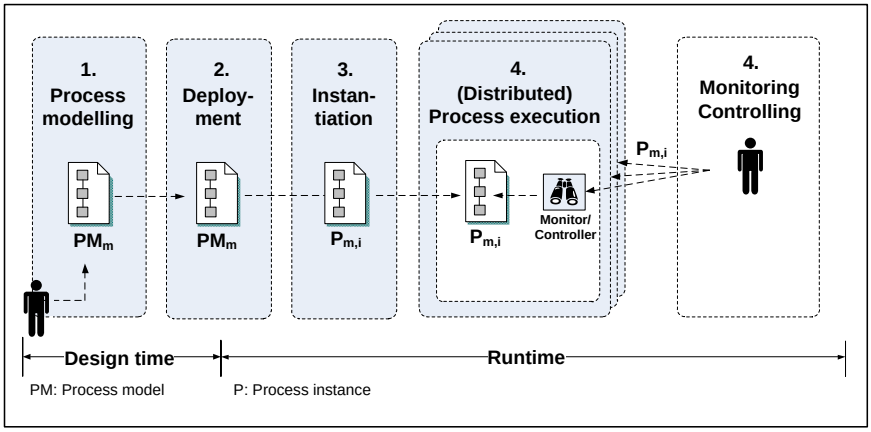


Abbildung 6.27: Ad-hoc Überwachung und Steuerung entfernt ausgeführter Prozesspartitionen durch anbieterseitig bereitgestellte Schnittstellen

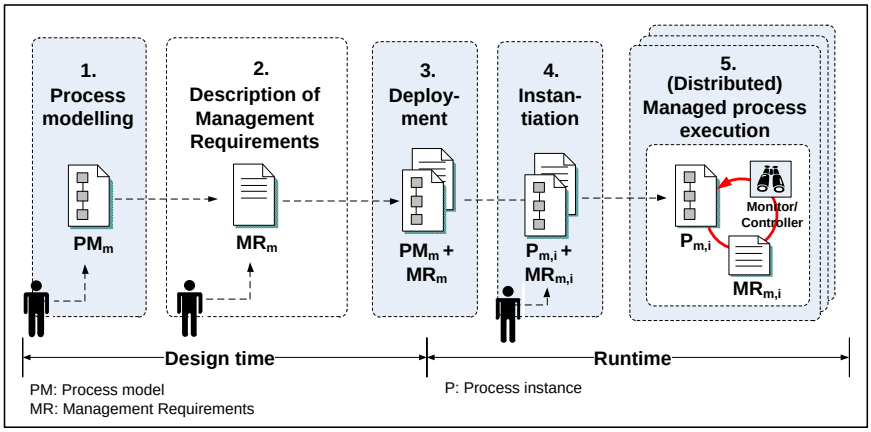


Abbildung 6.28: Automatische Überwachung und Steuerung entfernt ausgeführter Prozesspartitionen durch Spezifikation von Management-Anforderungen

vor unvorhergesehene Überwachung und/oder Steuerung, welche sich erst zur Laufzeit des Prozesses ergibt, sind zur Zeit der Entwicklung, des Deployments und der Instantiierung von Seiten des Auftraggebers keine vorbereitenden Maßnahmen erforderlich. Erst zur Laufzeit der betroffenen Prozessinstanz werden die durch die ausführende Partei bereitgestellten Schnittstellen zur Durchführung der relevanten Überwachungs- oder Steuerungsmaßnahmen in Anspruch genommen (vgl. Abbildung 6.27). Dazu muss dem Beobachter der entfernten Prozessausführung der eindeutige Identifikator der entfernt laufenden Prozessinstanz bekannt sein. Unter der Voraussetzung einer dauerhaften Netzwerkverbindung zwischen beobachtender und prozessausführender Partei können Überwachungs- und Steuerungsmaßnahmen damit ad-hoc durchgeführt werden.

Ist die Voraussetzung einer dauerhaften Netzwerkanbindung nicht erfüllt oder sind die durchzuführenden Überwachungs- und Steuerungsmaßnahmen bereits im Vorfeld der (verteilten) Prozessausführung bekannt, so können die-

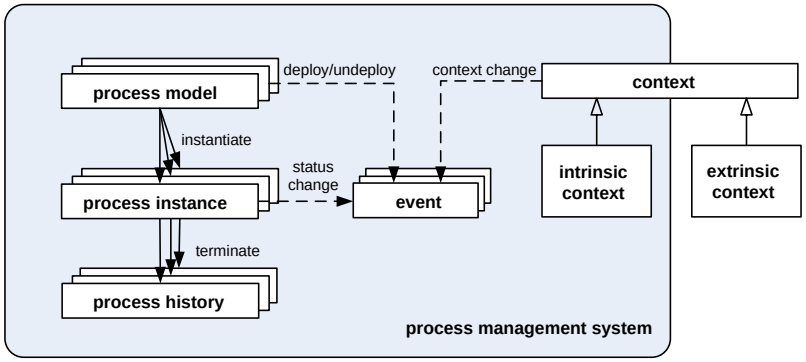


Abbildung 6.29: Referenzmodell für verteiltes Prozessmanagement (Überblick) (ZBH+10)

se formalisiert und während der Prozessausführung auch ohne ein manuelles Eingreifen des Beobachters durchgeführt werden (vgl. Abbildung 6.28). In Bezug auf das Leitbild der Paketverfolgung entspricht dies einer maschinellen Verarbeitung der Anforderungen des Paketkunden. Zwei mögliche Ansätze hierfür werden in den Abschnitten 6.3.4 und 6.3.5 vorgestellt.

6.3.2 Referenzmodell für verteiltes Prozessmanagement

Basierend auf den abstrakten Eigenschaften von Prozessmanagementsystemen im Allgemeinen (vgl. Kapitel 2), den Eigenschaften und Anforderungen verteilt ausgeführter Prozesse (vgl. Kapitel 4) und einer Analyse verschiedener praktischer und theoretischer Ansätze, bestehender abstrakter Modelle und eingesetzter Produkte zum Prozessmanagement (vgl. [Wun09]) wurden die für eine Überwachung und Steuerung verteilter Prozesse relevanten Entitäten identifiziert und in ein gemeinsames Modell integriert. Dieses Modell repräsentiert die den meisten Prozessmanagementsystemen gemeinsamen Aspekte mit besonderem Hinblick auf ihre Relevanz für eine Überwachung und Steuerung entfernt ausgeführter Prozesspartitionen. Im Rahmen einer verteilten Prozessausführung können die Ergebnisse dieser Analyse somit als Referenzmodell für die Klasse der Prozessmanagementsysteme aufgefasst werden, welche die Funktionalität der Prozessausführung sowie die Überwachung und Steuerung der entfernt ausgeführten Prozesse als funktionale Dienste anbieten [ZBH+10]. Das Referenzmodell stellt zugleich die Basis für die Abbildung eines Prozessmanagementsystems als *verwaltbare Ressource* dar (vgl. Abschnitt 6.3.3).

Die wesentlichen Entitäten des erarbeiteten Referenzmodells sind in Abbildung 6.29 im Überblick dargestellt. Dabei beinhaltet das übergeordnete Element *Prozessmanagementsystem* (*Process Management System*) eine endliche Anzahl von *Prozessmodellen* (*Process Model*), die auf dem betrachteten Ausführungssystem installiert (*deployed*) sind. Jede Instantiierung eines Prozessmodells führt zu der Erzeugung einer neuen *Prozessinstanz* (*Process Instance*) als Repräsentation eines zur Betrachtungszeit auf diesem Ausführungssystem laufenden Prozesses. Nach Beendigung einer Pro-

zessinstanz werden die entsprechenden Ausführungsdaten in einer *Prozesshistorie* (*Process History*) festgehalten. Jede Zustandsänderung an den Prozessinstanzen sowie die Zustandsübergänge zwischen Prozessmodell, Prozessinstanz und Prozesshistorie erzeugt ein *Ereignis* (*Event*) [ZBH+10]. Unabhängig von den Prozessen besitzt ein Prozessmanagementsystem weitere Eigenschaften, welche in dem dargestellten Referenzmodell unter dem Element *Kontext* (*Context*) zusammengefasst werden. Der Kontext eines Prozessmanagementsystems unterteilt sich in einen *intrinsischen Kontext* (*Intrinsic Context*) und einen *extrinsischen Kontext* (*Extrinsic Context*) [ZBH+10] (vgl. auch [RR06]). Der intrinsische Kontext betrifft Eigenschaften des internen Zustands des Ausführungssystems, wie zum Beispiel die momentane Auslastung der Prozess-Engine. Der extrinsische Kontext repräsentiert beliebige externe Eigenschaften des Ausführungssystems, zum Beispiel dessen geografische Position. Beide Arten von Kontext können statische oder dynamische Attribute besitzen [ZBH+10]. Ein Beispiel für ein statisches Attribut ist die Identität des Besitzers des betrachteten Ausführungssystems. Die Auslastung der Prozess-Engine stellt ein Beispiel für ein dynamisches Attribut des Kontextes dar. Alle Änderungen des relevanten Kontextes führen wiederum zur Erzeugung von Ereignissen. Für die Integration und Verwaltung von generischen Kontextmodellen können bestehende Kontextmanagementsysteme herangezogen werden (vgl. [KZTL08]). Die weitere Detaillierung des (extrinsischen) Kontextes ist daher nicht Inhalt dieser Arbeit.

Die Abbildungen 6.30 und 6.31 zeigen Verfeinerungen der Entitäten *Prozessmodell* und *Prozessinstanz*. Analog zu dem in Abschnitt 6.2.2 dargestellten Metamodell für die Migration von Prozessinstanzen besteht ein Prozessmodell im Allgemeinen aus wenigstens einer *Aktivität* (*Activity*), welche durch *Transitionen* (*Transitions*) bzw. eine entsprechende Schachtelung von Blockstrukturen mit anderen Aktivitäten zu einem vorgegebenen Kontrollfluss verknüpft sein kann. Dieser Kontrollfluss kann optional über *Bedingungen* (*Conditions*) gesteuert werden. Fachliche *Ereignisse* (*Events*) werden losgelöst vom Kontrollfluss betrachtet. Das Prozessmodell definiert zudem eine Menge von *Prozessvariablen* (*Data Fields*), welchen in der Regel ein *Datentyp* (*Data Type*) zugeordnet ist. Zudem kann (optional) zu jeder Aktivität eine Menge von festgeschriebenen Prozessbeteiligten (d. h. ein Individuum, eine Gruppe von Individuen, eine Rolle oder eine Menge von Rollen) definiert sein, welche für die Ausführung dieser Aktivität verantwortlich sein soll (*Predefined Performer*) [ZBH+10].

Das Referenzmodell einer Prozessinstanz erweitert das dargestellte Modell um die dazugehörigen Laufzeitinformationen. Im Vordergrund stehen hierbei Zustandsinformationen auf Prozessebene (*Status*) (vgl. Abschnitt 6.2.2), die aktuellen Werte der Prozessvariablen (*Data Value*), das Ergebnis der Evaluation von Bedingungen (*Evaluation Result*) sowie der Zustand von Aktivitäten und das Eintreten fachlicher Ereignisse. Hierbei ist zu berücksichtigen, dass Aktivitäten und Bedingungen mehrmals ausgeführt bzw. ausgewertet werden können (*Iteration*), zum Beispiel im Rahmen von mehrfach verwendeten Blockaktivitäten oder in Schleifen mit mehreren Durchläufen. Dabei

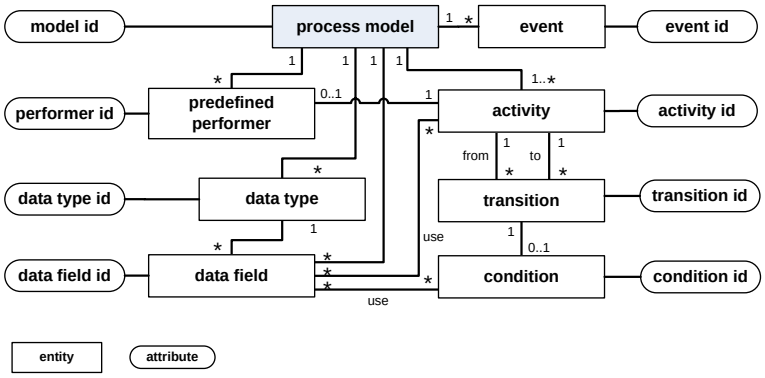


Abbildung 6.30: Referenzmodell für verteiltes Prozessmanagement: Prozessmodell (ZBH+10)

können die Aktivitäten in den verschiedenen Durchläufen durchaus verschiedene Zustände besitzen (*Status*) und die Bedingungen verschiedenartig ausgewertet werden (*Evaluation Result*). Ebenso können Ereignisse mehrfach eintreten (*Occurrence*), wobei insbesondere der Zeitpunkt des Auftretens von Interesse ist. Schließlich werden zur Laufzeit die (ggf. zuvor festgeschriebenen) Prozessbeteiligten konkretisiert bzw. ausgewählt. Somit ist zur Laufzeit jeder bereits ausgeführten Aktivitätsinstanz genau ein Prozessbeteiligter (*Actual Performer*) zugewiesen [ZBH+10].

Die Prozesshistorien spiegeln schließlich die Entitäten der beendeten Prozessinstanz statisch wider. Dabei entspricht der Zustand der Prozesshistorie dem jeweils letzten Zustand ihrer Prozessinstanz [ZBH+10]. Aufgrund ihrer identischen Darstellung sind die Prozesshistorien hier nicht graphisch dargestellt.

Um eine verteilte Ausführung und eine entsprechende Überwachung und Steuerung von entfernt ausgeführten Prozesspartitionen zu ermöglichen, ist es notwendig, dass Prozessmodelle und Prozessinstanzen jeweils einen eindeutigen Identifikator besitzen (*Model Id* bzw *Instance Id*). Werden mehrere Instanzen eines Prozessmodells erzeugt und entfernt ausgeführt, so ist hier ebenfalls eine Zuordnung der Prozessinstanzen zu ihrem Prozessmodell notwendig (*Model Id*). Desweiteren müssen auch die Entitäten des Prozessmodells bzw. der davon abgeleiteten Instanzen über eindeutige Identifikatoren verfügen, um eine sinnvolle Auswertung der individuellen Instanzdaten zu gewährleisten. Die entsprechenden Attribute sind in den Abbildungen 6.30 und 6.31 dargestellt.

Angelehnt an bestehende Ansätze (wie z. B. WfXML [SPG04] (vgl. Abschnitt 5.2.2) können die genannten Entitäten durch weitere Subentitäten verfeinert und mit geeigneten Attributen zur Repräsentation von Eigenschaften versehen werden, deren Werte dem Auftraggeber der Prozessausführung zur Verfügung stehen sollen. Eine mögliche weitere Verfeinerung des dargestellten Referenzmodells kann der Arbeit von Wunderlich [Wun09] entnommen werden.

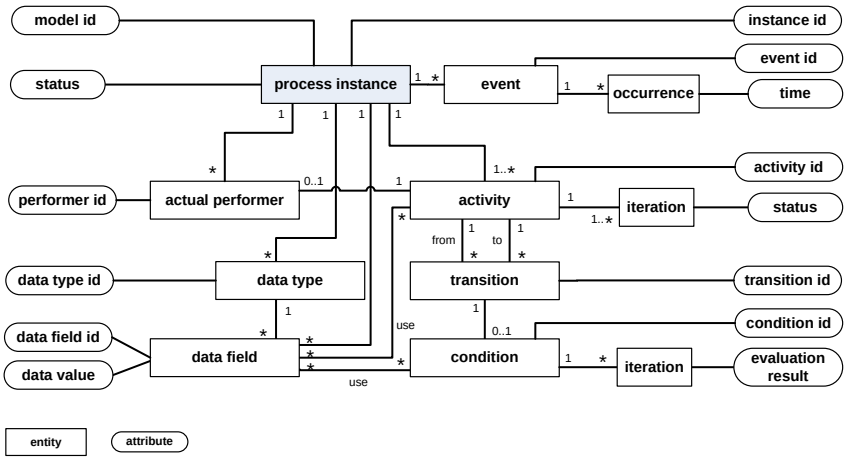


Abbildung 6.31: Referenzmodell für verteiltes Prozessmanagement: Prozessinstanz (ZBH+10)

6.3.3 Definition einer dienstbasierten Management-Schnittstelle

In Abschnitt 3.8.3 wurde das Konzept der *verwaltbaren Ressource* (*Manageable Resource*) als Basis einer flexiblen Bereitstellung von Informationen für die Anpassung von Systemkomponenten vorgestellt. Abbildung 6.32 zeigt ein als verwaltbare Ressource gekapseltes Prozessmanagementsystem nach dem in Abschnitt 6.3.2 dargestellten Referenzmodell. Dabei können die durch die Entitäten des Referenzmodells definierten *Eigenschaften* (engl. *Properties*) über eine dienstbasierte Management-Schnittstelle zugegriffen werden.

Im Kontext verwaltbarer Ressourcen spielt der Begriff der *Eigenschaften* eine wesentliche Rolle. Eine Eigenschaft kann aus der Perspektive des Nutzers der Management-Schnittstelle veränderbare oder unveränderbare Charakteristika oder Parameter der verwaltbaren Ressource umfassen [Wun09, ZBH+10]. Eine ausschließlich auslesbare Eigenschaft des Prozessmanagementsystems bzw. der zugreifbaren Prozessdefinitionen, -instanzen und -historien wird als *unveränderbare Eigenschaft* bezeichnet. Kann eine Eigenschaft durch den Nutzer der Management-Schnittstelle in ihrem Wert anpasst werden, so stellt sie eine *veränderbare Eigenschaft* dar. Die *Veränderbarkeit* einer Eigenschaft durch den Nutzer der Management-Schnittstelle ist jedoch von der allgemeinen Möglichkeit zu tatsächlichen Änderungen (*Veränderlichkeit*) der Eigenschaft abzugrenzen. Eine Eigenschaft wird als *veränderlich* bezeichnet, wenn davon ausgegangen werden kann, dass ein einmalig zu Beginn der Prozessausführung abgefragter Parameter nicht über die gesamte Laufzeit den gleichen Wert aufweist. Ein im Gegensatz dazu statische Eigenschaft wird dementsprechend als *unveränderlich* bezeichnet [Wun09]. Die zuvor genannte Einteilung entspricht den Möglichkeiten zur Modellierung verwaltbarer Ressourcen nach dem *Web Services Resource Metadata Standard* mit den Attributen *Veränderbarkeit* (*Modifiability*) und *Veränderlichkeit* (*Mutability*), welche auch in *WSDM-MUWS* verwendet werden [Wun09] (vgl. Abschnitt 3.8.3) und

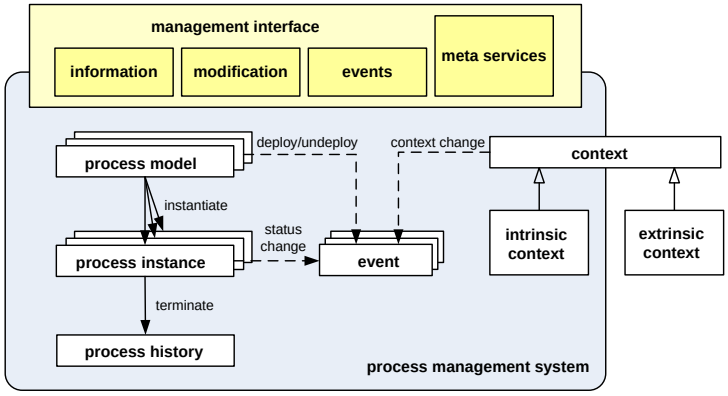


Abbildung 6.32: Prozessmanagementsystem als verwaltbare Ressource (ZBH+10)

somit auch bei individuellen Implementierungen eine standardisierte Darstellung erlauben (vgl. Abschnitt 7.3).

Um eine ressourcenschonende Beobachtung von relevanten, veränderlichen Eigenschaften der verwaltbaren Ressourcen zu erlauben, kann die Veränderlichkeit von Eigenschaften zudem hinsichtlich ihrer Änderungsdynamik klassifiziert werden [Wun09]. Hierbei kann zum Beispiel berücksichtigt werden, dass Konfigurationsparameter des Prozessmanagementsystems in der Regel seltener Änderungen erfahren als Zustandsinformationen laufender Prozessinstanzen [Wun09]. Eine Ermittlung von Wertänderungen mittels zyklischer Abfragen (*Polling*) kann somit an die Wahrscheinlichkeit der Veränderlichkeit angepasst werden. Ebenso kann das Informieren eines Beobachters durch das Senden einer Nachricht im Fall einer Änderung (ereignisbasierter Ansatz, vgl. Abschnitt 3.7) in bestimmten Intervallen oder nur für bestimmte Schwellwerte gewünscht sein, damit es beim Abonnement hochdynamischer Eigenschaften nicht zu einer unerwünschten Flut an Ereignissen kommt [Wun09].

Basierend auf den oben genannten Überlegungen und den in Abschnitt 4.4 festgestellten Anforderungen kann die hier vorgeschlagene Management-Schnittstelle für die Kapselung von Prozessmanagementsystemen als verwaltbare Ressourcen zunächst in vier Komponenten eingeteilt werden [Wun09, ZBH+10] (vgl. Abbildung 6.32):

- **Informationsschnittstelle**: Die Informationsschnittstelle dient ausschließlich zur individuellen Abfrage von unveränderbaren und veränderbaren Eigenschaften durch autorisierte Nutzer der Management-Schnittstelle. Dabei werden durch die Informationsschnittstelle Methodenaufrufe von außen entgegengenommen und in Bezug auf den Zustand der verwalteten Entitäten beantwortet.

- **Manipulationsschnittstelle**: Die Manipulationsschnittstelle dient zur Anpassung von Werten veränderbarer Eigenschaften durch autorisierte Nutzer der Management-Schnittstelle. Dabei werden durch die Manipulationsschnittstelle Methodenaufrufe von außen entgegengenommen und

die dort spezifizierten Änderungen auf den verwalteten Entitäten durchgeführt.

- **Ereignisschnittstelle**: Über die Ereignisschnittstelle werden bei Veränderung einer Eigenschaft auf Basis von zuvor spezifizierten Abonnements Nachrichten an alle Abonnenten des resultierenden Ereignisses gesendet. Es kann zwischen normalen (*nicht blockierenden*) und *blockierenden* Ereignissen unterschieden werden (vgl. [vLLM+08]). Im Fall eines normalen Ereignisses wird (sofern möglich) sofort nach dem Auftreten des Ereignisses mit der Ausführung des fachlichen Prozesses fortgefahren bzw. das Ereignis parallel zur Ausführung ausgelöst. Bei einem blockierenden Ereignis wird bei Auftreten des Ereignisses die Prozessausführung für einen definierten Zeitraum angehalten, um eine Reaktion auf das Ereignis zu erlauben, bevor die Prozessausführung voranschreitet. Ob ein Ereignistyp als blockierend abonniert werden soll, muss beim Abonnement des Ereignisses spezifiziert werden. Blockierende Ereignisse stehen nur auf der Ebene von Prozessinstanzen zur Verfügung.

- **Meta-Schnittstelle**: Die Meta-Schnittstelle stellt übergeordnete Informationen und Steuerungsmöglichkeiten zu den drei zuvor genannten Schnittstellen zur Verfügung. Dies umfasst Informationen zur aktuellen Konfiguration und zu den Fähigkeiten der Management-Schnittstelle (z. B. zu den angebotenen Eigenschaften und Ereignis-Abonnements) sowie Operationen zum Abonnement von Ereignissen, zur Konfiguration von Ereignis-Abonnements und zum Löschen von Abonnements.

Betrachtet man das Prozessmanagement als *verwaltbare Ressourcen* im Rahmen des Autonomic Computings (vgl. Abschnitt 3.8), so sind die *Sensoren* als Informations- und Ereignisschnittstelle und die *Effektoren* als Manipulationsschnittstelle repräsentiert. Von dem dargestellten allgemeinen Referenzmodell können nun einzelne konkrete Systemmodelle mit konkreten *Eigenschaften* abgeleitet werden. Dabei besitzt jede über die Schnittstellen zur Verfügung gestellte Eigenschaft eine Reihe von Attributen, welche eine Interpretation ihres Wertes erlauben (vgl. Abbildung 6.33). Wie bei jedem funktionalen Dienst ist hierfür zunächst die Angabe der zur Darstellung der Information verwendeten *Datenstruktur* (*Data Structure*) und die einer etwaigen Messung zugrunde liegenden *Metrik* (*Metric*) notwendig. Datenstrukturen und Metriken sind plattformabhängig bzw. abhängig von den als relevant betrachteten Kontextdaten. So kann zum Beispiel die Eigenschaft *geographische Position* einer Ausführungseinheit als GPS-Koordinate oder Adresse aus Straße, Hausnummer und Ort dargestellt werden. Bestehen zwischen dem Nutzer der Management-Komponente und dem Anbieter der bereitgestellten Informationen Unterschiede in der Darstellung der Informationen, so muss ggf. eine Transformation der Werte erfolgen. Eine entsprechende Architektur als Teil eines generischen Kontextmanagementsystems wurde in einer diesen Aspekt

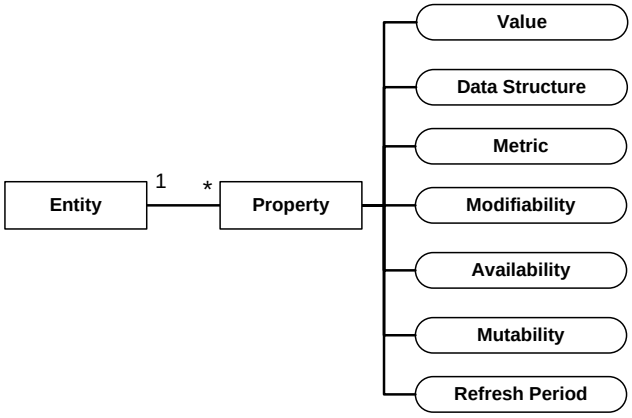


Abbildung 6.33: Attribute der Eigenschaften (Properties) von Entitäten (Entities) eines Prozessmanagementsystems als verwaltbare Ressource

vertiefenden Arbeit [KZTL08] sowie in der entsprechenden Arbeit von KUNZE [Kun08] dargestellt.

Ein Konzept zur Durchsetzung grundlegender Sicherheitseigenschaften in Bezug auf Autorisierung und Authentifizierung von Nutzern solcher dienstbasierter Schnittstellen ist der Arbeit von KOTTKE [Kot10] zu entnehmen. Unter der Voraussetzung, dass ein Zugriff auf die bereitgestellten Funktionalitäten nur für autorisierte Nutzer erfolgt, können die Eigenschaften des Prozessmanagementsystems mit seinen installierten Prozessmodellen, laufenden Prozessinstanzen und abgeschlossenen Prozesshistorien von Seiten der prozessausführenden Partei weiter hinsichtlich ihrer *zeitlichen Verfügbarkeit* (*Availability*), ihrer *Änderungshäufigkeit* (*Mutability*) bzw. *Aktualisierungshäufigkeit* (*Refresh Period*) und ihrer *Änderbarkeit* (*Modifiability*) klassifiziert und so den entsprechenden Schnittstellen zugewiesen werden [ZBH+10].

Mit Hilfe des Attributs *Änderbarkeit* kann für jeden auslesbaren Wert spezifiziert werden, ob ein Überschreiben der Eigenschaft mit einem neuen Wert erlaubt ist. Dabei sind nicht nur diejenigen Eigenschaften zu schützen, welche nicht im Einflussbereich des Auftraggebers der entfernt ausgeführten Prozesspartition liegen sollen, sondern es steht insbesondere die Erhaltung der Konsistenz der Prozessausführung und eine Aussage über die tatsächliche (sinnvolle) Änderbarkeit der Eigenschaften im Mittelpunkt der Betrachtung. Es werden drei mögliche Konfigurationen vorgeschlagen [Wun09]:

- **Read:** Die Eigenschaft kann während des Zeitraums ihrer Verfügbarkeit nur lesend zugegriffen werden. Hierunter fallen alle nicht beeinflussbaren Kontextdaten der Ausführungseinheit sowie Eigenschaften, deren Änderungen Auswirkungen auf Prozesse anderer Auftraggeber verursachen können. Beispiele sind Informationen über Identität, geographische Position und momentane Auslastung der Ausführungseinheit.
- **Read-Write:** Die Eigenschaft kann während des Zeitraums ihrer Verfügbarkeit lesend und schreibend zugegriffen werden. Hierunter fal-

len alle Eigenschaften, die in direktem Zusammenhang mit dem durch den Auftraggeber übergebenen Prozessmodell und den davon abgeleiteten Prozessinstanzen stehen und dadurch exklusiv in dessen Verantwortungsbereich fallen. Ein Beispiel sind Werte von Prozessvariablen einer entfernt ausgeführten Prozesspartition, welche zur Laufzeit gelesen, aber auch noch angepasst werden können.

- **Read-Limited-Write:** Die Eigenschaft kann bis zu einem bestimmten Zeitpunkt lesend und schreibend zugegriffen werden und darf anschließend nur noch gelesen werden. Hierunter fallen alle Eigenschaften, welche bis zu einem bestimmten Zeitpunkt determiniert werden müssen oder aufgrund ihrer Natur ab einem bestimmten Zeitpunkt nicht mehr geändert werden können. So ist z. B. eine Festlegung von Quality-of-Service-Parametern in der Regel nur bis zu Ausführungsbeginn des Prozesses möglich.

In der Annahme, dass jede entfernt ausgeführter Prozesspartition zunächst auf der ausgewählten Ausführungseinheit installiert wird, anschließend instantiiert wird und schließlich ausgeführt wird, kann die *zeitliche Verfügbarkeit* von bestimmten Eigenschaften vom Deployment eines Prozessmodells bis zum Abschluss der Ausführung einzelner Prozessinstanzen eingeschränkt sein. Ein Beispiel ist die Bereitstellung von Prognosen über die Eigenschaft *geschätzte Ausführungsdauer* der Entität Prozessinstanz (vgl. [Wun09]) als Annahme über die Zeitdauer, in der die Prozessinstanz vollständig ausgeführt werden kann. Diese steht vor Kenntnis des auszuführenden Prozesses sowie nach der Beendigung der Prozessausführung nicht (mehr) zur Verfügung. Ebenso ist die Eigenschaft *tatsächliche Ausführungsdauer* der Entität Prozessinstanz (als Zeit, die seit der Instantiierung der Prozessinstanz verstrichen ist) vor der Instantiierung des Prozesses noch nicht verfügbar. Abbildung 6.34 zeigt eine grobe Einteilung der zeitlichen Verfügbarkeit von Eigenschaften anhand der Lebenszyklusphasen von prozessorientierten Anwendungen auf der technischen Ebene (vgl. Abschnitt 2.8). Entsprechend der dargestellten Phasen kann die zeitliche Verfügbarkeit von Eigenschaften die Werte *Vorbereitung* (*Preparation*), *Wartezeit* (*Wait*), *Ausführung* (*Execution*) und *Nachverarbeitung* (*Postprocessing*) oder eine beliebige Kombination dieser Werte annehmen. Es ist zum Beispiel denkbar, dass die oben genannte Eigenschaft *tatsächliche Ausführungsdauer* der Entität Prozessinstanz in den Phasen *Ausführung* und *Nachverarbeitung* verfügbar ist (*Preparation, Execution*) und die Eigenschaft *geschätzte Ausführungsdauer* in den Phasen *Wartezustand* und *Ausführung* (*Wait, Execution*). Der Alias-Wert *Always* dient als verkürzte Form der ständigen Verfügbarkeit einer Eigenschaft, während eine ständige Nicht-Verfügbarkeit über eine fehlende Eigenschaft ausgedrückt wird [Wun09].

Wie bereits eingangs motiviert wurde, können sich Eigenschaften zudem in regelmäßigen oder unregelmäßigen Zeitabständen ändern oder sich weitestgehend statisch verhalten. Die *Änderungshäufigkeit* einer Eigenschaft gibt da-

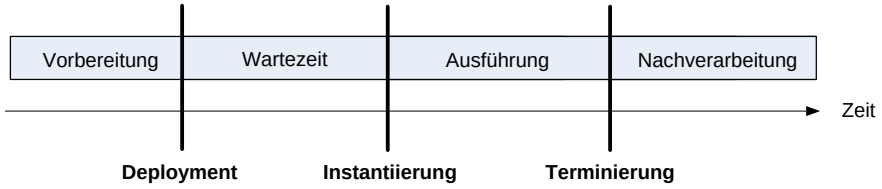


Abbildung 6.34: Phasen der zeitlichen Verfügbarkeit von Eigenschaften (tlw. nach (Wun09))

her an, welcher Dynamik eine Eigenschaft normalerweise unterliegt. Hierbei können auch individuelle Besonderheiten der einzelnen Ausführungseinheiten ausgedrückt werden. Wird zum Beispiel die Eigenschaft *geographische Position* mit einer relativ hohen Änderungshäufigkeit angegeben, so kann davon ausgegangen werden, dass es sich um eine mobile Ausführungseinheit handelt. Hat die Eigenschaft *Auslastung* eine niedrige Änderungshäufigkeit, so besteht weitestgehend Planungssicherheit über die bei der Ausführung entfernt ausgeführter Prozesspartitionen zu erwartenden Verzögerungen. Ebenso kann jedoch auch die prozessausführende Partei die Aktualisierung von bestimmten Eigenschaften begrenzen, um die über die Management-Schnittstelle zusätzlich verursachte Systemlast zu steuern (*Aktualisierungshäufigkeit*). Für den Auftraggeber der entfernt ausgeführten Prozesspartition ist die Unterscheidung zwischen Änderungshäufigkeit und Aktualisierungshäufigkeit in der Regel transparent, da er seine Reaktionen ohnehin nur auf Basis der bereitgestellten Werte vornehmen kann. Weichen Änderungshäufigkeit und Aktualisierungshäufigkeit jedoch stark voneinander ab, kann es zu Schwierigkeiten bei der korrekten Interpretation von Änderungen kommen. Ein Beispiel hierfür ist die nur gelegentliche Aktualisierung einer sich häufig ändernden Eigenschaft *Auslastung*, welche einen falschen Eindruck von Stabilität vermitteln kann.

Basierend auf dem Standard ISO 8601 für die Darstellung von Datum und Uhrzeit für den Informationsaustausch lässt sich eine grobe Klassifikation für die Änderungshäufigkeit bzw. die Aktualisierungshäufigkeit von Eigenschaften ableiten [Wun09]:

- ständig (*always*)
- Sekunden (*seconds*)
- Minuten (*minutes*)
- Stunden (*hours*)
- Tage (*days*)
- Monate (*months*)
- Jahre (*years*)
- niemals (*never*)

- unbekannt (*unknown*)

Eine Aktualisierungshäufigkeit im Bereich von Bruchteilen von Sekunden ist zwar durchaus möglich, ist praktisch jedoch durch den sequentiellen Versand von Nachrichten bzw. die entsprechenden Nachrichtenlaufzeiten nur von eingeschränkter Aussagefähigkeit und wird daher einer *ständigen* Änderung der Eigenschaft gleichgesetzt. Mit den Konfigurationen *Always* und *Never* kann daher festgelegt werden, dass sich der Wert einer Eigenschaft potentiell bei jedem Aufruf der Management-Schnittstelle ändert bzw. in der Regel keinen Änderungen unterliegt [Wun09].

Ein Vorschlag für eine Liste konkreter Eigenschaften der Entitäten des in Abschnitt 6.3.2 dargestellten Referenzmodells sowie die Zuordnung der genannten Attribute kann der Arbeit von Wunderlich [Wun09] entnommen werden. Ein Zugriff auf die bereitgestellten Informationen und Steuerungsmöglichkeiten ist über die dienst- bzw. ereignisbasierte Management-Schnittstelle möglich, wobei die Eigenschaften über jeweils einzelne, spezialisierte Dienste [Zap07] oder nach dem Prinzip von WSDM (vgl. Abschnitt 3.8.3) über einen generischen Dienst zum Auslesen bzw. Manipulieren der Eigenschaften möglich ist [ZBH+10]. Die zweite Möglichkeit bietet ein besseres Potential zur Erweiterung der angebotenen Eigenschaften, da die Dienstschnittstelle hierfür nicht geändert werden muss. Eine mögliche Implementierung zu dieser Variante wird daher in Abschnitt 7.3 vorgestellt.

Die dienstbasierte Management-Schnittstelle kann in jedem Fall durch eine einfache Client-Komponente auf der Seite des Auftraggebers der entfernt ausgeführten Prozesspartition zugegriffen werden. Dabei können prinzipiell beliebig viele Auftraggeber unabhängig voneinander auf die bereitgestellten Funktionen zugreifen, um die Prozesse, auf die sie berechtigt sind, nach individuellen Vorgaben zu beobachten oder zu steuern (vgl. auch [Wun09]). Alternativ können entfernt ausgeführte Prozesspartitionen jedoch auch auf der Seite der prozessausführenden Partei nach benutzerdefinierten Vorgaben individuell verwaltet werden. Im Folgenden werden zwei mögliche Ansätze hierfür vorgestellt.

6.3.4 Prozessbasiertes Management

Die Bereitstellung von Diensten zur Beobachtung und Steuerung entfernt ausgeführter Prozesspartitionen stellt eine flexible Infrastruktur dar, um analog zu der in Abschnitt 6.3.1 dargestellten Paketverfolgung über eine Netzwerkverbindung Informationen und Steuerungsbefehle zwischen der prozessausführenden Partei und dem Auftraggeber der entfernt ausgeführten Prozesspartition auszutauschen. Bisher besteht aber weder die Möglichkeit, die Anpassung der (verteilten) Prozessausführung geeignet zu automatisieren (vgl. Anforderung D_{16} in Abschnitt 4.4), noch eine Verringerung der über das Netzwerk zu übertragenden Managementinformationen zur Unterstützung mobiler Ausführungseinheiten ohne dauerhafte Netzwerkverbindung zu erreichen (vgl. Anforderung D_8 in Abschnitt 4.4).

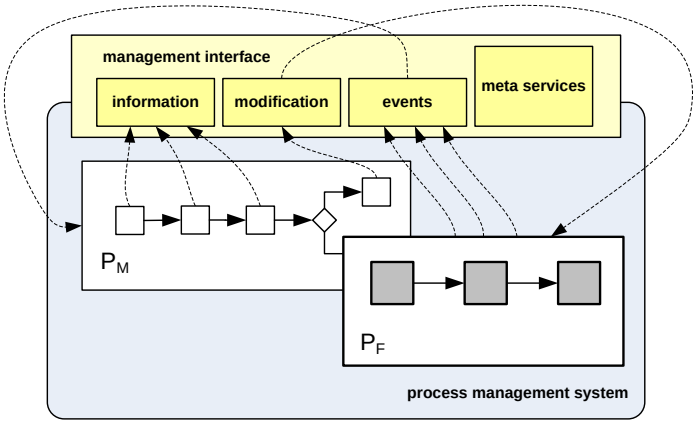
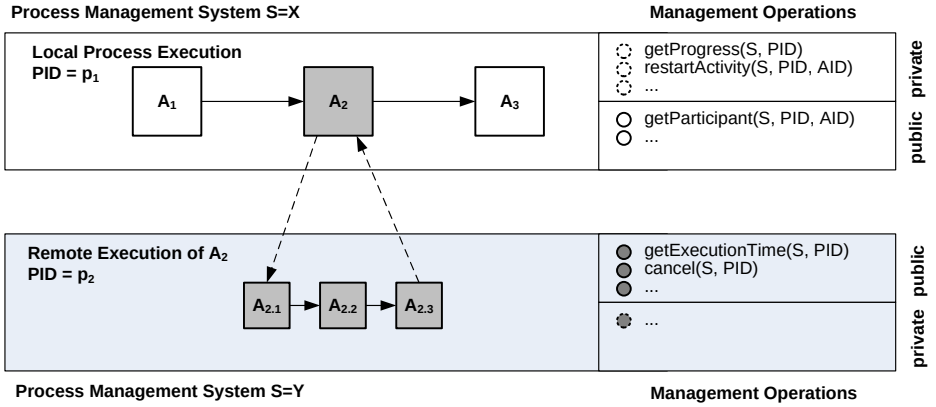


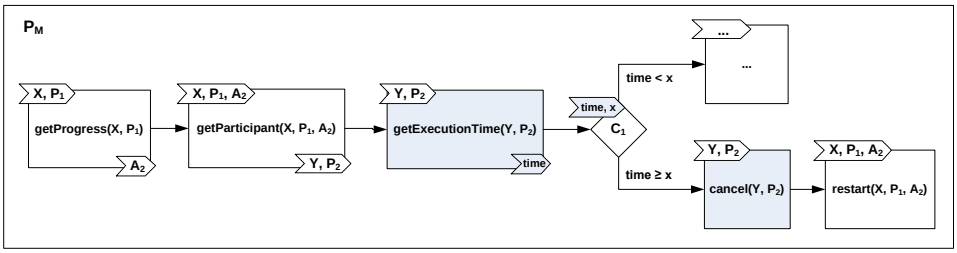
Abbildung 6.35: Fachlicher Prozess P_F mit technischem Management-Prozess P_M

Wie in Kapitel 3 erarbeitet wurde, ist eine automatische Anpassung möglich, wenn auf der Basis von (automatisch) erhobenen Informationen geeignete (benutzerdefinierte) Reaktionen abgeleitet und die hierfür notwendigen Handlungen automatisiert durchgeführt werden können. Es ist demnach nach einer geeigneten Möglichkeit zu suchen, die durch die Management-Schnittstelle bereitgestellten Informations- und Steuerungsfunktionalitäten zielgerichtet miteinander in Beziehung zu setzen. Hierbei ist es von Vorteil, wenn der automatische Anpassungsvorgang sowohl auf der Seite des Auftraggebers der entfernt ausgeführten Prozesspartition als auch durch die prozessausführende Partei selbst durchgeführt werden kann, da in letzterem Fall für die Durchführung eines Anpassungsvorgangs keine Netzwerkverbindung zwischen den beiden genannten Akteuren erforderlich ist.

Ein flexibler und mit wenig technischem Aufwand verbundener Lösungsansatz umfasst die Komposition von einzelnen Diensten der vorgestellten Management-Schnittstelle zur Abfrage und Manipulation von Eigenschaften der entfernten Prozessausführung zu einer weiteren prozessorientierten Anwendung, welche auf Basis der gleichen Beschreibungssprache auf derselben Ausführungsplattform ausgeführt werden kann wie die betrachtete entfernt ausgeführte fachliche Prozesspartition [Zap07]. Die komponierten Management-Dienste formen somit in Abgrenzung zum fachlichen Prozess P_F einen *Management-Prozess* P_M (vgl. auch [PvdH05]), wodurch die geforderte strikte Trennung zwischen fachlicher Logik und technischer Management-Logik erreicht werden kann (vgl. Abbildung 6.35). Gleichzeitig entsteht auf Basis der Flexibilität und Modularität dienstorientierter Architekturen eine lose Kopplung, welche es erlaubt, fachliche und technische Aufgaben bei Bedarf miteinander in Verbindung zu bringen und Management-Funktionalitäten auch organisationsübergreifend miteinander zu kombinieren. Im Gegensatz zur Anreicherung von fachlichen Prozessmodellen mit technischen Managementanweisungen wird der Umfang der fachlichen Prozessmodelle hierdurch nicht erhöht und die Ausführung der



(a) Verteilt ausgeführter Prozess und Management-Schnittstelle der beteiligten Ausführungssysteme



(b) Management-Prozess des Auftraggebers der entfernt ausgeführten Prozesspartition

Abbildung 6.36: Veranschaulichendes Beispiel eines aus Informations- und Steuerungsdiensten komponierten Management-Prozesses (Zap07)

hiervon abgeleiteten Prozessinstanzen somit prinzipiell nicht beeinträchtigt [Zap07].

Abbildung 6.36 zeigt ein anschauliches Beispiel für die Komposition eines Management-Prozesses zur Ableitung von Anpassungsbedarf während einer entfernt ausgeführten Prozesspartition. Ausgangspunkt ist die in Abbildung 6.36a dargestellte Situation, in der eine Ausführungseinheit X den fachlichen Prozess P_1 ausführt und dabei die Blockaktivität A_2 an die Ausführungseinheit Y auslagert. Durch die Ausführungseinheiten X und Y werden jeweils eine Reihe von Operationen zum Management lokal und entfernt ausgeführter Prozesse angeboten (zur Veranschaulichung sind jeweils einzelne, spezialisierte Dienste dargestellt). Die Operationen zum Management können öffentlich (*public*) oder nur intern (*private*) zugreifbar sein. Abbildung 6.36b zeigt einen möglichen, aus diesen Operationen aufgebauten Management-Prozess zur Behandlung von Verzögerungen der Prozessausführung. Tritt eine solche Verzögerung des betrachteten Prozesses P_1 auf, so wird der Management-Prozess instantiiert (z. B. über ein entsprechendes lokales Ereignis oder im Rahmen einer Fehlerbehandlungsmaßnahme). Im erste Schritt wird durch den Management-Prozess die aktuell ausgeführte Aktivität (A_2) identifiziert. Im zweiten Schritt wird die für A_2 verantwortliche Ausführungseinheit (Y) festgestellt und erhoben, unter welchem Identifikator (P_2) der Prozess auf dem Fremdsystem ausgeführt wird. Im dritten Schritt wird über eine Management-

Operation der Ausführungseinheit Y erfragt, wie lange die Ausführung von P_2 voraussichtlich noch dauern wird ($Time$). Überschreitet die verbliebene Ausführungszeit einen Wert x, so wird die entfernt ausgeführte Prozesspartition P_2 im vierten Schritt abgebrochen und im fünften Schritt die Aktivität A_2 auf dem lokalen System neu gestartet, um diese z. B. lokal auszuführen oder in einem erneuten Versuch an eine andere Ausführungseinheit auszulagern [Zap07].

Besteht der Management-Prozess nur aus Management-Diensten der prozessausführenden Partei, so kann der Management-Prozess bei Bedarf auch ohne eine dauerhafte Netzwerkverbindung zwischen den beiden Parteien ausgeführt werden. Da der fachliche Prozess nicht verändert werden muss, um den Management-Prozess anzupassen oder auszutauschen, können für verschiedene Prozessinstanzen unterschiedliche Management-Prozesse zum Einsatz kommen, was eine hohe Individualität gewährleistet. Zudem können Management-Prozesse leicht über bestehende Werkzeuge zum Prozessmanagement modelliert und angepasst werden. Die Entwicklung bzw. Anschaffung und Pflege weiterer Komponenten für das automatische Management entfernt ausgeführter Prozesspartitionen entfällt, da die bereits bestehenden Prozessmanagementsysteme auch zur Ausführung der Management-Prozesse zum Einsatz kommen können.

In der Arbeit von STRASSENBURG [Str09] wurde jedoch gezeigt, dass die für eine Flexibilisierung im Lebenszyklus einer prozessorientierten Anwendung erforderlichen Management-Prozesse bereits für weniger komplexe Management-Aufgaben sehr umfangreich werden und die hierfür erforderlichen Prozessmodelle den Umfang der fachlichen Prozessmodelle leicht übersteigen können. Ein wesentlicher Grund hierfür ist, dass die Auswertung von Regeln zur Ableitung von Anpassungsbedarf oft durch eine Vielzahl von komplexen Verzweigungen dargestellt werden muss (vgl. auch Abschnitt 3.6). Es kann daher nicht ausgeschlossen werden, dass die Ausführung der fachlichen Prozesse durch die zusätzliche Ausführung komplexer Management-Prozesse zur Überwachung und Steuerung entfernt ausgeführter Prozesspartitionen aufgrund einer höheren Auslastung der Ausführungseinheiten negativ beeinträchtigt wird. Zudem ist die Mächtigkeit der zu spezifizierenden Management-Ausdrücke durch die Möglichkeiten der jeweiligen Prozessbeschreibungssprache begrenzt. Insbesondere wird durch einen rein prozessorientierten Ansatz für die Spezifikation von Anpassungsmaßnahmen die für eine zeitnahe Anpassung relevante Auswertung von Regeln und die Verarbeitung von Ereignissen nicht bzw. nicht optimal unterstützt. Im nächsten Abschnitt wird daher ein alternativer Ansatz dargestellt, welcher die Ableitung von Anpassungsmaßnahmen auf Basis von regelbasierten Systemen und der Verarbeitung komplexer Ereignisse ermöglicht.

6.3.5 Regel- und ereignisbasiertes Management

Das zuvor in Abschnitt 6.3.4 genannte automatische Management der Prozessausführung durch die Spezifikation und Ausführung von ergänzenden Management-Prozessen überzeugt vor allem durch seine einfache Anwendbarkeit ohne zusätzliche technische Infrastrukturen. Basierend auf der in Abschnitt 6.3.3 vorgestellten dienstbasierten Management-Schnittstelle für Prozessmanagementsysteme können jedoch auch auf anderen Paradigmen beruhende Middlewaresysteme zur benutzerdefinierten Beobachtung und Steuerung entfernt ausgeführter Prozesspartitionen aufbauen, welche besser für eine zeitnahe Ableitung von Anpassungsbedarf geeignet sind als ein prozessorientierter Ansatz. In diesem Abschnitt wird daher ein alternativer Ansatz zum regel- und ereignisbasierten Management der Prozessausführung vorgestellt. Dazu wird die vorgeschlagene Management-Schnittstelle durch eine weitere *Management-Komponente* zu einer 2-Schichten-Architektur ergänzt.

Abbildung 6.37 zeigt eine grobe Übersicht zur Anbindung einer losegekoppelten Management-Komponente (Schicht 2), welche die Funktionalitäten der zugrunde liegenden dienstbasierten Management-Schnittstelle (Schicht 1) konsumiert, deren Ereignisse und Informationen auswertet sowie bei Bedarf (Re-)Aktionen daraus ableitet und diese entsprechend initiiert. Als Ausgangspunkt erstellt der Auftraggeber der entfernt auszuführenden Prozesspartition hierzu auf Basis des zu verteilenden Prozessmodells ein zusätzliches Dokument (*Management-Dokument*), welches die Anforderungen des Auftraggebers in Hinblick auf das Management des fachlichen Prozesses enthält (*Management-Regeln*).

Die als Management-Regeln spezifizierten Anforderungen umfassen dabei alle Objekte, Situationen und Aktionen, welche aus der Perspektive des Auftraggebers der entfernt ausgeführten Prozesspartition für eine den definierten Rahmenbedingungen entsprechende Ausführung und Verwaltung des Prozesses relevant sind und nicht bereits durch die fachliche Prozessbeschreibung abgedeckt sind [ZBH+10]. Die relevanten Objekte des Managements sind durch die Entitäten des Referenzmodells (d. h. Prozessmodelle, Prozessinstanzen und deren Datenobjekte) determiniert (vgl. Abschnitt 6.3.2). Situationen und Aktionen werden im Management-Dokument als Beziehung von (komplexen) Ereignismustern und Reaktionen beschrieben, welche bei Eintreten eines Ereignisses ausgelöst werden sollen. Ein Beispiel für eine solche Beziehung zwischen Objekt, Situation und Aktion ist die Überwachung der Dauer einer vorgegebenen Aktivität (*Objekt*), das Feststellen eines für längere Zeit fehlenden Fortschritts (*Situation*) und das Neustarten der Aktivität zur Bewältigung der als Fehler erkannten Situation (*Aktion*). Dabei müssen die Aktionen nicht notwendigerweise einen ändernden Charakter besitzen, sondern können z. B. auch zum Sammeln weiterer Informationen eingesetzt werden. Ein Beispiel ist der Vorgang, nach jeder erfolgreichen Beendigung einer Aktivität alle Informationen über den jeweils ausführenden Teilnehmer, die Ausführungsdauer und die geographische Position aufzuzeichnen. Schließlich ist es durch die Spezifika-

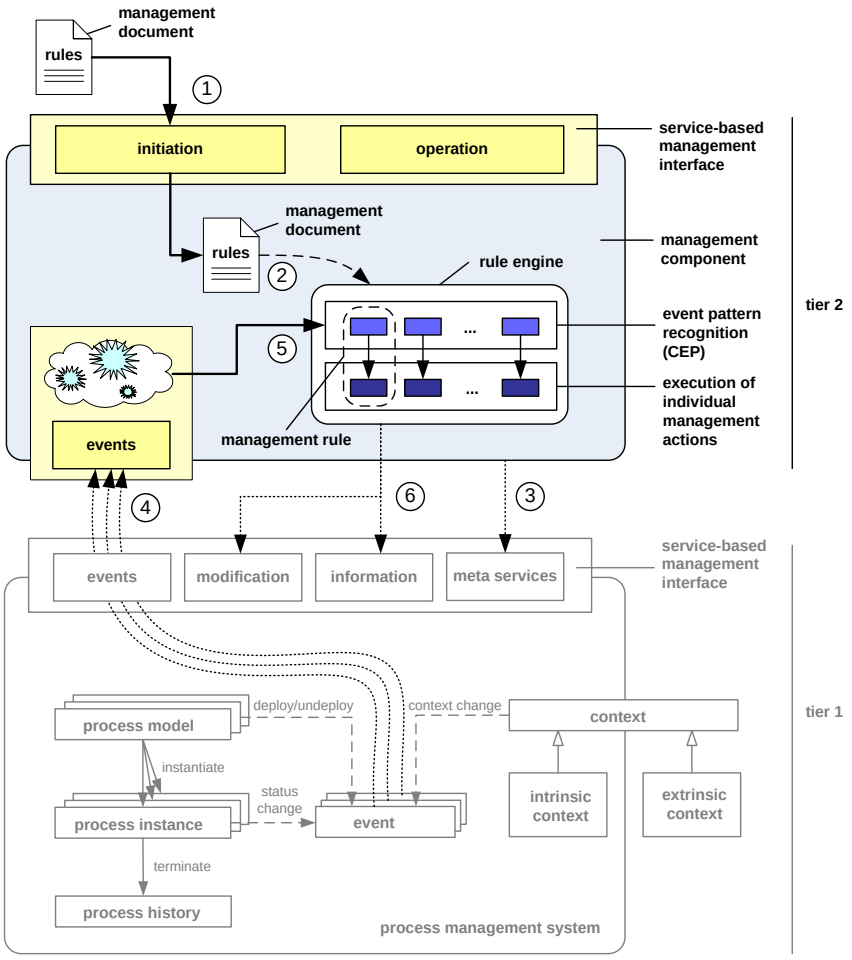


Abbildung 6.37: 2-Schichten-Architektur zum regel- und ereignisbasierten Management entfernt ausgeführter Prozesspartitionen (ZBH+10)

tion von Management-Regeln auch möglich, die Verteilung des Prozesses zu beeinflussen. Überschreitet zum Beispiel die Auslastung der Prozess-Engine einen vorgegebenen Schwellwert, so kann der Prozess automatisch angehalten und an eine Ausführungseinheit mit geringerer Auslastung übertragen werden werden [ZBH+10].

Für die Durchführung eines automatisierten Anpassungsvorgangs werden die durch den Auftraggeber der entfernt ausgeführten Prozesspartition spezifizierten Management-Regeln an eine Management-Komponente übergeben, welche für die Interpretation und Auswertung der Regeln sowie für die Durchführung etwaiger Reaktionen verantwortlich sein soll (Schritt 1 in Abbildung 6.37). Dazu werden in Ergänzung zu den Funktionalitäten der zugrunde liegenden Management-Schnittstelle durch die übergeordnete Management-Komponente zwei höherwertige dienstbasierte Schnittstellen angeboten, welche der Automatisierung der im Management-Dokument vorgegebenen Anforderungen dienen [Str09, ZBH+10]:

- **Initiierung** (*Initiation*): Über die Initiierungsschnittstelle können Management-Vorgänge gestartet werden, indem durch den Aufrufer ein neues Management-Dokument übergeben wird. Die dort enthaltenen benutzerdefinierten Management-Regeln werden an ein (internes) regelverarbeitendes System (*Rule Engine*) weitergereicht (Schritt 2) und die für den Management-Vorgang benötigten Ereignisse über die zugrunde liegende Management-Schnittstelle abonniert (Schritt 3). Als Antwort auf die (erfolgreiche) Initiierung eines Management-Vorgangs erhält der Auftraggeber eine Referenz zu dessen eindeutiger Identifikation. Werden relevante Ereignisse empfangen (Schritt 4), so werden diese zur Erkennung der in den Management-Regeln spezifizierten Ereignismuster an das regelverarbeitende System weitergeleitet (Schritt 5). Ist eine Regel erfüllt, so wird die dazu spezifizierte Reaktion ausgelöst (Schritt 6).

- **Vorgang** (*Operation*): Die Vorgangsschnittstelle dient zur Anpassung bereits initiierter Management-Vorgänge. Über die bei der Initiierung erhaltene Referenz können dabei durch den Aufrufer neue Management-Regeln hinzugefügt, nicht länger benötigte Management-Regeln entfernt, oder ein laufender Management-Vorgang beendet werden.

Die Management-Komponente stellt somit eine generische Client-Komponente dar, welche die manuelle Verwendung der zugrunde liegenden Management-Schnittstelle durch den Auftraggeber der entfernt ausgeführten Prozesspartition ersetzen bzw. sinnvoll ergänzen kann. Die aufwändige Entwicklung von individuellen Client-Anwendungen für die (automatische) Überwachung und Steuerung von Prozessen kann somit entfallen und stattdessen über die Spezifikation von plattformunabhängigen Management-Dokumenten erfolgen, welche auf dem gemeinsamen Referenzmodell für das Prozessmanagement als verwaltbare Ressource basieren. Wird zudem eine gemeinsame Struktur für den Aufbau und den Inhalt von Management-Dokumenten und den beinhalteten Management-Regeln verwendet, so kann das automatische Management sowohl auf Seiten des Auftraggebers der entfernt ausgeführten Prozesspartition als auch auf Seiten der prozessausführenden Partei erfolgen. Es kann somit je nach Bedarf eine Bandbreite von einer relativ starken (exklusiven) Kontrolle durch den Auftraggeber bis hin zu einer (vollständigen) Delegation der Überwachungs- und Steuerungsmaßnahmen an die prozessausführende Partei erfolgen (vgl. Abschnitt 6.3.1). Im Folgenden wird daher ein Vorschlag für einen allgemeinen Aufbau von Management-Dokumenten zur automatischen Überwachung und Steuerung von Prozessen vorgestellt, welcher den plattformübergreifenden Austausch von Anforderungen zum Management entfernt ausgeführter Prozesse unterstützt.

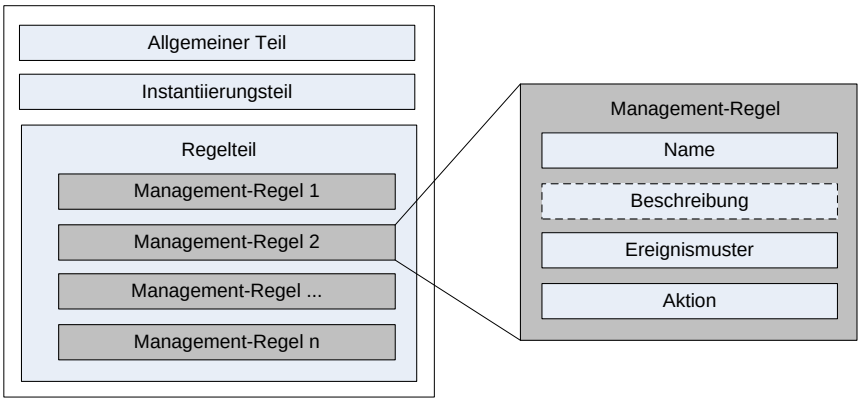


Abbildung 6.38: Allgemeine Struktur eines Management-Dokuments mit Management-Regeln (Str09, ZBH+10)

Inhalt und Struktur von Management-Dokumenten

Ein Management-Dokument enthält alle Informationen und benutzerdefinierten Rahmenbedingungen für die automatische Überwachung und Steuerung von prozessorientierten Anwendungen, welche auf einem zu dem in Abschnitt 6.3.2 vorgeschlagenen Referenzmodell konformen Prozessmanagementsystem ausgeführt werden. Abbildung 6.38 zeigt einen Überblick über den Inhalt und die mögliche Struktur eines Management-Dokuments, welches in seinem übergeordneten Aufbau an die Darstellung von BARESI und GUINEA [BG05] angelehnt ist (vgl. auch Abschnitt 5.7.3). Im Gegensatz zu dem von BARESI und GUINEA vorgestellten Konzept wird das Management-Dokument jedoch in dem hier vorgeschlagenen Ansatz nicht während des Deployments eines Prozessmodells in dessen Prozessbeschreibung eingearbeitet, sondern zur Laufzeit in Bezug auf die referenzierten Prozessmodelle und -instanzen ausgewertet. Im Folgenden wird die hierfür vorgeschlagene Struktur genauer erläutert.

Ein Management-Dokument enthält als wichtigsten Bestandteil die durch den Auftraggeber spezifizierten Regeln (*Management-Regeln*) zur Überwachung und Steuerung einer auf dem betrachteten System ausgeführten Prozesspartition. Dabei wird sowohl das Management von Prozessmodellen unterstützt, die zwecks mehrfacher Instantiierung fest auf der betrachteten Ausführungseinheit installiert wurden, als auch das Management von individuell (verteilt) ausgeführten Prozessinstanzen. Das Management-Dokument beinhaltet daher neben den *Management-Regeln* und einem *allgemeinen Teil* zur internen Verarbeitung auch einen *Instantiierungsteil*, welcher das Management der technisch relevanten Lebenszyklusphasen eines Prozesses unterstützt.

Listing 6.1 zeigt den Aufbau des in Abbildung 6.38 dargestellten *Regelteils* mit der übergeordneten Datenstruktur *Management-Regeln* (*Management-Rules*). Zur Verwaltung und zwecks späterer Anpassung besitzt jede einzelne *Management-Regel* (*Management-Rule*) einen innerhalb des Management-Dokuments eindeutigen Namen (*Name*) und optional eine Beschreibung (*De-*

```
1 MANAGEMENT-RULES
2   MANAGEMENT-RULE
3     NAME          : <String>
4     [DESCRIPTION  : <String>]
5     RULE-PATTERN  : <Event-Pattern>
6     RULE-ACTION   : <Service-Invocation>
7   END MANAGEMENT-RULE
8   ...
9 END MANAGEMENT-RULES
```

Listing 6.1: Struktur der Management-Regeln (ZBH+10)

scription). Das Element *Rule-Pattern* ist durch die Angabe eines (komplexen) Ereignismusters determiniert, welches zur Laufzeit durch ein ereignisverarbeitendes System auf das Eintreten der spezifizierten Situation überwacht werden kann. Der Aktionsteil (*Rule-Action*) definiert, welche Aktion zur Behandlung der festgestellten Situation ausgeführt werden soll. Die auszuführende Aktion wird dabei durch einen Dienstaufruf angegeben und kann somit direkt auf die Informations- oder Manipulationsschnittstelle von Schicht 1 der Management-Architektur verweisen oder einen beliebigen (lokalen oder entfernten) funktionalen Dienst integrieren. Dabei können über eine Komposition von Diensten prinzipiell auch komplexe Funktionalitäten realisiert werden [ZBH+10].

Listing 6.2 zeigt, wie die Management-Regeln entsprechend Abbildung 6.38 in das umgebene Management-Dokument eingebettet sind (Zeile 18). Darüber hinaus enthält das Management-Dokument zusätzliche Informationen, welche für die Korrelation, Zuweisung und Ausführung der Management-Regeln benötigt werden. Das Element *General-Information* (Zeilen 3-7) gibt dazu zunächst das Ausführungssystem an, welches überwacht bzw. gesteuert werden soll (*Management-Endpoint*). Dies ermöglicht es, die Management-Komponente sowohl für lokal als auch für entfernt ausgeführte Prozesse zu nutzen. Der *Management-Endpoint* stellt gleichzeitig den Abonnenten für die in den Management-Regeln spezifizierten Ereignisse dar. Das Management wird automatisch gestartet, sobald ein Management-Dokument über die Initiierungsschnittstelle an die Management-Komponente übergeben wird und das regelverarbeitende System damit beginnt, den Eingang abonnierter Ereignisse zu verarbeiten. Das Element *Management-Mode* gibt an, unter welchen Bedingungen das Management beendet werden soll. Die Beendigung des Managements spielt eine wichtige Rolle, da hierbei die auf dem regelverarbeitenden System installierten Regeln entfernt und das Abonnement der dazugehörigen Ereignisse abbestellt werden muss. Das Management wird daher entweder automatisch beendet, wenn die beobachtete Prozessinstanz terminiert ist (*Management-Mode=System*) oder es wird explizit durch den Initiator des Management-Dokuments über die Vorgangsschnittstelle beendet (*Management-Mode=User*). Letzteres ist vorteilhaft, wenn ein

```
1  MANAGEMENT-DOCUMENT
2
3    GENERAL-INFORMATION
4        MANAGEMENT-ENDPOINT       : <URL>
5        MANAGEMENT-MODE           : "system"|"user"
6        [NOTIFICATION-ENDPOINT    : <URL>]
7    END GENERAL-INFORMATION
8
9    INSTANTIATION-INFORMATION
10       PROCESS-MODEL-REFERENCE   : <STRING>
11       LOCAL-INSTANCE-REFERENCE  : <STRING>
12       [INSTANTIATION-TIME       : <DATE>]
13       [INSTANTIATION-DELAY      : <INTEGER>]
14       [INSTANTIATION-PARAMETERS]
15       [BLOCKING-EVENT-TYPES]
16   END INSTANTIATION-INFORMATION
17
18   MANAGEMENT-RULES
19
20 END MANAGEMENT-DOCUMENT
```

Listing 6.2: Struktur des Management-Dokuments (ZBH+10)

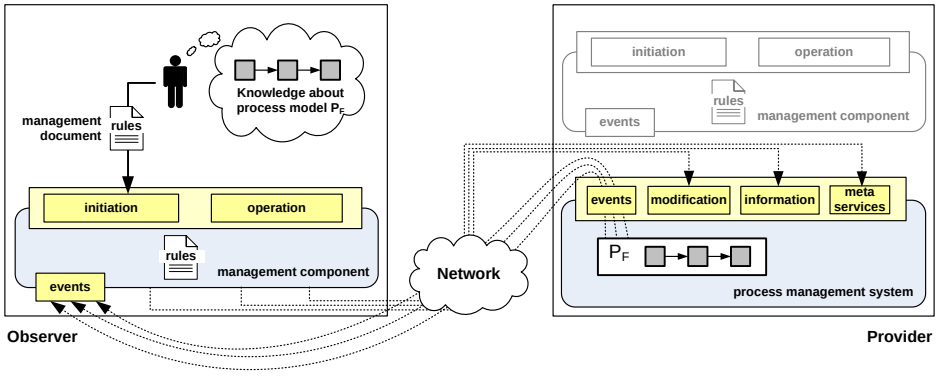
Ausführungssystem über die Ausführung einer einzelnen Prozessinstanz hinaus beobachtet werden soll (zum Beispiel als Kandidat für die Ausführung zukünftig zu verteilender Prozesspartitionen), wenn eine spätere Auswertung der Prozesshistorie erfolgen soll oder falls noch weitere Prozessinstanzen vom selben Prozessmodell instantiiert und auf dieselbe Weise beobachtet bzw. gesteuert werden sollen [ZBH+10]. Das optionale Element *Notification Endpoint* gibt an, dass im Fall einer automatischen Beendigung des Managements eine Nachricht an das angegebene System gesendet werden soll [Str09].

Der Instantiierungsteil des Management-Dokuments (*Instantiation-Information*, Zeilen 9-16) dient zur Verwaltung der zu beobachtenden bzw. zu steuernden Prozessinstanzen. Werden Prozessmodell und Prozessinstanz zusammen an das Ausführungssystem übergeben (wie etwa in dem in Abschnitt 6.2 vorgeschlagenen Verteilungsmodell), so kann durch die Elemente *Process-Model-Reference* und *Process-Instance-Reference* ein Zusammenhang zwischen Prozessmodell und Prozessinstanz hergestellt werden. Wird nur ein Prozessmodell zur späteren Instantiierung an die Ausführungsumgebung übergeben, so stellt das Element *Process-Instance-Reference* einen Platzhalter für die Prozessinstanz dar, welche zum Zeitpunkt des Deployments des Prozessmodells noch nicht existiert, jedoch bereits in den Management-Regeln des Management-Dokuments referenziert werden muss. Soll die Instantiierung – optional zu einem bestimmten Zeitpunkt (*Instantiation Time*) oder nach einer bestimmten Zeitdauer (*Instantiation Delay*) – durch die Management-Komponente erfolgen, so können optional Parameter für die

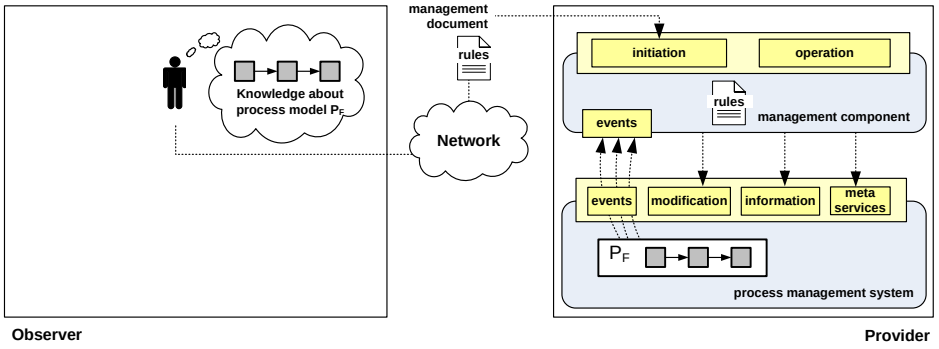
Instantiierung festgelegt werden (*Instantiation Parameters*). Schließlich kann im Management-Dokument angegeben werden, welche Ereignisse des betreffenden Prozesses als *blockierendes Ereignis* (*Blocking-Event-Types*) abonniert werden sollen (vgl. Abschnitt 6.3.3). Blockierende Ereignisse verhindern, dass die Prozessausführung voranschreitet, bevor eine Reaktion auf das Ereignis abgeleitet und initiiert wurde. Da blockierende Ereignisse jeweils nur für die Überwachung einzelner laufender Prozessinstanzen vorgesehen sind, ist ihre Spezifikation ebenfalls Teil der *Instantiation-Information* [Str09, ZBH+10].

Online- und Offline-Management entfernt ausgeführter Prozesspartitionen

Abhängig davon, ob die Verarbeitung von (komplexen) Ereignissen und die Ableitung geeigneter Reaktionen auf einer lokalen oder entfernten Management-Komponente beauftragt werden, kann die Durchführung von individuell festgelegten Überwachungs- und Steuerungsmaßnahmen sowohl im direkten Einflussbereich des Auftraggebers einer entfernt ausgeführten Prozesspartition als auch durch die prozessausführende Partei selbst stattfinden (vgl. Abbildung 6.39). Für ein lokales Management von Seiten des Auftraggebers werden alle relevanten Informationen und Ereignisse über ein Netzwerk zur Management-Komponente des Auftraggebers übertragen, wo sie ausgewertet werden und bei Bedarf als Reaktionen entsprechende Steuerungsbefehle über das Netzwerk zurück an die prozessausführende Partei gesendet werden. Aufgrund der Notwendigkeit einer (dauerhaften) Netzwerkverbindung wird diese Variante des Managements als *Online-Management* bezeichnet (vgl. Abbildung 6.39a). Es erlaubt die uneingeschränkte Kontrolle über die Bildung von Zusammenhängen zwischen Ereignissen bzw. Informationen und der Ableitung von Anpassungsmaßnahmen und somit eine höhere Kontrolle über die Einhaltung und Anpassung der Management-Regeln. Dies ist zum Beispiel notwendig, falls Management-Entscheidungen vertraulich sind, bestimmte (kritische) Entscheidungen erst durch einen menschlichen Administrator freigegeben bzw. bestätigt werden müssen oder falls die Management-Regeln besonders häufigen Änderungen unterliegen. Der Preis für diese höhere Kontrolle über das Management ist jedoch ein erhöhtes Nachrichtenaufkommen, welches das Netzwerk zwischen Auftraggeber und prozessausführender Partei belastet und – im Fall der Verwendung von blockierenden Ereignissen – die Prozessausführung um die Dauer der aus dem Management resultierenden Nachrichtenlaufzeiten verzögert. Zudem muss der Auftraggeber über eine lokale Management-Komponente verfügen (d. h. diese entwickeln bzw. erwerben, warten und betreiben), um Entscheidungen über den Anpassungsbedarf seiner Prozesse treffen zu können [ZBH+10]. Der dadurch entstehende Mehraufwand ist unter Umständen nicht immer vertretbar, zum Beispiel in Fällen, in denen der Auftraggeber der entfernt ausgeführten Prozesspartition selbst gar nicht über Hard-/Software für Maßnahmen zum Prozessmanagement verfügt [ZBH+10] (vgl. z. B. die *Process-Management-as-a-Service*-Szenarien in Abschnitt 4.2.6).



(a) Online-Management des fachlichen Prozesses P_F



(b) Offline-Management des fachlichen Prozesses P_F

Abbildung 6.39: Online- und Offline-Management von entfernt ausgeführten Prozesspartitionen

Für ein von Seiten des Auftraggebers ferngesteuertes Management wird das Management-Dokument an die Management-Komponente auf Seiten der prozessausführenden Partei gesendet und der dazugehörige Management-Vorgang auf dem entfernten System initiiert. Alle relevanten Ereignisse werden in diesem Fall lokal verarbeitet, die spezifizierten Reaktionen durch die prozessausführende Partei abgeleitet und direkt über die lokale Manipulationsschnittstelle umgesetzt. Da hierfür mit Ausnahme der initialen Übertragung des Management-Dokuments für die Durchführung des Managements keine dauerhafte Netzwerkverbindung zwischen dem Auftraggeber und der prozessausführenden Partei erforderlich ist, wird diese Variante als *Offline-Management* bezeichnet (vgl. Abbildung 6.39b). Hierbei können auch blockierende Ereignisse lokal verarbeitet werden, so dass die Ausführung des fachlichen Prozesses nicht durch eine (ggf. fehleranfällige oder langsame) Netzwerkkommunikation beeinflusst werden kann. Da zur Ableitung von Anpassungsbedarf keine Kommunikation mit dem Auftraggeber der entfernt ausgeführten Prozesspartition benötigt wird, ist diese Variante zudem für die Integration mobiler Ausführungseinheiten geeignet, welche unter Umständen nicht über eine dauerhafte Netzwerkanbindung oder nur über Netzwerke geringer Bandbreite mit hohen Datenübertragungskosten verfügen [ZBH+10].

Die vorliegende Architektur erlaubt es zudem, beide der vorgestellten Strategien flexibel miteinander zu kombinieren. So ist es zum Beispiel denkbar, die entfernt ausgeführte Prozesspartition nach fest vorgegebenen Regeln durch die prozessausführende Partei überwachen zu lassen und nach Abschluss der Prozessausführung die zur Auswertung relevanten Daten an den Auftraggeber zu übertragen. Eine weitere Möglichkeit besteht darin, das automatische Management vorhersehbarer Situationen auf Basis der Management-Regeln durch die prozessausführende Partei durchführen zu lassen und nur bei besonders kritischen bzw. unvorhergesehenen Problemen den Auftraggeber zu integrieren [ZBH+10]. Dabei ist hierfür von Seiten des Auftraggebers nicht unbedingt das Vorhalten einer vollständigen Management-Komponente notwendig, da über die dienstorientierte Architektur auch beliebige externe Dienste und Ereignisse in die Management-Regeln integriert werden können. Somit ist zum Beispiel auch die Information des Auftraggebers durch das Senden einer E-Mail möglich.

Die Nutzung einer Management-Komponente auf der Seite der prozessausführenden Partei erfordert jedoch über die in diesem Abschnitt vorgestellte Struktur der Management-Dokumente hinaus die Einigung auf eine geeignete Sprache zur Beschreibung der dort enthaltenen Ereignismuster und Reaktionen. Ebenso wie die Auswahl der Prozessbeschreibungssprache ist die Verwendung einer Sprache zur Beschreibung der konkreten Management-Regeln anwendungs- bzw. plattformabhängig und ist daher nicht Teil dieser Arbeit. Es wird jedoch im Rahmen einer beispielhaften Implementierung in Kapitel 7.4 ein Ansatz zur Integration von *Complex Event Processing* (*CEP*) auf Basis von *Event-Processing-Language*-Ausdrücken und dem Aufruf von Web Services als Reaktionen auf die hierdurch erkannten Situationen vorgestellt. Potentiellen Verwendern des hier vorgestellten Rahmenmodells zum Management entfernt ausgeführter Prozesspartitionen steht jedoch durch die Abstraktion des Management-Dokuments von konkreten Technologien die Integration anderer bestehender oder eigener Sprachen frei.

6.3.6 Zusammenfasssung und Einordnung

Neben den in Abschnitt 6.2 vorgestellten Möglichkeiten zur dynamischen Verteilung von individuellen Prozessinstanzen wurde in diesem Abschnitt mit der Erhebung und Bereitstellung von Informationen entfernt ausgeführter Prozesspartitionen ein weiterer wichtiger Aspekt zur Flexibilisierung verteilter Prozessausführung adressiert. Im Gegensatz zu vielen der genannten bestehenden Ansätze (vgl. Abschnitt 5.7) wurde hierbei das Ziel verfolgt, Flexibilität durch die Unabhängigkeit von Anweisungen zur Überwachung und Steuerung des Prozesses in technischer Hinsicht und den Anweisungen innerhalb der Prozessbeschreibung in fachlicher Hinsicht zu gewinnen. Der Vorteil hierbei ist, dass Konstrukte zur Überwachung und Steuerung für verschiedene Prozesse bzw. Prozessmodelle wiederverwendet, aber auch für einzelne Prozessinstanzen oder sogar für verschiedene Ausführungseinheiten während der

Ausführung einer Prozessinstanz individualisiert und auch zur Laufzeit angepasst werden können. Die Maßnahmen zur Überwachung und Steuerung sind somit – ebenso wie die dynamische Verteilung der Prozessausführung – als fortwährend anpassungsfähig anzusehen und erfüllen somit den im Rahmen dieser Arbeit geforderten Eigenschaften von Flexibilität (vgl. Abschnitt 3.1).

Konkreter Beitrag des hier vorgeschlagenen Ansatz ist die Umsetzung des Leitbildes der *Paketverfolgung* für beliebige entfernt ausgeführte Prozesspartitionen mit der Möglichkeit zur Ableitung automatischer und situationsbedingter Steuerungsmaßnahmen. Nach dem Vorbild der Paketverfolgung wird ein möglichst großer Informations- und Anpassungsspielraum erreicht, indem einerseits ein Modell zur anbieterseitigen Bereitstellung von (ansonsten intern gekapselten) Informations- und Steuerungsfunktionalitäten gewählt wird und indem andererseits mit der Kombination von Pull- und Push-Modellen verschiedene Arten des Informationsbezugs unterstützt werden. Dazu wurde ein Referenzmodell erarbeitet, welches die Repräsentation eines prozessausführenden Systems als *verwaltbare Ressource* und den individuell verhandelbaren Austausch von Eigenschaften dieser Ressource über eine generische Schnittstelle erlaubt. Durch das Anbieten der Schnittstelle als Dienst des Prozessmanagementsystems zum Management der dort ausgeführten Prozesse kann das unter Abschnitt 6.2 verwendete Konzept des *Process-Management-as-a-Service* (*PMaaS*) auch als Grundlage für ein dynamisches Management verteilt ausgeführter Prozesse adaptiert werden. Als wichtige Einschränkung bleibt jedoch festzuhalten, dass nur die vom jeweiligen Anbieter bereitgestellten Eigenschaften zugegriffen werden können und von Seiten des Dienstnutzers auf die angegebenen Werte hinsichtlich Korrektheit und Aktualität sowie auf die tatsächliche Durchführung der geforderten Anweisungen zur Steuerung der Prozessausführung vertraut werden muss. Durch diese Einschränkung wird der geforderten Autonomie der prozessausführenden Partei Rechnung getragen. Da bereits für die korrekte Ausführung des fachlichen Prozesses eine ausreichende Vertrauensbeziehung zwischen Auftraggeber und prozessausführender Partei vorausgesetzt wird und Vertrauensaspekte bei der Auswahl der prozessausführenden Parteien einfließen können, kann jedoch eine entsprechende Vertrauensbeziehung auch für das Management der entfernt ausgeführten Prozesspartition angenommen werden.

Basierend auf der vorgestellten dienstbasierten Management-Schnittstelle wurden daher zwei weitergehende Möglichkeiten zur Automatisierung des Anpassungsvorgangs vorgestellt, welche auch eine potentiell fehlende Netzwerkverbindung zwischen dem Auftraggeber und der prozessausführenden Partei berücksichtigen. Durch die Spezifikation von Management-Prozessen können dabei bereits bestehende Infrastrukturen verwendet werden, um auf Basis der bereitgestellten Informationen die Ausführung der betrachteten Prozesse zu beeinflussen. Ein weiterführender Beitrag dieser Arbeit umfasst die Definition einer ergänzenden Management-Komponente als Rahmenwerk für benutzerdefinierte Management-Vorgänge auf Basis von regelbasierten Systemen und der Verarbeitung komplexer Ereignismuster. In beiden Fällen kann durch die

Betrachtung des Prozessmanagementsystem als verwaltbare Ressource eine direkte Durchsetzung von Anpassungsmaßnahmen auf die laufende Prozessinstanz vorgenommen werden.

Eine Integration der hier vorgestellten Architektur zur Überwachung und Steuerung verteilt ausgeführter Prozesse mit dem Verfahren zur dynamischen Verteilung von Prozessinstanzen durch Migration stellt somit Möglichkeiten zur automatischen Erhebung von Informationen zur Verfügung, welche eine dynamische Verteilung von Prozessinstanzen zur Laufzeit beeinflussen können, und erlaubt zudem die gezielte Steuerung der verteilten Prozessausführung. Ein ganzheitlicher Ansatz dazu wird im Rahmen einer prototypisch implementierten Middleware-Infrastruktur exemplarisch in Kapitel 7 vorgestellt.

6.4 Konzept abstrakter Benutzungsschnittstellen

Für automatisiert ausführbare Aktivitäten wird im Rahmen von verteilt ausgeführten Prozessen bereits durch dienstorientierte Architekturen und die entsprechende Möglichkeit, Software-Funktionalitäten als Dienst lose gekoppelt in eine prozessorientierte Anwendung einzubinden, eine hohe Flexibilität erreicht. Wie in Abschnitt 6.2.3 dargestellt wurde, ist es möglich, bei einer verteilten Ausführung je nach gewähltem Abstraktionsgrad einen Dienst entfernt aufzurufen oder (falls verfügbar) durch eine lokal verfügbare Ressource zu ersetzen, welche eine äquivalente Funktionalität erbringt. Die Lokalität der Diensterbringung ist dabei in der Regel für Prozessteilnehmer bzw. Dienstnutzer transparent, da automatisiert ablaufende Aktivitäten normalerweise keine Benutzungsschnittstellen besitzen bzw. benötigen. Es ist daher weitestgehend unerheblich, von welcher Art von Endgerät und in welcher Situation ein Dienst aufgerufen wird, sofern die vorgegebene Funktionalität erfüllt wird. Wie in den Abschnitten 2.1, 4.2 und 5.8 gezeigt wurde, weisen prozessorientierte Anwendungen jedoch auch für die Integration von menschlichen Prozessteilnehmern und die (teil-)automatisierte Unterstützung ihrer Arbeitsvorgänge eine sehr große Bedeutung auf. So wurden zum Beispiel selbst rein dienstorientierte Prozessbeschreibungssprachen wie WS-BPEL um Konzepte zur Interaktion mit dem Anwender ergänzt (vgl. Abschnitte 5.2.4 und 5.8.2). Bei der Betrachtung der Flexibilität von prozessorientierten Anwendungen im Allgemeinen wurde zudem die Anpassbarkeit der Benutzungsschnittstelle an den aktuellen Kontext von menschlichen Prozessteilnehmern besonders hervorgehoben (vgl. Abschnitt 3.5.5). Dies hat besondere Relevanz, wenn der Benutzer mit einer prozessorientierten Anwendung interagiert, welche durch eine mobile Ausführungseinheit ausgeführt wird. Zum einen kann es hierbei zu einer häufigen Änderung des Einsatzortes und somit auch zu einer Änderung des Ausführungskontextes kommen. Zum anderen wird eine mobile Ausführung häufig überhaupt erst durch eine vielfältige Interaktion mit dem Benutzer motiviert, welche erfordert, dass der Benutzer den Prozess mit sich führt –

während für rein automatisch ausgeführte Prozesse oft keine Notwendigkeit zur Mobilität besteht [ZK07, ZBV09, ZVBK09].

Für eine Integration von Interaktionen mit Prozessteilnehmern und die Beschreibung von entsprechenden Benutzungsschnittstellen besteht jedoch weiterhin Flexibilisierungspotential (vgl. Abschnitt 5.8). Bislang werden entweder monolithische lokale Anwendungen integriert, welche über eine statische Benutzeroberfläche verfügen, webbasierte Anwendungen über das Netzwerk eingebunden, welche eine Benutzeroberfläche über den Browser darstellen können, oder konkrete (statische) Formulare für einzelne Benutzeroberflächen der als interaktiv gekennzeichneten Aktivitäten mit dem Prozess installiert und bei Bedarf ausgeführt [ZBV09, ZVBK09]. Wird ein Prozess dynamisch verteilt ausgeführt, so kann unter Umständen jedoch nicht vorhergesehen werden, welche konkreten Anforderungen am Ort der Prozessausführung vorherrschen werden. Eine flexible Anpassung von Benutzungsschnittstellen an den aktuellen Kontext des menschlichen Benutzers in Hinblick auf seine aktuelle Situation und die Art seines (stationären oder mobilen) Endgeräts ist somit bislang nicht möglich oder durch die Notwendigkeit zu einer dauerhaften Netzwerkverbindung eingeschränkt.

Um die fehlende Flexibilität für die dynamische Anpassung von Benutzungsschnittstellen zur Laufzeit von verteilt ausgeführten Prozessen zu adressieren, wird in diesem Abschnitt ein Konzept für eine abstrakte Beschreibung von Benutzeroberflächen vorgeschlagen. Die abstrakten Beschreibungen können dabei durch einen generischen (lokalen oder entfernten) Dienst verarbeitet werden, so dass ihnen hinsichtlich der Verteilung der Prozessausführung die gleiche Flexibilität wie den oben genannten automatischen (dienstbasierten) Aktivitäten zugrunde liegt. Das diesem Ansatz zugrunde liegende Prinzip abstrakter Benutzungsschnittstellen basiert auf dem im Rahmen von Abschnitt 5.8.5 vorgestellten CTT-Modell und wird in Abschnitt 6.4.1 vorgestellt. Das angepasste Meta-Modell für die Beschreibung abstrakter Benutzungsschnittstellen für verteilt ausgeführte Prozesse ist Inhalt von Abschnitt 6.4.2. Abschnitt 6.4.3 zeigt zwei verschiedene Varianten zur Integration der abstrakten Benutzungsschnittstellen in prozessorientierte Anwendungen auf. Die generelle Vorgehensweise zur dynamischen, kontextbasierten Verarbeitung der Prozesse mit ihren abstrakten Interaktionsbeschreibungen ist Inhalt von Abschnitt 6.4.4. Abschnitt 6.4.5 schließt mit einer Zusammenfassung und Einordnung dieses Beitrags in die vorliegende Arbeit.

6.4.1 Prinzipien und Entwicklungsgrundsätze

In den Abschnitten 3.5.5 und 5.8 wurden verschiedene Varianten für eine kontextbasierte Darstellung von Benutzungsschnittstellen für prozessorientierte Anwendungen im Allgemeinen und für die Anpassung von Benutzungsschnittstellen an multiple Endgeräte im Speziellen untersucht. Auf Basis der dort gewonnenen Erkenntnisse wird in diesem Abschnitt eine Vorgehensweise motiviert, welche es erlaubt, prozessorientierte Anwendungen flexibel auf verschie-

denartige Ausführungseinheiten zu verteilen, ohne dabei jeweils eine individuelle Anpassung der für die Interaktion notwendigen Beschreibungen vornehmen zu müssen und trotzdem eine geeignete (dynamische) Anpassung an die aktuelle Ausführungsumgebung bzw. den Kontext des Benutzers zu erreichen.

Vorüberlegung

Liegen Anweisungen zur Repräsentation von Benutzeroberflächen und Interaktionsmöglichkeiten in der Prozessbeschreibung in konkreter Form statisch vor, so können diese zur Laufzeit nicht individuell verarbeitet und in Abhängigkeit des aktuellen Ausführungskontextes angepasst werden. Sie müssen daher entweder zur Laufzeit des Prozesses von einem externen Speicherort bezogen werden oder bereits vor der Verteilung des Prozesses ausgetauscht und auf die jeweilige Ausführungsplattform am Zielort der Verteilung angepasst werden. Insbesondere im Fall von mobilen Ausführungseinheiten kann jedoch keine dauerhafte Netzwerkanbindung für eine Online-Anpassung vorausgesetzt werden und die tatsächliche Ausführungsplattform kann bei einer dynamischen Verteilung nicht im Voraus bestimmt werden. Die notwendige Flexibilität für eine dynamische Anpassung der Benutzungsschnittstellen kann daher für das vorliegende Problem nur über eine geeignete Abstraktion der Interaktion mit dem Benutzer erreicht werden [ZBV09, ZVBK09].

Eine abstrakte Beschreibung der Benutzeroberflächen bzw. Interaktionsmöglichkeiten bietet großes Potential, um die am Ausführungsort konkret vorherrschenden technischen Charakteristika der Ausführungseinheit (wie z. B. die Beschaffenheit von Displays oder Eingabegeräten), aber auch die aktuelle Situation des Benutzers (z. B. Aufenthalt in einer Besprechung oder Führen eines Kraftfahrzeugs) berücksichtigen zu können. Eine solche Abstraktion erfordert jedoch auch eine geeignete Beschreibungssprache, welche die vom Prozessmodellierer intendierte Benutzerinteraktion ausdrücken kann und erlaubt deren Elemente am Ort der Ausführung zu interpretieren und kontextspezifisch in eine Benutzeroberfläche umzusetzen. Geeignete Beschreibungen können im Wesentlichen durch die Definition von den in Abschnitt 5.8.5 vorgestellten *CTT-Modellen* aufgebaut werden. Kongruent zu dem bisher in dieser Arbeit verfolgten Konzept der Dienstorientierung kann die Aufgabe der Verarbeitung der (erweiterten) CTT-Modelle von einem generischen Dienst wahrgenommen werden, welcher die Beschreibung der durchzuführenden Interaktion (lokal oder entfernt) entgegennimmt und eine angepasste Benutzungsschnittstelle zurückliefert. Bietet eine Ausführungseinheit einen solchen Dienst an, so kann dieser für beliebige Prozesse und deren abstrakte Beschreibungen der Benutzeroberflächen bzw. Interaktionsmöglichkeiten wiederverwendet werden [ZBV09, ZVBK09].

Betrachtet man die (abstrakte) Beschreibung der Benutzeroberflächen und Interaktionsmöglichkeiten als Teil der Prozessbeschreibung, so kann aus den oben genannten Gründen im vorliegenden Fall nicht – wie bisher vorgeschlagen – vollständig auf eine Anpassung der Prozessbeschreibung verzichtet wer-

den. Da im Idealfall jedoch nur die aus der Prozessbeschreibung referenzierten (ggf. sogar separaten) Beschreibungen angepasst oder ausgetauscht werden müssen, werden der eigentliche Kontrollfluss des Prozesses und dessen fachlicher Inhalt in der Regel nicht berührt. Es kann somit weiterhin aus Prozesssicht von einem nicht-invasiven Konzept gesprochen werden, auch wenn für die Flexibilisierung der Benutzungsschnittstellen im Kontext verteilt ausgeführter Prozesse die Beschreibung ihrer Darstellung teilweise manipuliert werden muss. Im folgenden Abschnitt wird aufgezeigt, wie die Entwicklung, Einbettung und Ausführung abstrakter Benutzungsschnittstellen in den Lebenszyklus verteilt ausgeführter Prozesse integriert werden kann.

Entwicklungsmethodik

Kern der hier vorgeschlagenen Entwicklungsmethodik ist die Integration einer geeigneten Repräsentation von Benutzeroberflächen und Interaktionsmöglichkeiten für diejenigen Aktivitäten von verteilt ausgeführten Prozessen, welche zur Laufzeit ganz oder teilweise durch einen menschlichen Prozessteilnehmer bearbeitet werden sollen. Abbildung 6.40 zeigt als Ausgangspunkt eine (bestehende) Prozessbeschreibung, welche den Kontrollfluss des fachlichen Prozesses mit seinen Aktivitäten in einem Prozessmodell PM_m kapselt (Schritt 1). Werden in dem modellierten Prozess Aktivitäten als *manuell* bzw. *interaktiv* gekennzeichnet (vgl. Abschnitt 2.7.3) und ist eine Benutzungsschnittstelle nicht bereits durch den Aufruf einer integrierten Anwendung vorgegeben, so müssen zur Entwicklungszeit zusätzliche Beschreibungen für die Darstellung der Benutzungsschnittstellen erstellt werden.

Die für die Modellierung von interaktiven Aktivitäten in jedem Fall anfallende Arbeit der Beschreibung von Benutzungsschnittstellen kann idealerweise durch ein (graphisches) Werkzeug unterstützt werden, welches die modellierten Benutzungsschnittstellen in ein maschinenlesbares Format überträgt. Werden konkrete Benutzungsschnittstellen erzeugt, so kann das Ergebnis dieser Modellierung zum Beispiel aus einem HTML-Formular bestehen. Bei dem hier vorgestellten Ansatz wird bei der Modellierung von Benutzungsschnittstellen auf Basis bestehender Modellierungsansätze und Beschreibungssprachen ein *abstraktes Beschreibungsformat* (*Abstract Interface Description*, kurz *AID*) erzeugt (Schritt 2 in Abbildung 6.40). Wie auch eine konkrete Beschreibung von Benutzungsschnittstellen wird die AID beim Deployment des Prozessmodells integriert (Schritt 3) und kann nach einer Instantiierung des Prozesses (Schritt 4) auf beliebigen Ausführungseinheiten auf Basis einer lokalen Richtlinie für die Interaktion in eine konkrete Benutzungsschnittstelle (*Concrete User Interface*, kurz *CUI*) transformiert werden (Schritt 5). Hierbei kann die lokale Richtlinie nur gerätespezifisch und damit für die betrachtete Ausführungseinheit statisch sein oder kann aufgrund von veränderlichen Kontextdaten der Ausführungseinheit selbst dynamisch anpassbar sein. Die resultierende konkrete Benutzungsschnittstelle CUI_{mj} ist somit jeweils spezi-

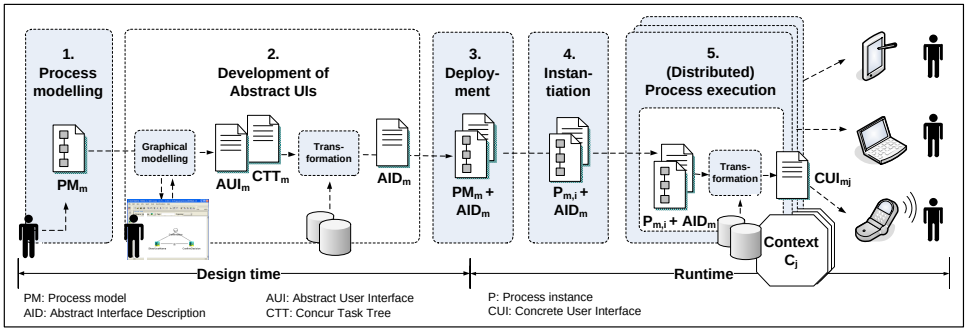


Abbildung 6.40: Entwicklungsmethodik abstrakter Benutzungsschnittstellen (horizontale Verteilung) (ZVBK09)

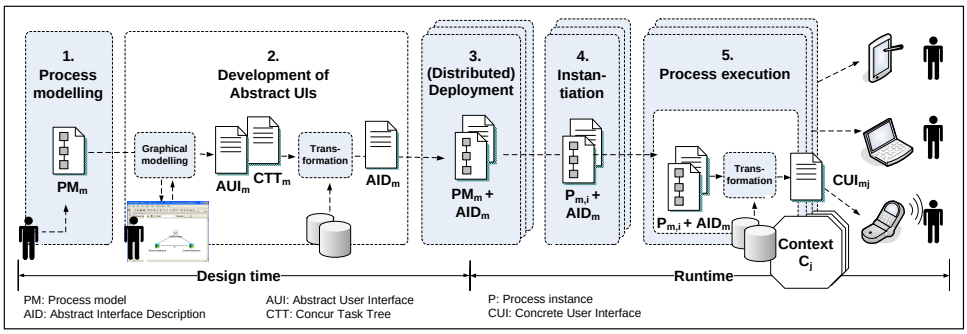


Abbildung 6.41: Entwicklungsmethodik abstrakter Benutzungsschnittstellen (vertikale Verteilung) (ZBV09)

fisch für das Prozessmodell PM_m und für den Kontext der Ausführungseinheit C_j.

Für das Konzept abstrakter Benutzungsschnittstellen ist es zudem unerheblich, ob der Prozess verteilt über mehrere Ausführungseinheiten ausgeführt wird (*horizontale Verteilung*, vgl. Abbildung 6.40) oder ob verschiedene Instanzen eines Prozessmodells als Ganzes, aber jeweils auf verschiedenartigen Ausführungseinheiten ausgeführt werden (*vertikale Verteilung*, vgl. Abbildung 6.41). Es können somit mit einem solchen Ansatz auch andere Verteilungsmodelle als die Migration von Prozessinstanzen unterstützt werden [ZBV09, ZVBK09] (vgl. Abschnitt 4.3.3).

Die dargestellten Schritte 2 und 5, welche den klassischen Lebenszyklus prozessorientierter Anwendungen erweitern, werden in den folgenden Abschnitten genauer vorgestellt. Dabei wird zunächst die Entwicklung und Beschreibung von abstrakten und modalitätsunabhängigen Benutzerinteraktionen (vgl. Abschnitt 6.4.2) sowie deren Einbettung in prozessorientierte Anwendungen thematisiert (vgl. Abschnitt 6.4.3). In Abschnitt 6.4.4 wird beschrieben, wie zur Laufzeit aus den abstrakten Beschreibungen wieder konkrete Benutzungsschnittstellen gewonnen werden können.

6.4.2 Metamodell abstrakter Benutzungsschnittstellen

Auf Basis einer Analyse bestehender Ansätze für die Beschreibung abstrakter Benutzungsschnittstellen [Vil08, ZBV09, ZVBK09] kann der *Concur-Task-Tree*-Ansatz (kurz *CTT*-Ansatz) als geeignete Grundlage für die Darstellung von Benutzungsschnittstellen im Kontext verteilt ausgeführter Prozesse identifiziert werden (vgl. auch Zusammenfassung in Abschnitt 5.8.5). Grund hierfür ist vor allem die gute Abbildbarkeit von interaktiv auszuführenden Aktivitäten eines Prozesses auf die *Aufgaben* (*Tasks*) in einem CTT-Modell. Desweiteren ist der Abstraktionsgrad von CTT-Modellen bei der Interaktion mit dem Benutzer nicht nur auf eine visuelle Darstellung von Benutzeroberflächen beschränkt, sondern bietet auch Potential für den Einsatz von anderen Ein- und Ausgabemodalitäten, welche zukünftig im Kontext sich rasch entwickelnder mobiler Ausführungseinheiten eine Rolle spielen können. Zudem kann die Erstellung von CTT-Modellen bereits durch eine geeignete werkzeuggestützte Entwicklungsmethodik (*ConcurTaskTreeEnviroment*, kurz *CTTE* [MPS02] in Verbindung mit *Transformation Environment for Interactive Systems Representations*, kurz *TERESA* [MPS04]) begleitet werden.

Ein auf den Kontext verteilt ausgeführter Prozesse spezialisiertes Metamodell für die Beschreibung abstrakter Benutzerinteraktionen nach dem CTT-Ansatz ist in Abbildung 6.42 dargestellt. Das Element *Presentation* stellt dabei eine einzelne Benutzungsschnittstelle dar. Unter Verwendung der visuellen Modalität kann ein *Presentation*-Element z.B. auf eine Bildschirmseite abgebildet werden. Jedes *Presentation*-Element besteht dazu aus einer sachlichlogisch zusammenhängenden, nichtleeren Menge von Elementen des Typs *Interactor* als Repräsentation von einzelnen Aktionen, welche im Rahmen der Interaktion innerhalb des *Presentation*-Kontextes durchgeführt werden sollen. Ein *Interactor* kann zum Beispiel bei Verwendung einer visuellen Modalität auf ein Textfeld oder bei einer sprachgesteuerten Interaktion auf eine akustische Aufforderung zum Sprechen abgebildet werden. Über ein Sortierungskriterium kann angegeben werden, in welcher Reihenfolge die *Interactor*-Elemente angeordnet werden sollen. Die durch einen *Interactor* zu erreichende Funktionalität wird durch sein Kindelement *UserInteractionDetails* genauer spezifiziert. Es enthält als wesentlichen Inhalt den Typ der Interaktion (*UserInteractionType*), welcher die Werte *Selection*, *Edit* oder *Control* annehmen kann. Das Element *Selection* drückt aus, dass dem Benutzer eine festgelegte Menge von Daten aufzuführen sind, aus welcher ein oder mehrere Datenelemente augewählt werden können, wobei diese nicht verändert werden dürfen. Das Element *Edit* besagt, dass der angezeigte Inhalt oder ein leerer Inhalt verändert bzw. dass neuer Inhalt eingegeben werden darf. Das Element Control dient im Allgemeinen der Aktivierung von Aktionen, z.B. der Navigation innerhalb der Benutzungsschnittstelle oder der Beendigung der Benutzerinteraktion [ZBV09, ZVBK09]. Zur Entwicklungszeit vorbelegte Daten oder Referenzen auf Datenobjekte anderer Interaktionselemente werden im Behälter *DataObject* gespeichert. Dabei kann jedes Interaktionselement aufgrund der

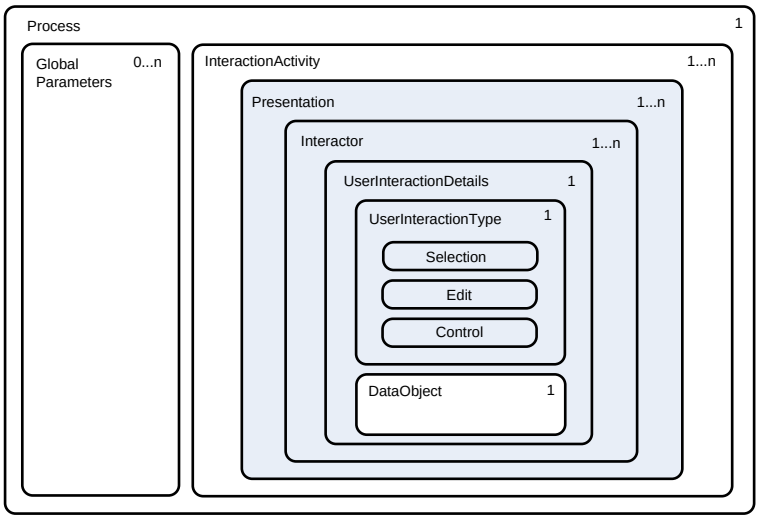


Abbildung 6.42: Metamodell für die Definition abstrakter Benutzungsschnittstellen in prozessorientierten Anwendungen (ZVBK09, ZBV09)

strengen Granularität nur genau ein *DataObject* besitzen. Dieses kann jedoch auch aus einer Liste von Daten bestehen, welche zum Beispiel in einer *Selection* angezeigt werden sollen oder die Eingabe des Benutzers aus einem *Edit*-Element aufnehmen [ZBV09, ZVBK09].

Die dargestellten Interaktionselemente können beliebig miteinander kombiniert werden und sind jeweils durch weitere beschreibende Eigenschaften gekennzeichnet, welche den Modellierer das beabsichtigte Verhalten der Benutzerinteraktion plattformunabhängig spezifizieren lassen [ZBV09]. Zum Beispiel kann angegeben werden, ob bei einem *Selection*-Element nur jeweils ein Datenelement ausgewählt werden darf (*SingleSelection*), oder ob eine Mehrfachauswahl möglich sein soll (*MultipleSelection*). Desweiteren kann bei einem Edit-Element angegeben werden, ob als Eingabe eine Zeichenkette oder numerische Daten zu erwarten sind. Auf der Ebene der *UserInteractionDetails* kann z. B. durch das Attribut *Optional* festgelegt werden, ob die mit dem Interaktionselement verbundene Aufgabe für den Benutzer verpflichtend ist oder ob dieses bei der Bearbeitung auch übersprungen werden darf. Zudem verfügt jeder *Interactor* über eine Beschreibung (*Description*), welche beliebige Erläuterungen zu der durchzuführenden Aufgabe aufnehmen kann. Eine genaue Auflistung aller Elemente und Attribute kann der Arbeit von VILENICA [Vil08] entnommen werden.

Die Einbettung der beschriebenen Interaktionen erfolgt über die Zuordnung von *Presentation*-Elementen zu einer *Interaktionsaktivität* (*InteractionActivity*). Diese repräsentiert eine fachliche Aktivität in der übergeordneten prozessorientierten Anwendung, für welche die Benutzungsschnittstelle erzeugt werden soll. Die Zuordnung zu einem bestimmten Prozessmodell erfolgt über das Element *Process*. Obwohl der originäre CTT-Ansatz erlaubt, Abhängigkeiten zwischen einzelnen *Presentation*-Elementen zu spezifizieren, wird für den vor-

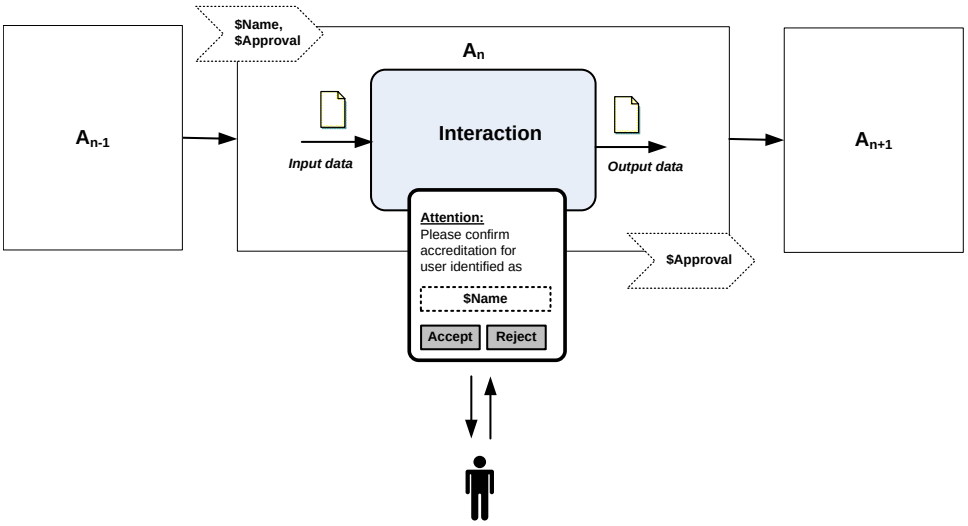
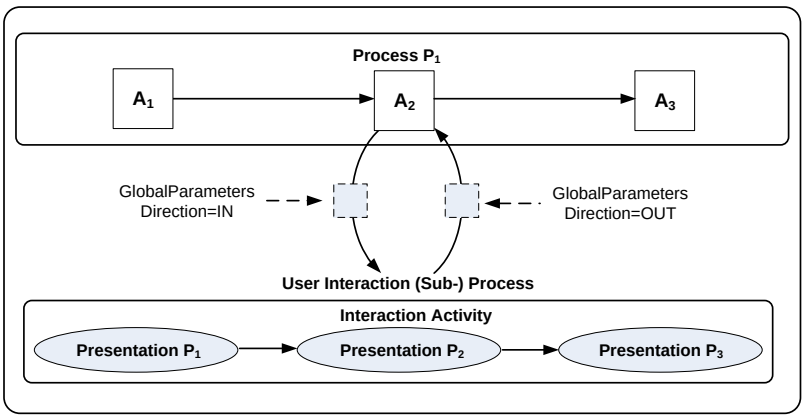


Abbildung 6.43: Einbettung und Datenfluss einer Interaktion in einer Aktivität

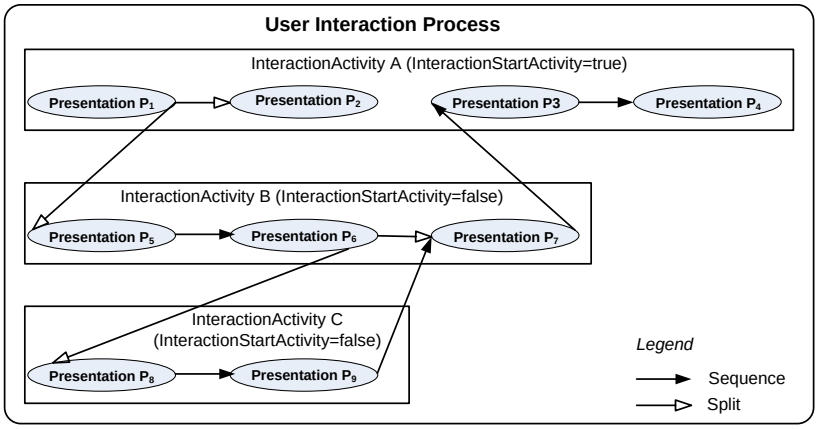
liegenden Einsatzzweck keine ausreichende Unterstützung für die Modellierung des Datenflusses zwischen dem Kontrollfluss im fachlichen Prozess und den einzelnen Interaktionselementen erreicht.

Abbildung 6.43 verdeutlicht diese Problematik an einem Beispiel. Hierbei soll die Aktivität A_n die Interaktion mit einem menschlichen Prozessteilnehmer umsetzen, um eine Bestätigung über die Teilnahme einer genannten Person an einer Veranstaltung zu bestätigen. Die hierfür benötigten Variablen *$Name* und *$Approval* werden über den Input-Container von A_n zur Verfügung gestellt. Der Wert von *$Name* soll dem Benutzer angezeigt werden und die positive oder negative Bestätigung soll später wieder in die Prozessvariable *$Approval* zurückgeschrieben werden, so dass sie über den Output-Container von A_n im weiteren Prozessverlauf zur Verfügung steht. Da beide Prozessvariablen erst zur Prozesslaufzeit determiniert werden können, ist eine statische Integration als *DataObjects* zur Entwicklungszeit nicht möglich. Die Prozessvariablen müssen daher der Interaktionsaktivität übergeben werden und dort den intendierten Interaktionselementen zugeordnet werden. Diese Rolle wird durch das Element *GlobalParameters* wahrgenommen, in denen die Prozessvariablen, welche in dem betrachteten Prozessmodell den Interaktionsaktivitäten zur Verfügung stehen sollen, übergeben werden können und auf lokale *DataObjects* abgebildet werden [ZBV09].

Eine mögliche Erweiterung des Modells erlaubt, durch die Einbettung von Kontrollflusselementen die Abfolgen von mehreren Interaktionen innerhalb einer Aktivität zu steuern. Während das hier vorgeschlagene Konzept es ermöglicht, (abstrakte) Interaktionen in Prozesse einzubinden, ist es somit ebenfalls möglich, Prozesse in Interaktionen einzubinden und damit insbesondere auch komplexere Interaktionen innerhalb einer Aktivität auf Prozessebene zu kapseln. Ein Anwendungsbeispiel ist die Darstellung elektronischer Fragebögen, welche die Prozessorientierung (ausschließlich) zur Steuerung der



(a) Makroperspektive: Einbettung einer Interaktionsaktivität in eine fachliche Aktivität



(b) Microperspektive: Verknüpfung mehrerer Interaktionsaktivitäten zu einem Interaktionsprozess

Abbildung 6.44: Makro- und Microperspektive von Interaktionsaktivitäten (ZVBK09)

Interaktion mit dem Benutzer einsetzt [ZBV09]. Es kann somit zwischen der oben genannten Makroperspektive und einer Mikroperspektive unterschieden werden. Abbildung 6.44 verdeutlicht beide Varianten anhand je eines Beispiels.

In der Makroperspektive ist jeder Aktivität des fachlichen Prozesses genau eine Interaktionsaktivität zugeordnet, welche eine Benutzungsschnittstelle für diese Aktivität repräsentiert (vgl. Abbildung 6.44a). Durch die Mikroperspektive (vgl. Abbildung 6.44b) kann die Abfolge von mehreren Interaktionen innerhalb einer Aktivität des fachlichen Prozesses von vorhergehenden Eingaben abhängig gemacht werden. Dazu muss angegeben werden, mit welchen Interaktionsaktivitäten die Interaktion begonnen werden kann (*InteractionStartActivity*). Zudem müssen die Interaktionsaktivitäten auf der Ebene der Interaktionsbeschreibung durch Kontrollflusskonnektoren und -bedingungen verknüpft werden. Es ist denkbar, dass im Interaktionsprozess dieselben Kontrollflusselemente zum Einsatz kommen, wie in dem übergeordneten fachlichen Prozess [ZVBK09]. Da dieselbe Interaktion jedoch auch durch die Verknüpfung

mehrerer fachlicher Aktivitäten mit jeweils einzelnen Interaktionsaktivitäten erreicht werden kann, wird die weiterführende Mikroperspektive im weiteren Verlauf dieser Arbeit nicht genauer betrachtet. Details hierzu können der Arbeit von VILENICA [Vil08] entnommen werden.

6.4.3 Modellierung von interaktiven Aktivitäten

Da das Metamodell abstrakter Benutzungsschnittstellen auf dem CTT-Ansatz beruht, ist es möglich, für die Modellierung der in den Abbildungen 6.40 und 6.41 dargestellten *Abstract Interface Descriptions* (*AID*s) bereits bestehende Werkzeuge für die Entwicklung von Task-Modellen einzusetzen. Der Entwicklungsvorgang der Benutzungsschnittstellen kann somit im Rahmen der CTT-Modellierung graphisch unterstützt und die Erzeugung von *AID*-Beschreibungen zu einem großen Teil automatisiert werden. In diesem Abschnitt wird gezeigt, wie die Modellierung von CTT-Modellen genutzt werden kann, um daraus über eine automatische Transformation abstrakte Benutzungsschnittstellen für verteilt auszuführende Prozesse zu generieren. Dazu wird beispielhaft die Modellierung mit TERESA [MPS04] herangezogen.

Abbildung 6.45 zeigt eine Verfeinerung von Schritt 2 in Abbildung 6.40 bzw. 6.41. Der Entwicklungsvorgang beginnt mit der (graphischen) Modellierung von CTT-Modellen, wobei die durch die interaktive Aktivität des Prozesses definierte Aufgabe den Wurzelknoten des CTT-Baumes darstellt. In der zweiten Ebene werden zunächst die Tätigkeiten des Benutzers identifiziert, welche zur Erreichung des übergeordneten Ziels durchgeführt werden müssen. Eine wichtige Unterscheidung besteht dabei darin, ob der Benutzer den Zustand der (prozessorientierten) Anwendung ändern oder mit deren Hilfe an Informationen gelangen soll [Pat00, Vil08]. Zudem können die Tätigkeiten des Benutzers eine zeitlich-logische Reihenfolge aufweisen. Alle identifizierten Tätigkeiten ab der zweiten Ebene des Baumes können dazu durch binäre Operatoren verknüpft werden (vgl. [Pat00, Vil08] zu Details).

Jede Aufgabe wird nun schrittweise weiter detailliert, bis sie (nach Ansicht des Modellierers) nicht weiter verfeinert werden kann. Die resultierenden Blätter des CTT-Baumes stellen in Folge die atomaren Einheiten dar, mit denen der Benutzer später interagieren soll [Pat00, Vil08]. Sie können auf die Interaktionselemente (*Interactor*-Elemente) des oben dargestellten Metamodells (vgl. Abschnitt 6.4.2) abgebildet werden. Die hierarchische Dekomposition der Aufgabe mit dem CTT-Modell erlaubt zudem die Darstellung von Zusammenhängen zwischen den jeweiligen Kindelementen eines Knotens. So können alle von einem Knoten abgeleiteten *Interactor*-Elemente ab einer bestimmten Ebene einem übergeordneten *Presentation*-Element zugeordnet und somit später dem Benutzer zum Beispiel auf derselben Bildschirmseite angezeigt werden. Die binären Operatoren geben die Reihenfolge der *Interactor*-Elemente und somit den späteren Aufbau der etwaigen Bildschirmseite vor. Schritt 1 in Abbildung 6.45 zeigt ein einfaches Beispiel für die Modellierung eines CTT-Baumes als Grundlage der in Abbildung 6.43 dargestellten möglichen

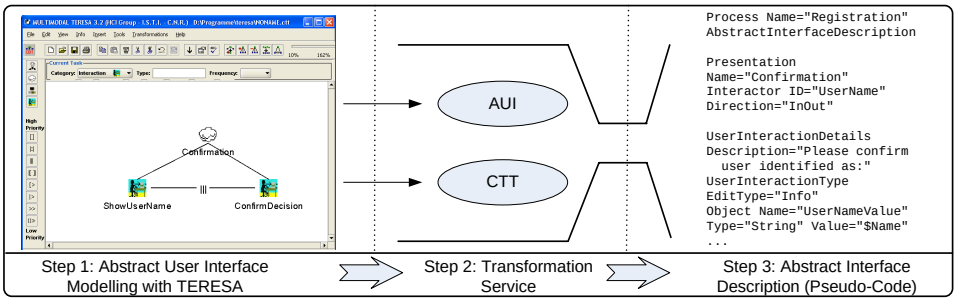


Abbildung 6.45: Modellierung abstrakter Benutzungsschnittstellen am Beispiel von TERESA (MPS04, ZVBK09)

Repräsentation einer Bestätigung über die Teilnahme einer genannten Person an einer Veranstaltung. Auf der obersten Ebene steht dabei das abstrakte Ziel der Interaktion (*Confirmation*). Auf der Ebene der Blätter des Baumes werden die nicht weiter verfeinerbaren Interaktionen der Ausgabe (*ShowUserName*) und der vom Benutzer erwarteten Eingabe (*ConfirmDecision*) angezeigt.

Nach der Erstellung des graphischen CTT-Modells erlaubt TERESA auf Basis weiterer Eingaben des Modellierers eine Verfeinerung der Interaktionsbeschreibung, zum Beispiel durch die Eingabe von Datenwerten, welche dem Benutzer bei der Interaktion präsentiert werden sollen, oder durch die Konfiguration der in Abschnitt 6.42 genannten Attribute (z.B. Optionalität eines Interaktionselements). Desweiteren erfolgt auf Basis der Spezifikation der Endgeräte, auf denen die Interaktion stattfinden soll, eine semi-automatische Transformation in eine oder mehrere *konkrete* Interaktionskomponenten für die jeweils angegebene(n) Plattform(en) [Pat00, Vil08]. Dieser Schritt wird im Rahmen des hier vorgeschlagenen Konzepts jedoch nicht zur Entwicklungszeit, sondern während der Prozessausführung auf dem jeweiligen Endgerät ausgeführt, welches dem Prozessteilnehmer für die Interaktion mit der prozessorientierten Anwendung dient (vgl. Abschnitt 6.4.4). Daher werden die aus der Modellierung resultierenden abstrakten Beschreibungen in Form des CTT-Modells und der daraus ableitbaren abstrakten Schnittstellenbeschreibung (*Abstract User Interface*, kurz *AUI*) weiterverarbeitet, da diese zusammen alle benötigten Informationen zu den einzelnen Interaktionskomponenten und zu deren zeitlich-logischen Beziehungen beinhalten [Vil08, ZVBK09] (Schritt 2 in Abbildung 6.45).

Die aus der Verknüpfung beider Beschreibungen gewonnene *Abstract Interface Description* (*AID*) kann nun auf geeignete Weise mit der dazugehörigen prozessorientierten Anwendung verknüpft werden. Dabei ist es prinzipiell möglich, dass die bereits bestehende Beschreibung der Benutzungsschnittstelle der betrachteten (interaktiv auszuführenden) Aktivität durch die AID ersetzt bzw. in der Aktivität eine Referenz auf die AID gesetzt wird (vgl. Abbildungen 5.24a bis 5.24c). Um jedoch während der dynamischen Verteilung eines Prozesses eine möglichst einheitliche Behandlung für sowohl automatisiert als auch interaktiv auszuführenden Aktivitäten zu erlauben, wird in dieser Arbeit

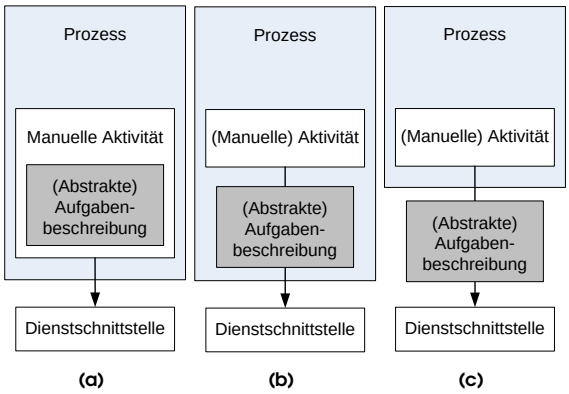


Abbildung 6.46: Erweiterung der Varianten für eine dienstorientierte Integration von (abstrakten) Aufgabenbeschreibungen für die manuelle Aktivität eines Prozesses (vgl. Abbildungen 5.24)

ein dienstorientierter Ansatz für die Integration von Interaktionsaktivitäten vorgeschlagen.

Durch die Möglichkeit zur Spezifikation abstrakter Benutzungsschnittstellen können die beiden in den Abbildungen 5.24d und 5.24e dargestellten Varianten zur dienstorientierten Integration von Aufgabenbeschreibungen für manuelle Aktivitäten eines Prozesses um drei weitere Varianten ergänzt werden, welche in Abbildung 6.46 dargestellt sind. Abhängig von der durch die Prozessbeschreibungssprache vorgegeben Art und Weise der Integration der Benutzungsschnittstellen können diese entweder durch eine spezielle Art von Aktivität integriert werden (vgl. Abbildung 6.46a), als Teil der Prozessbeschreibung durch eine Aktivität referenziert werden (vgl. Abbildung 6.46b) oder von einem Zugriffsort außerhalb der Prozessbeschreibung eingelesen werden (vgl. Abbildung 6.46c). In allen drei Fällen wird dabei die abstrakte Aufgabenbeschreibung über die (manuelle) Aktivität an einen generischen Dienst übergeben, welcher die Beschreibung verarbeitet und in eine konkrete Benutzungsschnittstelle transformiert. Da die für die Erzeugung einer konkreten Benutzungsschnittstelle benötigte Beschreibung komplett durch den auf der lokalen Ausführungsumgebung auszuführenden Prozess festgelegt ist, ist das Vorhalten von mehreren speziell angepassten Diensten zur Abbildung von verschiedenartigen Interaktionen nicht vorteilhaft. Es genügt daher pro Ausführungsumgebung einen einzigen lokalen Dienst zur Interaktion mit dem Benutzer bereitzustellen, um alle anfallenden Interaktionen aus lokalen und verteilt ausgeführten Prozessen ohne Notwendigkeit zu einer dauerhaften Netzwerkanbindung durchführen zu können. Ein Vorschlag für einen solchen *Interaktionsdienst* wird in Abschnitt 6.4.4 vorgestellt.

Bei den Varianten in den Abbildungen 6.46b und 6.46c kann die Aktivität als manuell bzw. interaktiv auszuführende Aktivität gekennzeichnet werden oder es kann bei der Spezifikation des Prozesses transparent gehalten werden, ob die Aktivität automatisch (d. h. durch eine Softwareanwendung) oder durch

einen Benutzer ausgeführt wird. Bei einer transparenten Modellierung muss der Interaktionsdienst wie alle anderen Dienste bzw. Anwendungen aus der jeweiligen Aktivität referenziert werden und die Ein- und Ausgabeparameter des Dienstes müssen mit den Variablen des Prozesses, die in der Interaktion verwendet werden sollen, in Verbindung gebracht werden. Zudem kann die abstrakte Beschreibung nicht implizit an den Interaktionsdienst übergeben werden, sondern stellt einen weiteren expliziten Aufrufparameter dieses Dienstes dar. Im einfachsten Fall kann daher die gesamte abstrakte Beschreibung der durchzuführenden Interaktion als Konstante innerhalb eines Datenfeldes des Prozesses transportiert werden bzw. durch ein solches Datenfeld referenziert werden. Dabei ist es schließlich auch möglich, die Interaktionsbeschreibung zur Laufzeit von einem externen System zu beziehen. Eine Offline-Ausführung der Benutzerinteraktion ist in diesem Fall jedoch natürlich nicht mehr uneingeschränkt möglich.

6.4.4 Dynamische Interaktionsverarbeitung

Die im Entwicklungsprozess von TERESA bei der Modellierung vorgenommene Konkretisierung der abstrakten Benutzungsschnittstellen für verschiedene Geräteklassen wird bei dem hier vorgestellten Ansatz zur Laufzeit des (verteilt ausgeführten) Prozesses wieder aufgegriffen. Die grundlegende Idee dabei ist, dass jede an der Ausführung eines dynamisch verteilten Prozesses beteiligte Ausführungseinheit bereits zumindest eine eigene Art von Benutzungsschnittstelle für die lokal ausgeführten (nicht verteilten) Prozesse besitzt, die nun durch die abstrakten Interaktionsbeschreibungen auch von beliebigen verteilt ausgeführten Prozessen zur Interaktion mit dem Benutzer verwendet werden können. Handelt es sich zum Beispiel um eine vollwertige Prozess-Engine, die auf einem leistungsfähigen Server läuft, so ist die Anbindung lokaler oder entfernter Benutzer unter Umständen über eine webbasierte Anwendung realisiert, welche über einen Internetbrowser von einem Desktop-PC oder einem Notebook zugegriffen werden kann. Handelt es sich hingegen um eine leichtgewichtige Prozess-Engine für die Offline-Ausführung von mehreren aufeinander folgenden Prozessschritten auf einem mobilen Gerät (zum Beispiel einem Mobiltelefon oder einem Tablet-PC), so werden Benutzerinteraktionen ggf. durch eine proprietäre lokale Anwendung auf diesem mobilen Gerät durchgeführt. Die resultierenden Benutzungsschnittstellen können sich dabei schon aufgrund der verschiedenen Geräteeigenschaften wesentlich unterscheiden. Hinzu kommt, dass (insbesondere bei einer mobilen Ausführung von Prozessschritten) auch der Kontext des Benutzers bei der Interaktion Berücksichtigung finden kann.

Abbildung 6.47 zeigt die grundsätzliche Methodik zur Verarbeitung der Beschreibung abstrakter Benutzungsschnittstellen zur Laufzeit des Prozesses als Verfeinerung von Schritt 5 in den Abbildungen 6.40 und 6.41. Erreicht der Kontrollfluss des Prozesses eine nach Abschnitt 6.4.3 integrierte (manuelle) Aktivität (hier A_2), so wird der in der Prozessbeschreibung für diese Aktivität de-

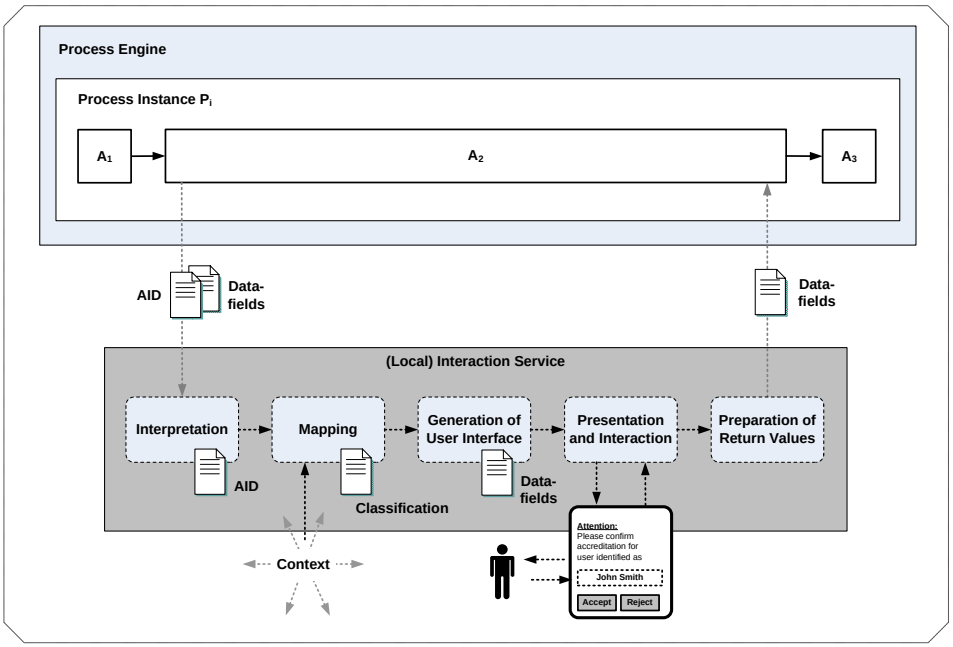


Abbildung 6.47: Verarbeitung abstrakter Benutzungsschnittstellen zur Laufzeit der (interaktiv auszuführenden) Aktivität A_2 (ZBV09)

finierte *Interaktionsdienst* (*Interaction Service*) aufgerufen. Die Beschreibung der abstrakten Benutzungsschnittstelle in Form der AID sowie die zur Interaktion benötigten Prozessvariablen (*Data Fields*) werden als Aufrufparameter an den Interaktionsdienst übertragen. Dabei ist die Abbildung der Prozessvariablen auf die Interaktionselemente durch die Interaktionsbeschreibung festgehalten (Element *GlobalParameters*).

Der Interaktionsdienst kapselt nun die zur Entwicklungszeit durch TERESA unterstützte Transformation der abstrakten Beschreibung in konkrete Interaktionselemente (vgl. Abschnitt 6.4.3. Die Transformation der CTT-Modelle und der abstrakten Beschreibung von Benutzungsschnittstellen (*AUI*) kann bei TERESA jedoch originär nicht ohne zusätzliche Eingaben des Benutzers erfolgen, da zur Entwicklungszeit prinzipiell eine Bereitstellung für sehr viele verschiedene Plattformen denkbar ist. Bei dem hier verfolgten Ansatz wird jedoch zur Laufzeit eines Prozesses jede Instanz einer Aktivität durch genau eine Ausführungseinheit verwaltet, wodurch die Konfiguration aus Geräteeigenschaften und aktuellem Kontext zum Zeitpunkt der Ausführung eindeutig bestimmt werden kann. Das bei TERESA erforderliche manuelle Vornehmen dieser Einstellungen kann daher entfallen und durch einen einfachen automatisierten Prozess ersetzt werden.

Der Interaktionsdienst verarbeitet seine beiden Aufrufparameter, indem die abstrakte Beschreibung der Benutzungsschnittstellen interpretiert und auf lokal verfügbare konkrete Interaktionselemente abgebildet wird. Dazu besitzt ein Interaktionsdienst eine (anpassbare) Klassifikation von konkreten Interaktionselementen mit Zuordnung zu den Elementen des in Abschnitt 6.4.2

vorgestellten abstrakten Metamodells. Bei der Auswahl eines konkreten Interaktionselements spielt dabei z. B. auch die Kardinalität der anzuzeigenden bzw. einzugebenden Daten eine Rolle. Steht ein optimales Element zur Repräsentation der abstrakt definierten Interaktion auf dem aktuellen System nicht zur Verfügung, so wird ein Element gewählt, welches dem angegebenen Verhalten der Benutzungsschnittstelle am nächsten kommt. Stehen mehrere Möglichkeiten zur Abbildung zur Verfügung, so kann auf Basis der aktuellen Kontextdaten des Benutzers eine geeignete Repräsentation gewählt werden. Nach der Auswahl wird die resultierende konkrete Benutzungsschnittstelle erzeugt und entsprechend der Zuordnung der *GlobalParameters* mit den Daten aus den übergebenen Prozessvariablen verknüpft. Basierend auf den Geräteeigenschaften (z. B. insbesondere der Displaygröße) und der zeitlich-logischen Vorgaben aus der abstrakten Interaktionsbeschreibung werden die Interaktionslemente nun zu sinnvollen Gruppen von *Presentations* zusammengefasst und z. B. als Zusammenstellung eines Bildschirminhalts abgebildet. Die resultierende Benutzungsschnittstelle wird dem Benutzer nun entweder aktiv präsentiert (d. h. der Benutzer wird über das Vorliegen einer neuen Aufgabe informiert) und/oder die Benutzungsschnittstelle wird entsprechend angezeigt, sobald der Benutzer den Interaktionsdienst von sich aus aufruft. Ist die Interaktion mit dem Benutzer beendet, so werden die durch Eingaben des Benutzers veränderten Werte von Prozessvariablen als Rückgabewert des Interaktionsdienstes an die aufrufende Aktivität zurückgegeben und so in den Prozess zurückgeschrieben. Der Interaktionsdienst kann nun beendet werden oder für weitere Interaktionen zur Verfügung stehen [ZBV09].

Der Interaktionsdienst kann im Rahmen dieses Konzepts geeignet an die Leistungsfähigkeit, die Eigenschaften und den möglichen Kontext verschiedener Ausführungseinheiten angepasst werden. Es besteht jedoch darüber hinaus auch die Möglichkeit, aus der prozessorientierten Anwendung auch Interaktionsdienste anderer Ausführungseinheiten oder Interaktionsdienste ohne Anbindung an eine bestimmte Ausführungseinheit aufzurufen. Somit können auch Benutzer in die Ausführung einzelner Aufgaben einbezogen werden, welche selbst keiner Ausführungseinheit zugeordnet sind bzw. keine „eigene“ Ausführungseinheit besitzen. Es können somit auch mit relativ geringem Aufwand Anwender in *Process-Management-as-a-Service*-Szenarien unterstützt werden (vgl. Abschnitt 4.2.6). Eine mögliche Implementierung des Interaktionsdienstes für mobile Endgeräte ist Inhalt von Kapitel 7.6.

6.4.5 Zusammenfassung und Einordnung

Auf Basis dienstorientierter Architekturen können Aktivitäten verteilt ausgeführter Prozesse von verschiedenen Ausführungseinheiten in gleicher Weise ausgeführt oder durch die jeweils lokale Ausführungseinheit neu gebunden werden. Für Aktivitäten, welche die Interaktion eines menschlichen Prozessteilnehmers erfordern, ist eine vergleichbare Abstraktion bisher nicht gegeben. Insbesondere bei dynamisch verteilt ausgeführten Prozessen ist eine

statische Festlegung von konkreten Benutzungsschnittstellen jedoch ein Problem, da die tatsächlichen Geräteeigenschaften und Ausführungskontexte für die Repräsentation geeigneter Benutzungsschnittstellen erst zur Laufzeit bestimmt werden können. Sowohl die Kapselung von (potentiellen) Benutzungsschnittstellen hinter Dienstschnittstellen als auch die dynamische Erzeugung von Benutzungsschnittstellen durch eine zentrale Instanz werden daher insbesondere im Kontext mobiler Ausführungseinheiten, für welche manuell auszuführende Aktivitäten eine große Bedeutung haben, als nicht vorteilhaft beurteilt.

Um auch eine weitestgehend uneingeschränkte Verteilung von Prozessen mit interaktiven Aktivitäten zu erlauben, wurde in diesem Abschnitt ein Ansatz vorgestellt, welcher eine abstrakte Beschreibung von Benutzungsschnittstellen für die vorgesehene Bearbeitung von Aufgaben in prozessorientierten Anwendungen erlaubt. Wesentlicher Beitrag ist dabei die Übertragung eines bestehenden Ansatzes zur Modellierung von *Concur-Task-Tree*-Modellen auf die Modellierung und Ausführung von Benutzerinteraktionen in prozessorientierten Anwendungen. Basierend auf einem Metamodell zur Beschreibung abstrakter Benutzungsschnittstellen kann dabei zur Entwicklungszeit der prozessorientierten Anwendung mittels graphischer Modellierung eine abstrakte Beschreibung für die im Rahmen der Prozessausführung durchzuführende Interaktion erstellt werden. Die abstrakte Beschreibung wird in individuell festlegbarer Weise mit dem Prozessmodell verknüpft und kann zur Laufzeit – in Abhängigkeit der auf der jeweiligen Ausführungseinheit vorherrschenden Bedingungen – durch eine einfache Vorgehensweise automatisch in eine konkrete Benutzungsschnittstelle transformiert werden. Zur Einbettung des Ansatzes in die bisher in dieser Arbeit verfolgte *Process-Management-as-a-Service*-Strategie wird diese Funktionalität durch einen *Interaktionsdienst* realisiert, welcher durch die lokale Ausführungseinheit individuell gebunden werden kann und auf Basis der abstrakten Beschreibung der zu erfüllenden Aufgabe eine im jeweiligen Anwendungskontext geeignete Schnittstelle für den Benutzer erzeugt.

Im Gegensatz zu den bisher als Beitrag dieser Arbeit vorgestellten Ansätzen kann die Einbindung der abstrakten Benutzungsschnittstelle nicht vollständig ohne die Anpassung der ursprünglichen Prozessbeschreibung geschehen. Eine spontane Verteilung eines (unveränderten) laufenden Prozesses mit bereits statisch definierten konkreten Benutzungsschnittstellen ist daher unter Umständen aufgrund der Notwendigkeit zu vorbereitenden Maßnahmen zur Anpassung der Benutzungsschnittstellen nicht ohne weiteres möglich. Eine Möglichkeit zur Optimierung des hier dargestellten Ansatzes besteht daher in einer möglichst automatischen Transformation von bestehenden Benutzungsschnittstellen in das abstrakte Beschreibungsformat.

6.5 Prognoseverfahren zur proaktiven Anpassung der Verteilung

In den drei vorangegangenen Abschnitten wurden Maßnahmen und Verfahren vorgestellt, um auf der Basis von relevanten Umgebungsänderungen Anpassungen an der verteilten Ausführung prozessorientierter Anwendungen vorzunehmen. Dabei wurde bislang angenommen, dass die Situation, welche die verteilte Ausführung eines Prozesses auslöst oder beeinflusst, zum Betrachtungszeitpunkt bereits eingetreten ist und die Anpassung der Prozessausführung somit eine *Reaktion* auf die eingetretene Situation darstellt. Wie in Abschnitt 3.2 erläutert wurde, stellt aber auch die Möglichkeit zu *proaktivem Verhalten* einen wesentlichen Flexibilitätsaspekt dar. Dabei wird durch offensives Verhalten eine Anpassung ermöglicht, *bevor* eine bestimmte Situation eintritt, oder die Situation wird durch das Verhalten ausgelöst bzw. gänzlich vermieden [GP00, VMSS07].

Eine proaktive Anpassung setzt voraus, dass das Eintreten bzw. Ausbleiben einer in der Zukunft liegenden Situation zum Betrachtungszeitpunkt bekannt ist oder als wahrscheinlich angenommen wird, und dass das Eintreten bzw. Ausbleiben dieser Situation zu einer (notwendigen) Anpassung in einer kausalen Beziehung steht. In Bezug auf prozessorientierte Anwendungen sind die auszuführenden fachlichen Aktivitäten und die Bedingungen, unter denen sie ausgeführt werden sollen, in der Prozessbeschreibung festgeschrieben. Die relevanten Ausführungskontexte, welche für eine (technische) Anpassung der Ausführung von Bedeutung sind, können jedoch von Prozess zu Prozess variieren. Soll zum Beispiel die Situation vermieden werden, dass eine wichtige Ressource für die Ausführung eines Prozesses zur Ausführungszeit einer Aktivität in einer bestimmten Ausführungsumgebung nicht verfügbar ist, so stellt die (zukünftige) Verfügbarkeit dieser Ressource unter Umständen den Auslöser für eine Anpassung der Prozessausführung in Hinblick auf ihre Verteilung dar. Ist die Verfügbarkeit von Ressourcen nicht aus einer verlässlichen Datenquelle abrufbar (z. B. aus einem Dienstplan), so kann zur Ermöglichung einer proaktiven Anpassung prinzipiell die Verfügbarkeit dieser Ressource auf der Basis von historischen Daten prognostiziert werden. Da in unterschiedlichen Prozessen jedoch beliebige unterschiedliche Ressourcen referenziert werden können, stellt die Bereitstellung geeigneter historischer Daten für die Prognose aller möglicher solcher Ausführungskontexte eine große Herausforderung dar. Werden prozessorientierte Anwendungen zudem verteilt ausgeführt, so wird diese Problematik noch verschärft, da auf Fremdsystemen unter Umständen gar kein Wissen über die Ausführungskontexte von (verteilt) auszuführenden Prozessen eines anderen Ursprungssystems vorliegt.

Als Ergänzung zu den drei genannten Verfahren zur Flexibilisierung verteilter Prozessausführung wird in diesem Abschnitt daher ein Verfahren zur Kontextdatenprognose vorgestellt, welches auch eine vorausschauende Verteilung des Prozesses auf Ausführungseinheiten und somit eine proaktive Anpassung der verteilten Prozessausführung ermöglicht. Dabei wird das für dy-

namische Anwendungen entwickelte Rahmenwerk der *strukturierten Kontextdatenprognose* [Mei09, MZL10, ZML11] auf die spezielle Problematik verteilt ausgeführter prozessorientierter Anwendungen angewendet. Im Vordergrund steht dabei die Möglichkeit zur Integration von (grundlegendem) Anwendungswissen, welches zur Laufzeit durch die tatsächlichen Zusammenhänge im Ausführungskontext ergänzt werden kann, so dass auch in einem verteilten Szenario Prognosen mit anwendungsspezifischer Genauigkeit und Effizienz (insbesondere in Hinblick auf den Ressourcenverbrauch) vorgenommen werden können. Die Problematik, die prinzipielle Vorgehensweise und die Integration von Kontextdatenprognosen für verteilt auszuführende Prozesse werden im folgenden Abschnitt genauer erläutert (vgl. Abschnitt 6.5.1). Eine Methodik zur Einbindung von Anwendungswissen für die Erstellung von Prognosemodellen über beliebige Kontexte ist Inhalt von Abschnitt 6.5.2. In Abschnitt 6.5.3 wird die Durchführung von Kontextdatenprognosen zur Laufzeit der beteiligten Ausführungssysteme im Allgemeinen vorgestellt. Abschnitt 6.5.4 zeigt auf Basis der strukturierten Kontextdatenprognose mögliche Strategien zur Anpassung von Prognosen an dynamische Kontexte und zur weitestgehenden Unterstützung von Ad-hoc-Prognosen auf. Als Beispiel für die Verwendung von strukturierten Kontextdatenprognosen für verteilt ausgeführte Prozesse wird in Abschnitt 6.5.5 die Vorhersage von Dienstverfügbarkeiten für mobile Ausführungseinheiten dargestellt. Der Beitrag zur Kontextdatenprognose für die proaktive Anpassung der Verteilung schließt mit einer Zusammenfassung und der Einordnung der Ergebnisse in den Gesamtkontext dieser Arbeit (vgl. Abschnitt 6.5.6).

6.5.1 Prinzipien und Entwicklungsgrundsätze

In Abschnitt 3.5.6 und in den Ansätzen der Abschnitte 5.4, 5.5 und 5.6 wurden verschiedene Möglichkeiten für die Vorhersage von Kontextdaten im Allgemeinen sowie für die Partitionierung von Prozessen und die Zuordnung von Prozesspartitionen zu Ausführungseinheiten untersucht. Auf Basis der dort gewonnenen Erkenntnisse wird in diesem Abschnitt eine Vorgehensweise motiviert, welche es erlaubt, die Verteilung prozessorientierter Anwendungen auch proaktiv auf erwartete Umgebungsänderungen anzupassen.

Vorüberlegung

Eine proaktive Anpassung der Verteilung prozessorientierter Anwendungen kann prinzipiell auf der Basis des gegenwärtigen Zustands und/oder auf Basis von Erwartungen an zukünftige Zustände vorgenommen werden. In beiden Fällen können zwei wesentliche, bei der proaktiven Anpassung zu berücksichtigende variable Faktoren identifiziert werden:

- **Ausführungspfad der Prozessinstanz:** Die im Kontext dieser Arbeit betrachteten Prozessmodelle verfügen in der Regel über einen fest vorgegebenen Kontrollfluss, welcher jedoch durch die Möglichkeit zur Ver-

zweigung, zur Behandlung von Ereignissen oder durch die Kompensation von Aktivitäten alternative Ausführungspfade auf der Ebene individueller Prozessinstanzen erlaubt (vgl. Abschnitt 2.4). Da die Entscheidung für einen bestimmten Ausführungspfad durch Variablenwerte und Ereignisse bestimmt werden kann, ist eine vollständige Ableitung des tatsächlichen Prozessverlaufs einzelner Prozessinstanzen im Voraus in der Regel nicht möglich. Eine Vorhersage des tatsächlichen Ausführungspfades einer Prozessinstanz kann jedoch durch die Erhebung von Erfahrungswerten für bereits ausgeführte Instanzen des dazugehörigen Prozessmodells und die entsprechende Bildung von Wahrscheinlichkeiten für die Ausführung eines bestimmten Teilpfades vorgenommen werden. Da Verzweigungen jedoch mehrfach und auch verschachtelt auftreten können, ist dabei in der Regel eine hohe Unsicherheit gegeben.

Eine Ausnahme stellen Prozessmodelle mit streng sequentiellem Kontrollfluss dar (d.h. ohne Möglichkeiten zur Verzweigung, zur Behandlung von Ereignissen oder zur Kompensation von Aktivitäten). Hierbei ist eine Vorausbestimmung des gesamten Ausführungspfades möglich, sofern während der Ausführung keine Fehler auftreten, welche einen Abbruch des Prozesses zur Folge haben.

- **Ausführungskontext der Prozessinstanz:** Der Ausführungskontext umfasst die Eigenschaften der Ausführungsumgebung, welche für die vorgesehene funktionale und/oder nicht-funktionale Ausführung einer Prozessinstanz relevant sind. In dynamischen Umgebungen kann sich der Ausführungskontext während der Ausführung eines Prozesses ändern. Typische Beispiele sind mobile Ausführungseinheiten, deren geographische Position sich zur Laufzeit eines Prozesses ändern kann, oder Ressourcen, welche während der Laufzeit des Prozesses unterschiedliche Verfügbarkeiten aufweisen können. Eine Vorhersage des Ausführungskontextes ist prinzipiell durch die Erhebung von Erfahrungswerten zu den jeweils relevanten Objekten der Ausführungsumgebung möglich. Dabei ist der relevante Ausführungskontext jedoch durch die Prozessmodelle der auszuführenden Prozesse bzw. deren nicht-funktionale Rahmenbedingungen und darüber hinaus durch den konkreten Ausführungspfad einzelner Prozessinstanzen (siehe oben) determiniert.

Wenn der auszuführende Prozess über einen streng sequentiellen Kontrollfluss verfügt und der hierfür relevante Ausführungskontext während der Ausführung des Prozesses stabil bleibt, sind alle zukünftigen Zustände der Prozessinstanz in einer vorgegebenen Ordnung determiniert und der gegenwärtige Zustand des Ausführungskontextes ist ausreichend, um eine gezielte (proaktive) Anpassung der Verteilung zu ermöglichen. In diesem Fall können die für die Prozessausführung optimalen Ausführungseinheiten aufgrund ihrer aktuellen Eigenschaften identifiziert, entlang des festgelegten

Ausführungspfades reserviert und im Voraus an die aus dieser Verteilung resultierenden Prozesspartitionen gebunden werden (vgl. [Gol09] zu möglichen Ansätzen).

Ist der Ausführungspfad des Prozesses variabel und der Ausführungskontext statisch, so ist ebenfalls der gegenwärtige Zustand des Ausführungskontextes ausreichend, um eine proaktive Anpassung der Verteilung zu ermöglichen. Im Gegensatz zum zuvor genannten Fall würden jedoch Ausführungseinheiten reserviert bzw. gebunden werden, die im tatsächlichen Prozessverlauf unter Umständen gar nicht benötigt werden [Gol09]. Diese Problematik kann durch die Integration der oben genannten Erfahrungswerte für Kontrollflussentscheidungen relativiert werden. Eine mögliche Vorgehensweise hierfür ist es, innerhalb des Prozessmodells jede Verzweigung in alternative Kontrollflusspfade mit Wahrscheinlichkeiten zu versehen, nur Ausführungseinheiten zu binden bzw. zu reservieren, welche auf einem Pfad mit einer gewissen Ausführungswahrscheinlichkeit zum Einsatz kommen würden, und ansonsten auf eine reaktive Strategie zurückzufallen. Die für die Bildung dieser Wahrscheinlichkeiten benötigten Erfahrungswerte können dabei in der Evaluierungsphase des klassischen Lebenszyklus von (nicht-verteilten) prozessorientierten Anwendungen gewonnen werden (vgl. [GCC+04]).

Der für diese Arbeit interessante Fall tritt ein, wenn der für eine Prozessinstanz relevante Ausführungskontext (unter der Annahme nachvollziehbarer Zusammenhänge) dynamisch ist. In diesem Fall können zwei mögliche Strategien für eine proaktive Anpassung identifiziert werden:

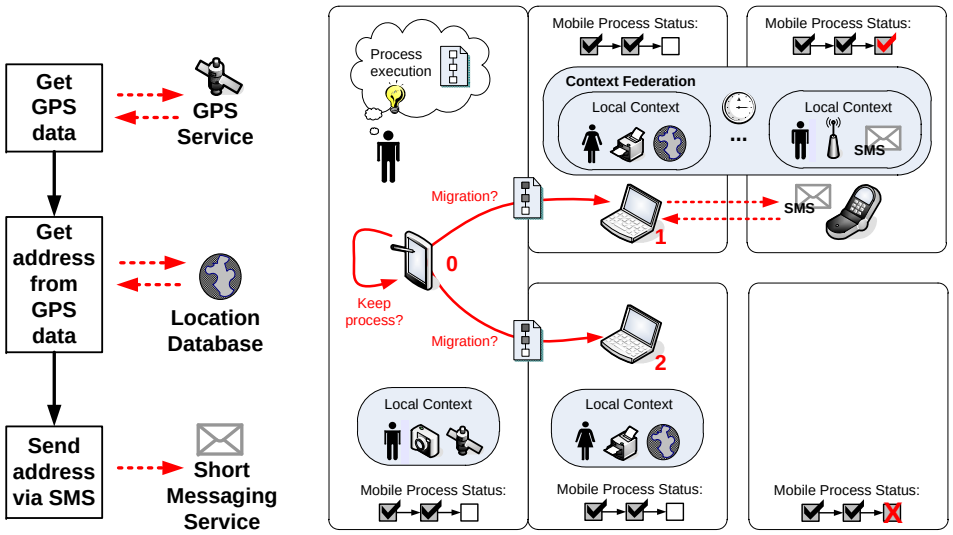
- **Verteilung auf Basis des gegenwärtigen Zustands:** Ein Abgleich der auszuführenden Aktivitäten eines Prozesses und der hierfür benötigten Ausführungskontexte kann prinzipiell zu jedem Zeitpunkt der Prozessausführung vorgenommen werden. Steht nur der gegenwärtige Zustand eines dynamischen Ausführungskontextes zur Verfügung, so kann die Bildung von Prozesspartitionen und deren Zuordnung zu Ausführungseinheiten auf Basis der zur Verfügung stehenden Informationen vorausberechnet werden. Tritt jedoch eine Änderung des relevanten Ausführungskontextes ein, so muss die (verteilte) Ausführungsfolge unter Umständen jedes Mal neu berechnet werden [ZKL09a]. In Abhängigkeit der gewählten Berechnungsstrategie, der Anzahl der Aktivitäten im Prozessmodell, der möglichen Verzweigungen des Kontrollflusses und der Anzahl der für eine Zuweisung in Frage kommenden Ausführungseinheiten kann diese Vorausberechnung eine hohe Komplexität annehmen und somit dem Ziel einer effizienten Ausführung entgegen stehen [Gol09]. In Abhängigkeit der Dynamik des Ausführungskontextes und der für die Berechnung zur Laufzeit zur Verfügung stehenden Ressourcen ist daher unter Umständen nur eine kurzfristige Vorausberechnung der Verteilung möglich bzw. sinnvoll (z. B.

für die nächsten drei Aktivitäten oder bis zur nächsten Verzweigung des Kontrollflusses) [ZKL09a].

- **Verteilung auf Basis von Prognosen über zukünftige Kontexte:** Stehen ergänzend zum gegenwärtigen Zustand Prognosen über die zukünftige Entwicklung des Ausführungskontextes zur Verfügung, so können diese Prognosen entweder in die Vorausberechnung der vorzunehmenden Verteilung einbezogen werden und im Idealfall (d. h. bei Eintreten der vorausgesagten Situation) helfen, die Anzahl der nötigen Neuberechnungen der (verteilten) Ausführungsfolge verringern. Auf der anderen Seite stehen die Prognosen jedoch auch zur Verfügung, um spontane (nicht geplante) Entscheidungen über die Verteilung zu unterstützen. Im Kontext flexibel verteilt ausgeführter Prozesse ist genau diese Art der Unterstützung besonders interessant, da hierdurch die Qualität der spontan durchgeführten Verteilungen erhöht werden kann.

Im Gegensatz zu monolithischen Anwendungen, welche für ein proaktives Verhalten Zukunftsprognosen über einen statischen Anwendungsbereich verwenden können, sind für die (verteilte) Ausführung von Prozessen jeweils unterschiedliche Ausführungskontexte relevant. Während der Ausführungspfad individueller Prozessinstanzen anhand der mehrfachen Ausführung eines Prozesses auf Basis des Prozessmodells und der oben genannten Erfahrungswerte mehr oder weniger gut mit einer einheitlichen Methode vorhergesagt werden kann (z. B. durch die Bildung eines Mittelwerts über die oben genannten Häufigkeiten), ist eine Prognose über generische Kontextdaten, welche die Verteilung eines Prozesses beeinflussen können, eine große Herausforderung. In Abschnitt 3.5.6 wurde anhand bestehender Ansätze gezeigt, dass ein für generische Kontexte einsetzbares Prognoseverfahren oft große Einschränkungen bei der Genauigkeit der Prognoseergebnisse oder hinsichtlich des Verbrauchs an Speicher oder Rechenleistung mit sich bringt. Diese Problematik wird bei einer verteilten Ausführung von Prozessen noch verschärft, da ein dauerhaftes Sammeln von Daten in einem festgelegten Ausführungsbereich und ein „Erlernen“ von Zusammenhängen zur Erkennung relevanter Situationen nicht ohne weiteres möglich ist. Dies soll im Folgenden an einem Beispiel aus dem Anwendungsbereich der kontextbasierten Kooperation (vgl. Abschnitt 4.2.4) verdeutlicht werden (vgl. Abbildung 6.48).

Abbildung 6.48a zeigt eine einfache prozessorientierte Anwendung, um es einem mobilen Benutzer zu ermöglichen, seine geographische Position zu erheben (*Get GPS data*), diese in eine Adresse aus Straße, Hausnummer und Ort umzuwandeln (*Get address from GPS data*) und das Ergebnis per Kurznachrichtendienst an eine vordefinierte Telefonnummer zu übermitteln (*Send address via SMS*) [ZML11]. Nach dem Konzept der kontextbasierten Kooperation kann diese als Prozess strukturierte Funktionalität durch eine verteilte Ausführung des Prozesses erfüllt werden, wenn der Prozess in Abhängigkeit des aktuellen Ausführungskontextes an (mobile) Ausführungseinheiten mit den benötigten Fähigkeiten delegiert wird. In diesem Beispiel soll angenom-



(a) Beispielprozess **(b)** Kontextdatenprognose zur Unterstützung der Prozessmigration

Abbildung 6.48: Prognosen für kontextbasierte Kooperation (Beispiel) (ZML11)

men werden, dass die ersten beiden Aktivitäten des dargestellten Prozesses bereits erfolgreich ausgeführt werden konnten, die Kurznachricht jedoch zum Betrachtungszeitpunkt nicht gesendet werden kann, da das aktuell für die Ausführung verantwortliche Gerät keinen Zugang zu einem entsprechenden Kurznachrichtendienst besitzt. Zudem ist im direkten Ausführungskontext kein anderes Gerät mit einem geeigneten Kontext (d. h. mit Zugang zu einem Kurznachrichtendienst) vorhanden, um die Prozessausführung fortzusetzen. Die aus dieser Situation entstehenden Möglichkeiten für die weitere Ausführung des Prozesses sind in Abbildung 6.48b dargestellt: Erstens könnte der Prozess auf der aktuellen Ausführungseinheit (Gerät 0) verbleiben, wo er jedoch nicht ausgeführt werden kann, wenn keine geeignete Änderung des Ausführungskontextes eintritt. Zweitens könnte der Prozess an eine andere Ausführungseinheit (Gerät 1) übergeben werden, welche zwar zur Zeit ebenfalls einen für die Ausführung der dritten Aktivität „nutzlosen“ Kontext besitzt, aber in der nahen Zukunft aufgrund seiner Mobilität mit großer Wahrscheinlichkeit auf ein kurznachrichtenfähiges Gerät treffen wird. Drittens könnte der Prozess an eine andere (hinsichtlich des aktuellen Kontextes gleichwertige) Ausführungseinheit (Gerät 2) übergeben werden, welche jedoch mit großer Wahrscheinlichkeit *nicht* in naher Zukunft auf ein kurznachrichtenfähiges Gerät treffen wird. Es ist offensichtlich, dass nur die Migration des Prozesses auf Gerät 1 eine erfolgreiche Ausführung des Prozesses verspricht und die Ausführung in den beiden anderen Fällen wahrscheinlich stark verzögert oder sogar fehlschlagen wird [ZML11]. Das Beispiel verdeutlicht somit, dass in dynamischen Anwendungsbereichen die Fähigkeit zur Kontextdatenprognose nicht nur zur Optimierung nicht-funktionaler Aspekte eingesetzt

werden kann, sondern im Einzelfall auch über die erfolgreiche Ausführung eines verteilten Prozessen entscheiden kann.

Die Prozessausführung selbst und die (potentielle) Verteilung der Prozessausführung machen eine Vorhersage der benötigten Daten dabei schwierig: Eine Ausführungseinheit kann zwar im Allgemeinen für *prozessunabhängige* Kontextdaten (wie zum Beispiel die Auslastung der Prozess-Engine) Prognosen bereitstellen, da die hierfür benötigten Zusammenhänge und Anforderungen bereits zur Entwicklungszeit der Ausführungsumgebung festgelegt werden können. Prognosen über *prozessabhängige* Kontextdaten wie die oben genannte Verfügbarkeit eines speziellen Dienstes für die Ausführung einer einzelnen Aufgabe in einem individuellen Prozessmodell können jedoch nicht zur Entwicklungszeit der Ausführungsumgebung integriert werden, da unter Umständen für jede Prozessausführung andere Kontextdaten relevant sind. Die hierfür benötigten anwendungsspezifischen Zusammenhänge müssen daher zur Laufzeit hinzugefügt und dynamisch verarbeitet werden können, so dass nach einer möglichst kurzen Zeit der Datenerhebung möglichst aussagekräftige Prognosen abgefragt werden können. Bei einer verteilten Ausführung ist es zudem notwendig, Informationen verschiedener Ausführungseinheiten für die Bereitstellung einer Prognose zu integrieren. In dem oben genannten Anwendungsfall ist es zum Beispiel nicht möglich, das potentielle Zielausführungssystem einfach nach einer Prognose über die Verfügbarkeit eines Kurznachrichtendienstes zu fragen, da Kurznachrichtendienste für die dort bislang ausgeführten Prozesse unter Umständen noch nie eine Rolle gespielt haben, somit nicht zum *relevanten Kontext* dieser Ausführungseinheit gehören und keine Erfahrungswerte zu einer Verfügbarkeit solcher Dienste als Basis für eine aussagekräftige Prognose bestehen. Ein für die verteilte Ausführung prozessorientierter Anwendungen geeignetes Prognosesystem muss daher Möglichkeiten für die (möglichst dynamische) Integration von Anwendungswissen besitzen, muss in der Lage sein, die individuellen, regelmäßig wiederkehrenden Eigenschaften von Ausführungssystemen zu erfassen und muss beides zur Laufzeit miteinander in Verbindung bringen [ZML11]. Im Folgenden werden daher die grundlegenden Prinzipien einer Entwicklungsmethodik für Prognosen im Kontext verteilt ausgeführter Prozesse vorgestellt, welche die in diesem Abschnitt erarbeiteten Vorüberlegungen aufgreifen.

Entwicklungsmethodik

Um die Prognose beliebiger Kontextdaten mit den oben genannten Anforderungen durch ein geeignetes Rahmenwerk zu unterstützen, wurde das Konzept der *strukturierten Kontextdatenprognose* (*Structured Context Prediction*, kurz *SCP*) [Mei09, MZL10] erarbeitet und auf den Problembereich der dynamisch verteilt ausgeführten Prozesse übertragen [ZML11]. Abgeleitet von einer Analyse bestehender Ansätze stützt sich die strukturierte Kontextdatenprognose auf zwei wichtige Prinzipien [MZL10, ZML11]:

- Um den Aufwand für das Erfassen von Zusammenhängen zur Laufzeit möglichst gering zu halten und die Qualität späterer Prognosen zu erhöhen, besteht die Möglichkeit, zur Entwicklungs- und/oder zur Laufzeit abstraktes und/oder konkretes Wissen über die Anwendungsdomäne zu spezifizieren. Sofern bekannt ist, welche Variablen ein Eintreten oder Ausbleiben einer für den Anwendungskontext relevanten Situation beeinflussen, können diese beschrieben, zur Laufzeit ergänzt und als Ausgangsbasis von Prognosen verwendet werden. Das Wissen über die abstrakten Zusammenhänge einer Anwendungsdomäne wird dabei in Form von sogenannten *Prognosenetzen* festgehalten.
- Auf Basis des vorliegenden Anwendungswissens kann die Genauigkeit und Effizienz späterer Prognosen durch die individuelle Auswahl und hybride Anwendung von verschiedenen Prognosemethoden hinsichtlich der Anforderungen des jeweiligen Anwendungsfalls abgestimmt werden. Das (abstrakte) Wissen über die Zusammenhänge zwischen verschiedenen Variablen einer Anwendungsdomäne (Prognosenetz) und die Zuordnung von Prognosemethoden wird dabei in Form von sogenannten *Prognosemodellen* miteinander in Verbindung gebracht.

Im Kontext von (verteilt ausgeführten) prozessorientierten Anwendungen können Prognosemodelle für allgemeine (prozessmodellunabhängige) Anwendungskontexte (wie z. B. die Auslastung oder die geographische Position der Prozess-Engine) oder für spezielle (prozessmodellabhängige) Anwendungskontexte (wie z. B. die Verfügbarkeit einer speziellen Ressource) erstellt werden. Beide Arten von Prognosemodellen werden in der Regel zur Entwicklungszeit der Ausführungsumgebung bzw. des Prozessmodells erzeugt, können jedoch prinzipiell auch zur Laufzeit interpretiert, ausgetauscht oder erweitert werden. Da zielführende Prognosen jedoch nur auf Basis einer ausreichenden Menge an gesammelten Trainingsdaten erstellt werden können, ist das spontane Einbringen von Prognosemodellen zur Laufzeit einzelner Prozessinstanzen natürlich in seiner Anwendbarkeit begrenzt.

Abbildung 6.49 zeigt eine entsprechende Vorgehensweise zur Integration von Prognosemodellen und zur Erstellung von Prognosen für (verteilt ausgeführte) prozessorientierte Anwendungen. Da für eine proaktive Anpassung im Kontext dieser Arbeit die Möglichkeit zur dynamischen Verteilung notwendig ist, wird die Kontextdatenprognose im Folgenden mit dem in Abschnitt 6.2 vorgestellten Verfahren zur Migration von laufenden Prozessinstanzen integriert betrachtet. Dabei erfolgt wie schon in den vorhergehenden Ansätzen zur Flexibilisierung eine strenge Entkopplung zwischen dem fachlichen Prozess und den für die Anpassung relevanten Maßnahmen. Die durchzuführenden Maßnahmen zur Kontextdatenprognose werden in dieser Arbeit dem Kontextmanagement zugeordnet, welches auch schon für die Erhebung von Daten für die Anpassung prozessorientierter Anwendungen im Allgemeinen zuständig ist.

Für die Erstellung von Prognosemodellen wird bei Bedarf Anwendungswissen aus dem Kontext des Prozessmanagements im Allgemeinen (AK) oder aus

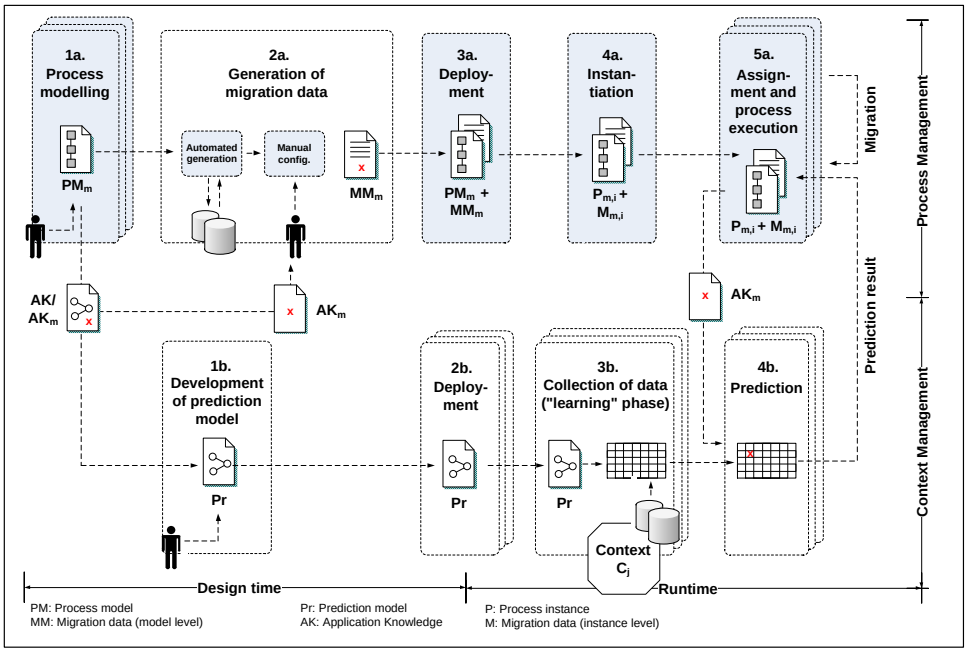


Abbildung 6.49: Integration der strukturierten Kontextdatenprognose zur proaktiven Anpassung der Verteilung von prozessorientierten Anwendungen

einzelnen oder mehreren Prozessmodellen im Speziellen (AK_m) herausgezogen und kann auf diese Weise in die Erstellung eines abstrakten Prognosemodells einfließen. Dieser Schritt (Schritt 1b in Abbildung 6.49) erfordert eine manuelle Modellierung durch den Anwendungsentwickler oder eine andere Person mit entsprechendem Anwendungswissen. Das erstellte Prognosemodell kann nun im Rahmen des Kontextmanagements durch ein Prognosesystem interpretiert werden. Dazu muss es dem ausgewählten Ausführungssystem bzw. den ausgewählten Ausführungssystemen übergeben werden (Schritt 2b *Deployment*). Das Kontextmanagementsystem der betrachteten Ausführungseinheit beginnt dementsprechend mit der Erhebung von Trainingsdaten, um auftretende Zusammenhänge im Kontext des Prognosemodells zu erkennen. Hierbei kann zum Beispiel *adaptives Online-Lernen* eingesetzt werden (vgl. [Mei09, MZL10]). Schritt 3b in Abbildung 6.49 zeigt das „Lernen“ von Zusammenhängen, indem in festgelegten Zeitabständen die Werte von abhängigen und unabhängigen Variablen erhoben und geeignet verdichtet werden (*Trainingsdaten*). Auf Basis der gesammelten und ggf. verdichteten Daten können schließlich Prognosen zur Laufzeit einzelner Prozessinstanzen erstellt werden, um die Entscheidung über deren dynamische Verteilung zu unterstützen. Schritt 4b in Abbildung 6.49 veranschaulicht die Prognose eines Wertes beispielhaft anhand der Prognosemethode einfacher Wahrscheinlichkeitstabellen.

Eine Verteilung von Prognosemodellen an alle potentiell beteiligten Ausführungseinheiten bzw. die Erhebung von Trainingsdaten durch alle potentiell beteiligten Geräte ist (abhängig von den Prognosemodellen und der Anzahl potentieller Ausführungseinheiten) mit relativ großem Aufwand ver-

bunden. Ein Austausch von Prognosemodellen zum Zeitpunkt der Prognose ist zwar prinzipiell möglich, würde aber für die aktuelle Prognose keinen Mehrwert erbringen, da in diesem Fall zu dem angefragten Prognoseobjekt auf der potentiellen Zielausführungseinheit noch keine individuellen Trainingsdaten bestehen (vgl. die Erläuterungen zum Beispiel in Abbildung 6.49). In dieser Arbeit wird daher eine Kompromisslösung vorgeschlagen: Abstraktes Anwendungswissen AK wird als Prognosemodell Pr modelliert und auf verschiedenen Ausführungssystemen als Grundlage für Prognosen für spezielle Anwendungskontexte verwendet. Das für die Ableitung spezieller Prognosen benötigte Anwendungswissen AK_m wird mit dem (verteilt ausgeführten) Prozess migriert und steht dadurch zum Zeitpunkt der Prognose zur Integration mit gelernten Zusammenhängen einzelner Ausführungseinheiten zur Verfügung. Im oben genannten Beispiel (vgl. Abbildung 6.49) kann z. B. als abstraktes Anwendungswissen AK modelliert werden, dass Dienstverfügbarkeiten im Allgemeinen mit der Verfügbarkeit von (gewissen Arten von) Netzwerken in Verbindung stehen. Als konkretes Anwendungswissen AK_m kann in Rahmen der Verteilungsinformationen zum fachlichen Prozess (z. B. hier im Rahmen der Migrationsdaten) mitgeliefert werden, dass die Wahrscheinlichkeit für die Verfügbarkeit von SMS-Diensten hoch ist, wenn Verbindung zu einem UMTS- oder GSM-Netzwerk besteht. Eine (mobile) Ausführungsumgebung, welche ausreichend Daten über ihre Netzwerkverbindungen gesammelt hat, kann somit eine Aussage über die Verfügbarkeit von SMS-Diensten machen, obwohl SMS-Dienste bisher nicht zum relevanten Kontext dieser Ausführungsumgebung gehörten und daher zu diesem speziellen Prognoseobjekt bisher keine Erfahrungswerte vorliegen.

Konkretes Anwendungswissen zu einem bestimmten Prozessmodell oder sogar zu einer bestimmten Prozessinstanz kann bei einer manuellen Anpassung der Migrationsdaten integriert oder bei Bedarf sogar zur Laufzeit des Prozesses hinzugefügt werden. Es muss stets einen Zusammenhang zum Prognosemodell zulassen und diesen in geeigneter Art und Weise spezifizieren. Im Folgenden werden die Erstellung von Prognosemodellen (Schritt 1b in Abbildung 6.49) und die Vorhersage von Kontexten (Schritte 2b bis 5b in Abbildung 6.49) auf Basis der strukturierten Kontextdatenprognose genauer erläutert.

6.5.2 Erstellung von Prognosemodellen

Bei der strukturierten Kontextdatenprognose handelt es sich um ein Rahmenwerk zur Prognose von Anwendungskontexten. Dabei wird das für etwaige Prognosen relevante Wissen über eine bestimmte Anwendungsdomäne in Form eines *Prognosemodells* repräsentiert. Ein Prognosemodell gibt an, wie eine Prognose über ein betrachtetes Prognoseobjekt durchgeführt werden soll, um die im Anwendungskontext benötigte Genauigkeit des Prognoseergebnisses und den dabei hinzunehmenden Ressourcenverbrauch zu erreichen bzw. nicht zu überschreiten [MZL10]. Dazu werden bekannte Abhängigkeiten zwischen den für die Prognose relevanten Variablen spezifiziert. Jeder (abhängigen) Varia-

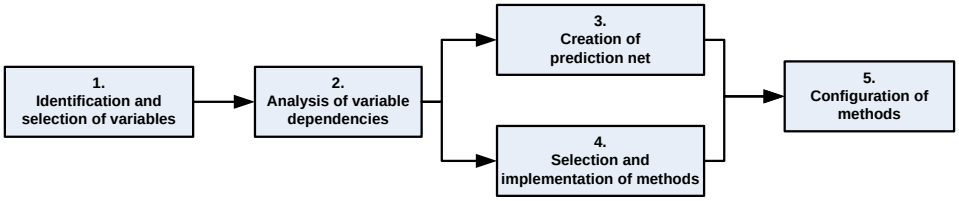


Abbildung 6.50: Allgemeine Vorgehensweise zur Erstellung von Prognosemodellen (Mei09, MZL10, ZML11)

blen wird eine konkrete *Prognosemethode* zugewiesen, welche für die Vorhersage ihres Variablenwertes genutzt werden soll. Die einzelne Prognosemethode verwendet wiederum die Werte anderer Variablen als Eingabeparameter, welche ihrerseits vorhergesagt werden oder welche bekannt sind (also z. B. durch einen Sensor erhoben oder durch Kommunikation mit anderen Ausführungseinheiten in Erfahrung gebracht werden können) [MZL10, ZML11].

Die Erstellung von Prognosemodellen kann dementsprechend grob in fünf Schritte gegliedert werden [MZL10, ZML11] (vgl. Abbildung 6.50): Der Anwendungsentwickler bzw. der Prozessmodellierer identifiziert zunächst die im Anwendungskontext relevanten Variablen (Schritt 1) und untersucht diese hinsichtlich ihrer Abhängigkeiten untereinander (Schritt 2). Die Abhängigkeiten der Variablen werden in einer graphischen Struktur, dem sogenannten *Prognosenetz*, festgehalten (Schritt 3). Parallel dazu werden geeignete Prognosemethoden an die Variablen zugewiesen (Schritt 4). Im letzten Schritt werden die Prognosemethoden individuell konfiguriert (Schritt 5).

Die Schritte 3, 4 und 5 werden durch die strukturierte Kontextdatenprognose unterstützt. Die wichtigsten Bestandteile des Ansatzes werden im Folgenden kurz hervorgehoben.

Prognosenetze

Ein *Prognosenetz* enthält die zur Entwicklungszeit bekannten Zusammenhänge zwischen den Variablen, welche für eine Prognose zu berücksichtigen sind. Dazu gibt es durch eine Strukturierung der Variablenbeziehungen für jede relevante Variable an, wie der Wert einer Variablen zu einem bestimmten Zeitpunkt auf der Basis der Werte einer oder mehrerer anderer Variablen zu demselben Zeitpunkt und/oder zu früheren Zeitpunkten vorhergesagt werden kann. Ein stark vereinfachtes Beispiel ist die Prognose des geographischen Aufenthaltsortes einer Person. Dabei kann zum Beispiel eine Annahme über den Aufenthaltsort der Person in naher Zukunft (z. B. in einer Stunde) auf Basis ihrer derzeitigen Position und einer ausreichenden Menge an vorhergehenden Positionen bestimmt werden [Mei09, MZL10, ZML11].

Ein Prognosenetz ist formal als endlicher gerichteter Graph $\mathcal{N} = (W, E)$ definiert. Die Knotenmenge W repräsentiert dabei eine Menge von Varia-

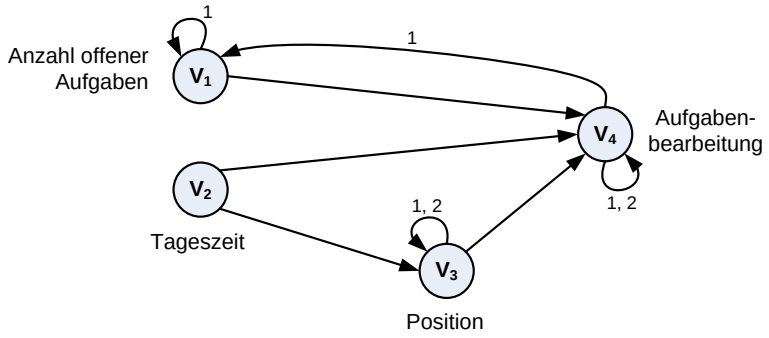


Abbildung 6.51: Beispiel eines einfachen Prognosenetzes (in Anlehnung an (Mei09, MZL10))

blen $\{V_1, ..., V_n\}$ und eine Kante $V_i \overset{\Delta}{\rightarrow} V_{i'} := (V_i, \Delta, V_{i'})$ der Kantenmenge $E \subseteq W \times \mathbb{N}_0 \times W$ drückt aus, das der Wert von V_i zum Zeitpunkt $j - \Delta$ für eine Prognose des Wertes von $V_{i'}$ zum Zeitpunkt j zu berücksichtigen ist, wobei Δ einen Zeitabstand angibt. Für eine bessere Lesbarkeit ist die Notation $V_i \rightarrow V_{i'} := V_i \overset{0}{\rightarrow} V_{i'}$ und $V_i \overset{1,...,l}{\longrightarrow} V_{i'} := \bigcup_{k=1}^{l} V_i \overset{k}{\rightarrow} V_{i'}$ als Abkürzung definiert. Ein Prognosenetz darf keine Zyklen der Form $V_i \overset{\Delta_1}{\rightarrow} ... \overset{\Delta_l}{\rightarrow} V_i$ mit $\sum_{k=1}^{l} \Delta_k = 0$ enthalten [Mei09, MZL10, ZML11].

Ein Beispiel für ein einfaches Prognosenetz ist in Abbildung 6.51 gezeigt. Der fachliche Hintergrund dieses Beispiels ist die Prognose des Verhaltens eines menschlichen Prozessteilnehmers in Bezug auf die Bearbeitung von Aufgaben in seiner Aufgabenliste (boolesche Variable V_4). Dabei wird unterstellt, dass das Bearbeitungsverhalten von der Tageszeit (Variable V_2, diskrete Zeitintervalle, z. B. 9:00-17:00 und 17:00-9:00 Uhr), der Position (Variable V_3, diskrete Orte, z. B. „Büro“, „zu Hause“ und „unterwegs“) und der Menge an Aufgaben in der Aufgabenliste des Prozessteilnehmers (Variable V_1, positive Ganzzahlen) abhängig ist. Die Beziehung $V_i \rightarrow V_{i'}$ besagt dabei, dass der Wert der Variablen $V_{i'}$ zu einem bestimmten Zeitpunkt vom Wert der Variablen V_i zum selben Zeitpunkt abhängig ist. Diese Beziehung gilt zum Beispiel für den einfachen Zusammenhang zwischen Tageszeit und Position. Die Beziehung $V_i \overset{1,...,l}{\longrightarrow} V_{i'}$ zeigt entsprechend an, dass der Wert der Variablen $V_{i'}$ zu einem bestimmten Zeitpunkt von den Werten der Variablen V_i zu $1, ..., l$ Zeitabständen früher abhängig ist. Diese Beziehung gilt im dargestellten Beispiel unter anderem für die Position [Mei09, MZL10]).

Prognosenetze sind von *Bayeschen Netzen* [DGH92] inspiriert, da diese explizit das Einbeziehen des Faktors Zeit erlauben. Während Bayesche Netze jedoch im Allgemeinen Abhängigkeiten zwischen Variablen beschreiben, setzen Prognosenetze die den Variablen zugewiesenen Prognosemethoden miteinander in Verbindung. Sie stellen daher nicht selbst eine Prognosemethode dar, sondern dienen als Repräsentation, um andere (bestehende oder neue) Prognosemethoden zielgerichtet miteinander zu integrieren. Ein Beispiel ist die Verwendung von Wahrscheinlichkeitstabellen zur Prognose von Werten einer Variablen auf Basis einer Nominalskala und die parallele Verwendung von Regressionsverfahren zur Prognose von Werten einer anderen Variablen auf Basis einer Ra-

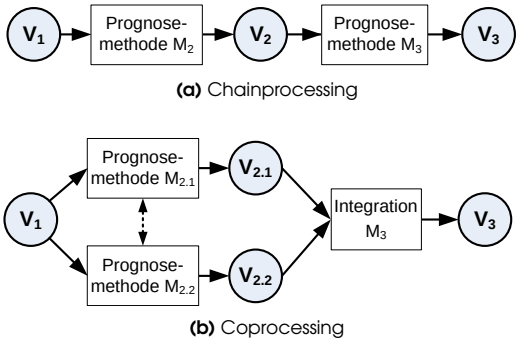


Abbildung 6.52: Hybride Anwendung von Prognosemethoden (Beispiele) (MZL10, ZML11)

tioskala, um eine lineare Abhängigkeit abbilden zu können ohne dabei einen besonders großen Speicherbedarf zu erfordern [MZL10, ZML11]. Die Integration solcher Prognosemethoden entlang der Kanten des Prognosenetzes wird im Folgenden genauer vorgestellt.

Integration von Prognosemethoden

Prognosenetze sind im Verfahren der strukturierten Kontextdatenprognose ein wichtiges Hilfsmittel für den Anwendungsentwickler, um die Beziehungen zwischen Variablen unter Berücksichtigung verschiedener Zeitpunkte zu spezifizieren. Auf der Basis dieser grundsätzlichen Beziehungen können parallel geeignete Prognosemethoden zugewiesen werden, um für jede Variable eine möglichst genaue und effiziente Prognose ihres Wertes zu ermöglichen. Dabei erlaubt die strukturierte Kontextdatenprognose auch einen hybriden Einsatz von mehreren unterschiedlichen Prognosemethoden.

Abbildung 6.52a stellt eine einfache graphische Repräsentation eines Prognosemodells dar, welches durch sogenanntes *Chainprocessing* [Hil95] die sequentielle Berechnung zweier Variablen erlaubt. In dem dargestellten Fall wird basierend auf dem Wert bzw. den Werten von V_1 die Ausgabe der ersten Prognosemethode M_2 in der Variablen V_2 gespeichert, welche als Eingabe für die zweite Prognosemethode M_3 zur Berechnung von V_3 dient. Diese Vorgehensweise kann zum Beispiel für die Bildung von Clustern, zur Generalisierung oder zur Vorverarbeitung von Variablen als Vorbereitung für die nachfolgende Anwendung anderer Prognosemethoden genutzt werden. Zudem erlaubt diese Vorgehensweise, Zwischenergebnisse in zusätzlichen Variablen festzuhalten, welche nun wiederum auch für andere Prognosen in Kombination mit anderen Variablen wiederverwendet werden können. Eine redundante Speicherung und die unnötige Wiederholung von Verarbeitungsschritten innerhalb der Prognose kann somit reduziert bzw. vermieden werden [MZL10, ZML11]. Ein Beispiel ist eine mögliche Weiterverarbeitung der Position (Variable V_3) in Abbildung 6.51 zur Prognose anderer Anwendungskontexte, wie zum Beispiel der Verfügbarkeit von Netzwerken (vgl. [Mei09]).

Abbildung 6.52b zeigt entsprechend die Berechnung einer Variablen durch *Coprocessing* [Hil95]. Die Werte der Variablen $V_{2.1}$ und $V_{2.2}$ werden hierbei

durch jeweils unterschiedliche Prognosemethoden $M_{2.1}$ und $M_{2.2}$ berechnet, welche kooperativ oder kompetitiv arbeiten und dabei nebenläufig bzw. parallel ausgeführt werden können. Das Ergebnis wird schließlich durch eine geeignete Methode in der Variablen V_3 integriert. Hierfür kann zum Beispiel die Bildung eines Mittelwertes oder ein Mehrheitsentscheid eingesetzt werden [MZL10, ZML11]. Durch den Einsatz von Coprocessing kann das Prognoseergebnis hinsichtlich der Genauigkeit verbessert werden, wobei insbesondere den durch die einzelnen Prognosemethoden verursachten Fehlern begegnet werden kann (vgl. auch [Die00, Bis06]).

Auf Basis dieser Überlegungen wird das resultierende Prognosemodell als Struktur aus den relevanten Variablen, deren Beziehungen untereinander, der Zuweisung geeigneter Skalen und Prognosemethoden sowie der Assoziation geeigneter Datenquellen zur Erhebung von Datenwerten dargestellt. Das in dieser Arbeit verwendete Metamodell für die Beschreibung dieser Struktur ist in Abbildung 6.53 zusammengefasst. Das zentrale Element *Prediction Model* enthält dabei zunächst auf Basis der Informationen aus dem Prognosenetz eine Menge von Variablen (*Variables*), wobei abhängige Variablen über eingehende Kanten (*ParentEdges*) Beziehungen zu anderen Variablen ausdrücken können. Das Attribut *Time Offset* gibt dabei den Zeitabstand Δ aus dem Prognosenetz an. Jede Variable besitzt nach diesem Modell zudem einen eindeutigen Namen (*Name*), eine Skala (*Scale*) und – für den Fall, dass es sich hierbei um eine Nominalskala handelt – die Menge der erlaubten Werte (*Nominal Type*). Falls der Wert der Variablen gemessen bzw. erhoben werden kann (z. B. über einen Sensor), so kann angegeben werden, welche Art von Schnittstelle hierfür aufgerufen werden soll (*Measuring Adapter*).

Die für die Vorhersage des Variablenwertes bestimmten Prognosemethoden (*Method Preknowledge*) können unabhängige oder nur abhängige Variablen in ihre Berechnung einbeziehen. Entsprechend wird über das Attribut *Learning Method Preknowledge* festgelegt, ob ein Online-Lernen von Zusammenhängen für die Prognosemethoden mit oder ohne die Werte der unabhängigen Variablen erfolgen soll. Das Attribut *Default Learning Distance* definiert, nach welcher Zeitdauer der Datenerhebung eine Bildung von Zusammenhängen zwischen den Werten der abhängigen und der unabhängigen Variablen erlaubt ist.

Allgemeine Konfigurationsparameter des Prognosemodells umfassen schließlich die Einstellungen hinsichtlich der Definition der Länge eines Zeitschritts (*Length of Time Step*), die Menge der zu speichernden historischen Daten (*Length of History*), so dass ältere Daten verworfen werden können, und die Häufigkeit der Datenerhebung (*Frequency of History*), welche angibt, wie oft innerhalb eines Zeitschritts Daten gemessen bzw. erhoben werden sollen. Die Elemente *Method Preknowledge* und *Measuring Adapters* können zudem um weitere individuelle Eigenschaften (*Specific Settings*) und den Verweis auf ihre Implementierungen (*Implementations*) ergänzt werden. Nähere Details hierzu können der Arbeit von MEINERS [Mei09] entnommen werden.

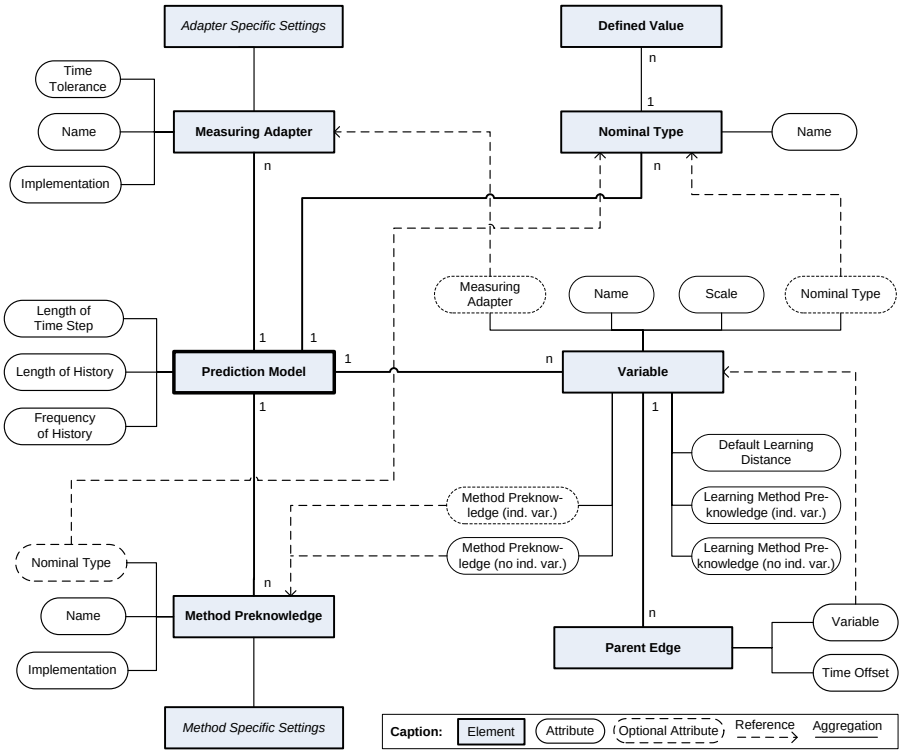


Abbildung 6.53: Metamodell für die Erstellung von Prognosemodellen (ZML11)

6.5.3 Durchführung von Prognosen

Um ausreichend Erfahrungswerte für die Durchführung von Prognosen über den Ausführungskontext von prozessorientierten Anwendungen zu erhalten, muss dass durch den Anwendungsentwickler erstellte Prognosemodell an das Ausführungssystem übergeben werden, welches für die Prozessausführung verantwortlich sein soll. Dies bedeutet, dass im Fall einer verteilten Ausführung, ein gemeinsames (zur Laufzeit erweiterbares) Prognosemodell mit grundlegendem Anwendungswissen an alle potentiell an der Ausführung teilnehmenden Ausführungseinheiten verteilt werden muss, damit mit der Erhebung von jeweils individuellen Datensätzen zu den im Prognosenetz spezifizierten Variablen begonnen werden kann. Für jede im Prognosenetz angegebene Beziehung zwischen zwei Variablen enthält ein solcher Datensatz jeweils die Kombination der unabhängigen und der abhängigen Variablen (vgl. [Mei09] zu Details). Die Messung bzw. Erhebung der hierfür benötigten Kontextdaten sowie die Integration des Prognosesystems findet im Rahmen des hier vorgeschlagenen Ansatzes durch ein unterstützendes Kontextmanagementsystem statt (vgl. auch Abbildung 6.49).

Soll zur Laufzeit der Anwendung eine Prognose erstellt werden, so wird das Prognosenetz in dem für die Prognose relevanten Zeitraum *aufgefaltet*. Ein *aufgefaltetes Prognosenetz* (*Unfolded Prediction Net*) stellt die Variablen des Prognosenetzes zu jedem definierten Zeitpunkt innerhalb des relevanten Zeitraumes dar. Zu einem Prognosenetz $\mathcal{N} = (W, E)$ ist das aufgefaltete Progno-

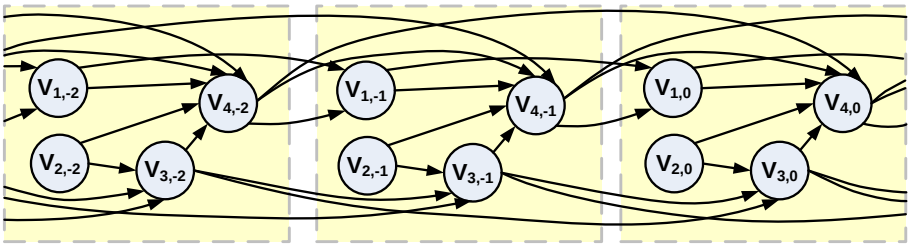


Abbildung 6.54: Ausschnitt des aufgefalteten Prognosenetzes aus Abbildung 6.51 (Mei09, ZML11)

senetz formal als Graph $\mathcal{N}_A = (W_A, E_A)$ mit $W_A = \bigcup_{j \in \mathbb{Z}} \{V_{1,j}, ..., V_{n,j}\}$ und $E_A = \{(V_{i,j}, V_{i',j'}) \in W_A \times W_A : \exists (V_i, \Delta, V_{i'}) \in E$ mit $j' - j = \Delta\}$ definiert [Mei09]. Soll zum Beispiel die Position (vgl. Variable V_3 in Abbildung 6.51) auf Basis der vorhergehenden Position vorhergesagt werden, so enthält das Prognosenetz die Kante $V_3 \overset{1}{\rightarrow} V_3$. Das aufgefaltete Prognosenetz enthält entsprechend die Kanten ...$\rightarrow V_{3,-2} \rightarrow V_{3,-1} \rightarrow V_{3,0} \rightarrow V_{3,1} \rightarrow V_{3,2} \rightarrow$ ... wobei ..., -2, -1, 0, 1, 2, ... Zeitpunkte darstellen (Markov-Kette) [ZML11]. Abbildung 6.54 stellt den Ausschnitt des aufgefalteten Prognosenetzes aus dem Beispiel in Abbildung 6.51 in dem genannten Zeitraum graphisch dar.

Prognosenetze erlauben nur Prognosen entlang der Kanten des Graphs, was die Prognose zur Laufzeit vereinfacht. Sie sind daher insbesondere auch im Umfeld leistungsschwächerer mobiler Geräte einsetzbar, für welche die Kontextdatenprognose aufgrund der höheren Dynamik des Kontextes eine besonders große Bedeutung besitzt [MZL10, ZML11]. Das aufgefaltete Prognosenetz erspart zudem die explizite Berücksichtigung von Zeitdifferenzen an den Kanten des Prognosenetzes. Nähere Details sowie ein Beweis für die Zyklenfreiheit aufgefalteter Prognosenetze ist der Arbeit von MEINERS [Mei09] zu entnehmen.

Da das Prognosemodell die vom Anwendungsentwickler bestimmten Prognosemethoden zur Vorhersage der im Prognosenetz enthaltenen Variablen spezifiziert, kann bei Bedarf jederzeit eine Prognose der Variablenwerte angefragt werden. Für jeden Knoten im relevanten Ausschnitt des aufgefalteten Prognosenetzes wird dazu eine Prognoseeinheit (*Predictor*) erzeugt, welche auf Basis der Graphstruktur die Werte anderer Variablen einholt und diese an die vorgegebene Prognosemethode weiterreicht, um den Wert der Variablen zum gegebenen Zeitpunkt bestimmen zu lassen. Diese Vorgehensweise wird entlang den Kanten des Graphs solange durchgeführt, bis der jeweils erforderliche Wert berechnet bzw. bekannt ist. Der folgende Algorithmus (vgl. Listing 6.3) formalisiert die Prognose des Wertes $v_{i,j}$ der Variablen V_i zum Zeitpunkt j. Da für jede angefragte Prognose eine einzelnen Prognoseeinheit nicht mehr als eine Prognose durchführen muss, kann – ohne Betrachtung der integrierten Prognosemethoden – eine lineare Komplexität erreicht werden [Mei09, MZL10, ZML11].

```
1  if v_{i,j} already predicted then
2    return v_{i,j}
3  else
4    if v_{i,j} known then
5      return v_{i,j}
6    else
7       let parent-predictors predict their values
8       predict v_{i,j} with own method using these values
9       return v_{i,j}
10   end if
11 end if
```

Listing 6.3: Algorithmus zur Prognose des Wertes $v_{i,j}$ der Variable V_i zum Zeitpunkt j (Pseudocode) (Mei09, MZL10, ZML11)

6.5.4 Prognosen zu dynamischen Kontexten

Um Prognosen für bislang unbekannte Anwendungskontexte zu unterstützen, können Prognosemodelle und/oder Datensätze von Erfahrungswerten zur Laufzeit ergänzt bzw. konkretisiert werden. Dabei sind in Hinblick auf eine verteilte Prozessausführung prinzipiell drei unterschiedliche Vorgehensweisen möglich [ZML11]:

- **Austausch von Datensätzen:** Es können durch einzelne Ausführungseinheiten gesammelte Datensätze zur Laufzeit ausgetauscht oder integriert werden, um einzelne Prognosen durch eine (größere) Menge von historischen Daten zu unterstützen oder durch diese eine breitere Grundlage für zukünftige Prognosen aufzubauen. Dabei ist zu beachten, dass die gesammelten Daten teilweise die Individualität der Ausführungseinheiten bzw. ihrer Ausführungskontexte widerspiegeln. Eine Übernahme von fremden Datensätzen bzw. deren Integration kann somit auch zu einer Verschlechterung der Prognosequalität führen, z.B. wenn gegenläufige Trends integriert werden.
- **Anpassung des Prognosemodells:** Das Prognosemodell kann zur Laufzeit verändert, erweitert oder ersetzt werden, so dass neues oder angepasstes Anwendungswissen über neue Variablen und/oder Zusammenhänge zwischen neuen und bestehenden Variablen dynamisch eingebracht werden kann oder Prognosemethoden neu konfiguriert werden können. Mit dieser Möglichkeit können Prognosen ermöglicht werden, welche für die Ausführung neuer prozessorientierter Anwendungen benötigt werden. Es wird jedoch zwischen dem Einbringen des Anwendungswissens und der ersten Prognose über den neuen Anwendungskontext ein gewisser Zeitraum benötigt, in dem die Erhebung von Datensätzen zum „Erlernen“ der individuellen Zusammenhänge möglich ist.
- **Einbringen von Anwendungswissen auf Instanzebene:** Es ist möglich, zur Laufzeit externes Wissen über bestimmte Variablenwerte zu

übertragen und für eine bestimmte Prognose mit den bereits „gelernten“ individuellen Zusammenhängen der übrigen Variablen des Prognosenetzes zu integrieren. Das bekannte externe Wissen kann dabei von der Anwendung an das Prognosesystem übergeben werden. Die Variablenwerte gelten in diesem Fall als bekannt und können die Prognosen durch das Vorhandensein von „sicherem“ Wissen gezielt beeinflussen.

Im Fall von dynamisch verteilt ausgeführten Prozessen können prinzipiell alle drei Strategien zum Einsatz kommen, um eine proaktive Anpassung der Prozessausführung zu erlauben. Die Variante des anwendungsseitigen Einbringens von externem Wissen auf Instanzebene eignet sich jedoch besonders gut, um auch die in Abschnitt 6.5.1 identifizierten Ad-hoc-Prognosen von bisher lokal unbekannten Anwendungskontexten zu unterstützen. Bekannte Zusammenhänge über prozessmodell- oder sogar prozessinstanzabhängige Ausführungskontexte können dabei als konkrete Datensätze bzw. als „sicheres“ Wissen mit dem verteilt auszuführenden Prozess (z. B. im Rahmen der Migrationsdaten) transportiert werden und bei Bedarf für eine Prognose herangezogen werden. Als Grundlage für die Prognose dient dabei ein relativ allgemeines Prognosemodell, auf dessen Basis die individuellen Eigenschaften der Ausführungsumgebungen erhoben werden können [ZML11]. Als ein mögliches Beispiel für ein solches Prognosemodell und dessen Anreicherung mit externem Wissen wird im folgenden Abschnitt die Vorhersage von Dienstverfügbarkeiten in dynamischen Umgebungen vorgestellt.

6.5.5 Vorhersage von Dienstverfügbarkeiten

Die Vorhersage der Verfügbarkeit von lokalen Ressourcen stellt für die dynamische Verteilung der Prozessausführung einen besonders wichtigen Aspekt dar. Nach der in Abschnitt 2.1 vorgenommenen Definition kann es sich bei einer Ressource prinzipiell um einen menschlichen Prozessteilnehmer, eine Maschine oder eine Software-Anwendung bzw. um eine Gruppe der genannten Individuen handeln. Da insbesondere die Kapselung von Softwarefunktionalitäten als elektronische Dienste ein großes Flexibilitätspotential für die Ausführung von prozessorientierten Anwendungen besitzt (vgl. Abschnitt 3.4), wird in diesem Abschnitt als Beispiel für die Anwendung der strukturierten Kontextdatenprognose die Verfügbarkeit von Diensten als aufzurufene Ressourcen zur Ausführung einzelner Aktivitäten eines Prozesses betrachtet. Dabei sei angenommen, dass die aus der Prozessbeschreibung referenzierten Dienste, wie in Abschnitt 6.2.3 beschrieben, eine schwache Bindung besitzen und somit prinzipiell gegen äquivalente Dienste austauschbar sind. Als Rahmenbedingung sei zudem angenommen, dass die zur Prozessausführung benötigten Dienste nicht zu allen Zeitpunkten in allen Ausführungsumgebungen zur Verfügung stehen. Beispiele für Funktionalitäten solcher Dienste seien hier unter anderem das Versenden einer Kurznachricht (*SMS-Dienst*), das Ausdrucken eines Dokuments (*Druckdienst*) oder das Abrufen von Aktienkursen (*Aktiendienst*).

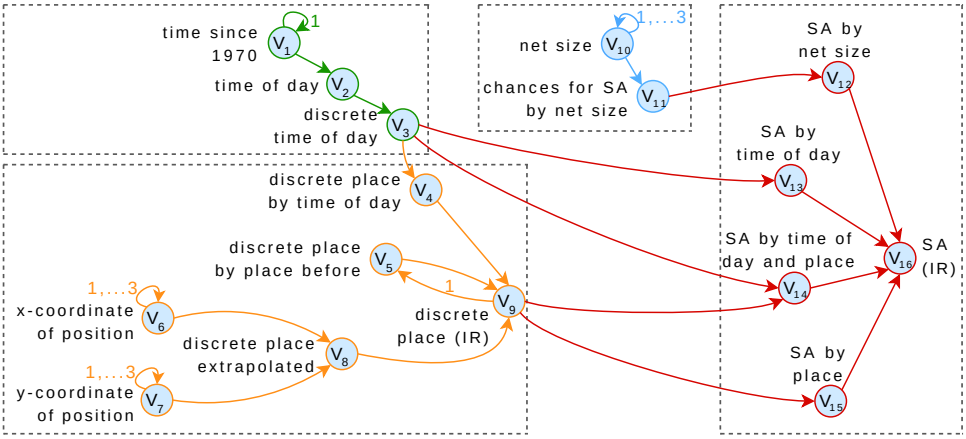


Abbildung 6.55: Prognosenetz zur Vorhersage von Dienstverfügbarkeiten (SA = *Service Availability*) (Mei09, MZL10, ZML11)

Wie bereits in den vorhergehenden Abschnitten motiviert wurde, kann bei einer verteilten Ausführung der genaue Typ von Dienst, dessen Verfügbarkeit von einer bestimmten (Ziel-)Ausführungseinheit prognostiziert werden soll, erst zur Laufzeit des Prozesses und somit erst kurz bzw. unmittelbar vor der benötigten Prognose determiniert werden. Dabei ist anzunehmen, dass die Verfügbarkeit eines Dienstes mit seiner Funktionalität in einem engen Zusammenhang steht. Zum Beispiel ist die Wahrscheinlichkeit für die Verfügbarkeit eines Druckdienstes innerhalb eines großen Bürogebäudes oder in einem Wohngebäude relativ hoch, draußen im freien Gelände jedoch relativ gering. Im Gegensatz dazu ist die Verfügbarkeit eines SMS-Dienstes im freien Gelände aufgrund der besseren Konnektivität zu einem (mobilen) Netzwerk besser einzuschätzen als im Keller eines Bürogebäudes. Ein Aktiendienst ist mit großer Wahrscheinlichkeit (nur) über das Internet zu erreichen, also von der Verfügbarkeit einer Internetverbindung abhängig. Andere (dienstleistungsorientierte) Funktionalitäten sind unter Umständen nur innerhalb eines gewissen Zeitfensters verfügbar, zum Beispiel nur tagsüber oder nur an Werktagen [MZL10, ZML11].

Auf Basis dieser Überlegungen ist die Prognose von „Dienstverfügbarkeit“ für beliebige Dienste im Allgemeinen zu ungenau. Ebenso ist es aufgrund der prinzipiell unendlichen Anzahl an möglichen Diensten nicht praktikabel, individuelle Prognosemodelle für jeden einzelnen Dienst zu erstellen, zu verteilen und (vorsorglich) mit der Erhebung von Daten über ihre Verfügbarkeit zu beginnen, damit im Fall einer Prognose zu der Verfügbarkeit dieses Dienstes die benötigten historischen Daten zur Verfügung stehen. Es kann jedoch auf der Basis der strukturierten Kontextdatenprognose ein Prognosenetz erstellt werden, welches die oben genannten Überlegungen zur Dienstverfügbarkeit im Allgemeinen berücksichtigt, auf Basis der dort gegebenen Zusammenhänge die Verfügbarkeit individueller – lokal relevanter – Dienste „erlernt“ und zur Laufzeit mit weiteren Informationen zu der Verfügbarkeit konkreter und bisher unbekannter Dienste angereichert werden kann.

Abbildung 6.55 zeigt ein (vereinfachtes) Prognosenetz für die Prognose von Dienstverfügbarkeiten, welches die zur Entwicklungszeit angenommenen Abhängigkeiten der Dienstverfügbarkeit zu den als relevant identifizierten Variablen modelliert. Dabei wird angenommen, dass die Dienstverfügbarkeit von der Größe des zugänglichen Netzwerks abhängig ist, da die Anzahl von Knoten innerhalb eines Netzwerks die Anzahl möglicher Dienstanbieter beeinflussen kann. Zudem wird die Position der Ausführungsumgebung berücksichtigt, womit insbesondere einer potentiellen Mobilität der Ausführungseinheit Rechnung getragen wird. Als möglicher Einflussfaktor für die Position der Ausführungseinheit wird die Tageszeit herangezogen, da mobile Ausführungseinheiten oft an einen (bestimmten) Benutzer gebunden sind, dessen Position in diesem Beispiel von seiner Tätigkeit zu einer bestimmten Uhrzeit abhängig ist [Mei09, MZL10, ZML11].

Die Abbildung der als relevant identifizierten Einflussfaktoren auf die in Abbildung 6.55 dargestellten Mengen von Variablen resultiert aus der Zuweisung geeigneter Prognosemethoden und der Anwendung von *Coprocessing* (vgl. Abschnitt 6.5.2). Auf Basis der gemessenen Zeit (V_1) wird die Tageszeit als periodische Variable bestimmt (V_2) und diskretisiert (V_3), zum Beispiel als stundenweise Zeitfenster. Die Position der Ausführungseinheit (diskrete Orte) zu einem bestimmten Zeitpunkt (V_9) wird parallel aus der Tageszeit (V_4), der vorhergehenden Position (V_5) und der aus X- und Y-Koordinate ($V6$ und V_7) extrapolierten Position ($V8$) berechnet. Die Dienstverfügbarkeit *SA* zu einem bestimmten Zeitpunkt (V_{16}) wird schließlich durch die parallele Anwendung von vier Prognosemethoden berechnet, welche die bisher erhobenen bzw. prognostizierten Zwischenergebnisse bezüglich der Netzgröße (V_{12}), der Tageszeit (V_{13}), des Zusammenhangs zwischen Tageszeit und Ort (V_{14}) und der reinen Position (V_{15}) beisteuern. Da es sich aufgrund der Diskretisierung weitestgehend um nicht-numerische Werte handelt, werden zur Prognose der genannten Variablen in erster Linie Wahrscheinlichkeitstabellen eingesetzt. Die Prognose der Dienstverfügbarkeit auf Basis der Netzgröße verwendet die Methode der linearen Regression. Die vier Teilergebnisse werden in diesem Beispiel durch einen einfachen Mehrheitsentscheid integriert und das Ergebnis der Integration in Variable V_{16} gespeichert. Die in der Abbildung dargestellten farblichen Unterschiede zwischen den Kanten dienen hierbei nur der Übersichtlichkeit bzw. der besseren Zuordnung der Kanten zu den Variablen [Mei09, MZL10, ZML11] (vgl. insbesondere [Mei09] zu Details).

Das Prognosenetz kann nun in einen dienstunabhängigen Teil (Variablen V_1 bis V_{10}) und einen dienstabhängigen Teil (Variablen V_{11} bis V_{16}) untergliedert werden [Mei09, ZML11]. Der dienstunabhängige Teil enthält das allgemeine Anwendungswissen zu Ort, Zeit und Netzgröße und kann auch als Hintergrund für Prognosen über andere (prozessmodellunabhängige) Anwendungskontexte verwendet werden. Der dienstabhängige Teil und die zur Laufzeit erhobenen Erfahrungswerte können für beliebige Typen von Diensten wiederverwendet werden. Wird die Verfügbarkeit eines bestimmten Dienstes festgestellt, so muss nur der dienstabhängige Teil der gesammelten Datensätze aktuali-

siert werden. Dazu wird für jeden Dienst ein neuer Eintrag in der jeweiligen Wahrscheinlichkeitstabelle hinzugefügt. Die Erhebung von Daten bezüglich der Verfügbarkeit von Diensten kann daher in individueller Art und Weise geschehen, zum Beispiel durch eine (aktive) periodische Abfrage aller zur Verfügung stehenden Dienste oder durch das passive Vermerken der Dienstverfügbarkeit, wenn der Dienst im Rahmen der regulären Prozessausführung durch die Ausführungseinheit erfolgreich aufgerufen werden konnte. Während die erstgenannte Strategie mehr Datensätze über die Verfügbarkeit von verschiedenen (unter Umständen lokal nicht relevanten) Diensten liefert (und daher auch mehr Speicherplatz erfordert), entspricht die zweite Strategie eher der zugrunde gelegten Definition von *Kontext* (vgl. Abschnitt 3.5.1), da hierbei nur die jeweils *lokal relevanten* Informationen erhoben werden [ZML11].

In beiden Fällen kann das durch die Erhebung von Datensätzen zu den Variablen des Prognosenetzes aufgebaute Vorwissen zur Laufzeit genutzt werden, um bei Bedarf Datensätze des dienstabhängigen Teils zwischen verschiedenen Ausführungsseinheiten auszutauschen. So kann zum Beispiel gesammeltes Wissen darüber ausgetauscht werden, an welchem diskreten Ort bestimmte Dienste zur Verfügung stehen. Durch die einfache Übertragung von Datensätzen mit den Verteilungsinformationen der auszuführenden Prozessinstanz können zudem die Einträge der Wahrscheinlichkeitstabellen im dienstabhängigen Teil des Prognosemodells für bislang unbekannte Dienste individuell gesetzt werden. Zum Beispiel kann auf diesem Weg ein neuer Eintrag in die Wahrscheinlichkeitstabelle zur Prognose von V_{15} eingefügt werden, welcher besagt, dass die Verfügbarkeit von Druckdiensten im Bürogebäude 100% beträgt. In Kombination mit einer Prognose der dienstunabhängigen Variablen des Prognosenetzes (z. B. der Prognose, dass das Bürogebäude mit großer Wahrscheinlichkeit in den nächsten 30 Minuten erreicht werden wird), kann somit auch ad-hoc eine relativ aussagekräftige Prognose über die Verfügbarkeit bislang lokal unbekannter Diensttypen bereitgestellt werden [ZML11]. Eine Anwendung dieses Konzepts sowie eine Evaluation der hierdurch ermöglichten Prognoseergebnisse erfolgt im Rahmen der Verbesserung von Migrationsentscheidungen für die kontextbasierte Kooperation in Abschnitt 8.4.1.

6.5.6 Zusammenfassung und Einordnung

Das im Rahmen dieser Arbeit vorgestellte Verteilungskonzept erlaubt prinzipiell gleichermaßen eine proaktive Anpassung der Verteilung auf Basis aktueller Informationen und auf Basis von Prognosen über zukünftige Ausführungskontexte. Die Anpassung der Verteilung selbst ist dabei in beiden Fällen mit relativ geringem Aufwand durchführbar. Aufgrund der Individualität prozessorientierter Anwendungen und der Verteilung ist jedoch die Erhebung geeigneter historischer Daten für die Durchführung der Prognosen über die jeweils relevanten Ausführungskontexte ein großes Problem. Es wird daher ein ebenfalls flexibles Prognosekonzept benötigt, welches in der Lage ist,

Prognosen über wechselnde Anwendungskontexte zu unterstützen und dabei möglichst auch ad-hoc Prognosen über Anwendungskontexte zu erlauben, für die bisher auf den potentiellen Zielausführungseinheiten nicht explizit Erfahrungswerte vorliegen.

Das hier angewendete Rahmenwerk der *strukturierten Kontextdatenprognose* relativiert die oben genannte Problematik, indem einerseits Anwendungswissen in Form von Prognosemodellen zur Laufzeit eingebracht werden kann, so dass langfristig Prognosen zu neuen oder geänderten Anwendungskontexten analog zur Ausführung neuer bzw. inhaltlich angepasster Prozessmodelle durchgeführt werden können. Prognosen über bisher lokal unbekannte Ausführungskontexte können zudem kurzfristig durch den gezielten Transfer bzw. Austausch von speziellen Zusammenhängen und individuell zur Laufzeit gesammelten Datensätzen und die dadurch ermöglichte Integration von Informationen erfolgen. Als Beispiel wurde hierzu die Prognose von Dienstverfügbarkeiten in dynamischen mobilen Umgebungen vorgestellt.

Naturgemäß ist die Anwendbarkeit von Zukunftsprognosen bei einer sehr hoher Dynamik bzw. bei vielfältigen Änderungen außerhalb der erwarteten Zusammenhänge stark begrenzt. Zudem besteht trotz der hier vorgeschlagenen Flexibilisierung weiterhin ein relativ hoher Aufwand für die Erstellung und ggf. Verteilung von Prognosemodellen. Dieser Aufwand ist daher für einzelne Prozesse mit nur geringer Wiederholungsrate (d. h. Prozessmodelle, von denen nur vereinzelt oder einmalig Prozessinstanzen abgeleitet und ausgeführt werden) unter Umständen als unverhältnismäßig hoch einzuschätzen. Zudem kommt es durch die Verteilung der Prognosemodelle im Vorfeld der Prozessausführung auch für später nicht an der Prozessausführung beteiligte Ausführungseinheiten zu einem kontinuierlichen Aufwand für die Erhebung von Trainingsdaten. Es muss daher im Einzelfall abgewogen werden, welchen Mehrwert die Fähigkeit zur Kontextdatenprognose im Vergleich zu den als Alternativen genannten (naiven) proaktiven Anpassungsstrategien aufweist. Die hier vorgestellte Methodik der Entwicklung eines allgemeinen Prognosemodells und dessen Anreicherung mit Anwendungswissen auf Instanzebene stellt daher einen Kompromiss dar, welcher die Anforderungen dynamisch verteilt ausgeführter Prozesse so weit wie möglich erfüllt.

Als Zwischenergebnis kann festgehalten werden, dass die proaktive Anpassung der verteilten Prozessausführung durch die Flexibilität des Prognosesystems bedingt wird. Durch die strukturierte Kontextdatenprognose können die durch die einzelnen Ausführungseinheiten bereitgestellten Informationen (vgl. Abschnitt 6.3) für die Ableitung von zukünftigem Anpassungsbedarf integriert werden. Darauf aufbauend ist somit prinzipiell auch die Entwicklung komplexerer Anpassungsstrategien der verteilten Ausführung unter Berücksichtigung möglicher zukünftiger Kontextdaten möglich.

6.6 Zusammenfassung

Für die Flexibilisierung verteilter Prozessausführungen wurden mit der dynamischen Verteilung, mit der Überwachung und Steuerung entfernt ausgeführter Prozesspartitionen und der kontextbasierten Darstellung von Benutzungsschnittstellen für menschliche Prozessteilnehmer drei wesentliche Aufgabenstellungen identifiziert. Die Hauptproblematik bei allen drei Themengebieten stellt dabei die Tatsache dar, dass bei einer prozessorientierten Anwendung zu deren Entwicklungszeit eine spätere Verteilung oder sogar die konkrete Verteilungskonfiguration (im Wesentlichen zusammengesetzt aus dem Zielort der Verteilung, der Verteilungsgranularität und der Zuordnung auszuführender Partitionen zu Ausführungseinheiten) unter Umständen nicht bekannt ist. Es wurden daher im Rahmen dieser Arbeit zu jedem der genannten Teilaspekte geeignete generelle Prinzipien und Entwicklungsmethodiken vorgestellt, um zu einer höheren Flexibilität für die verteilte Ausführung prozessorientierter Anwendungen beizutragen. Dabei wurde jeweils ein im Rahmen dieses Beitrags erarbeiteter möglicher Ansatz zur Umsetzung der getroffenen Entwicklungsentscheidungen vorgestellt. Die konkreten Beiträge dieses Kapitels gliedern sich in

- einen Ansatz zur *nicht-invasiven Migration* von laufenden Prozessinstanzen, um damit eine ungeplante Verteilung der Prozessausführung auf mehrere Ausführungseinheiten abzubilden [ZKML10, ZHKK10],

- einen Ansatz zur anbieterseitigen Bereitstellung von Überwachungs- und Steuerungsfunktionalitäten auf Basis eines grundlegenden Referenzmodells zur Abbildung eines Prozessmanagementsystems als *verwaltbare Ressource* [Zap07, vRZ07, ZBH+10], und

- einen Ansatz zur Beschreibung *abstrakter Benutzungsschnittstellen* und zu deren kontextbasierter Transformation in konkrete Interaktionselemente, welche die konkreten Eigenschaften der Ausführungsumgebung und den situativen Kontext des individuellen menschlichen Prozessteilnehmers berücksichtigen [ZVBK09, ZBV09].

Gemeinsam ist den drei Teilansätzen die Betrachtung einer als Dienst gekapselten Ausführungseinheit mit Möglichkeiten zur Aus- bzw. Fortführung von (laufenden) Prozessinstanzen, zur Bereitstellung von Überwachungs- und Steuerungsfunktionalitäten und zur Interpretation von abstrakten Beschreibungen zur Repräsentation von Benutzungsschnittstellen. Es wird somit das Paradigma des *Process-Management-as-a-Service* (*PMaaS*) als wesentliches Unterstützungskonzept für die Flexibilisierung verteilter Prozessausführung adaptiert. Zudem erfolgt in allen drei Teilansätzen eine explizite Berücksichtigung mobiler Ausführungseinheiten und deren besonderer Anforderung an eine Prozessausführung ohne permanente Netzwerkverbindung.

Als Ergänzung der genannten Beiträge wurde betrachtet, wie eine dynamische Anpassung der Verteilung nicht nur reaktiv, sondern auch proaktiv vorgenommen werden kann. Da eine prozessorientierte Anwendung nahezu beliebige (funktionale und nicht-funktionale) Aspekte und Anwendungskontexte integrieren kann, wurde die Bereitstellung von Prognosen über dynamische Anwendungskontexte als ergänzende Aufgabenstellung identifiziert. Als zu der oben genannten Problematik komplementäres Problem stellt sich hierbei die Tatsache dar, dass zur Entwicklungszeit von potentiell an einer verteilten Prozessausführung beteiligten Ausführungseinheiten nicht alle möglichen Inhalte späterer Prozessausführungen bekannt sein können. Als ergänzenden Beitrag enthält dieses Kapitel daher weiterhin

- einen Ansatz zur *Kontextdatenprognose* für dynamisch verteilt ausgeführte Anwendungen und dessen Nutzbarmachung für die Anpassung der Verteilung prozessorientierter Anwendungen [MZL10, ZML11],

Bei Vorliegen der vorgeschlagenen Infrastruktur ist eine dynamische Verteilung von prozessorientierten Anwendungen sowie die Überwachung und Steuerung verteilt ausgeführter Prozesse möglich, ohne dass zur Entwicklungszeit oder bei der Instantiierung des jeweiligen Prozesses Kenntnis über eine spätere Verteilung und die zur Laufzeit tatsächlich vorliegende Verteilungskonfiguration bestehen muss. Es ist somit unter den angegebenen Rahmenbedingungen eine *spontane Verteilung* des Prozesses möglich, welche die Anforderungen an eine *fortwährende Anpassbarkeit* der Verteilungskonfiguration und der Maßnahmen zur Überwachung und Steuerung von entfernt ausgeführten Prozesspartitionen erfüllt. Die verteilte Ausführung von prozessorientierten Anwendungen entspricht daher dem dieser Arbeit zugrunde gelegten Flexibilitätsbegriff. Der notwendige Aufwand für eine Anpassung der Verteilung von individuellen prozessorientierten Anwendungen ist als gering einzuschätzen, da die prozessorientierten Anwendungen hierfür nicht selbst einzeln verändert werden müssen. Der Anpassungsaufwand begrenzt sich daher auf eine (im Idealfall) einmalige Installation und Bereitstellung der unterstützenden dienstbasierten Infrastruktur und (bei Bedarf) auf die Definition von benutzerdefinierten Rahmenbedingungen für die Verteilung, Überwachung und Steuerung des betrachteten Prozesses. Die Verarbeitung des fachlichen Kontrollflusses und der Ausführung der im Prozessmodell festgelegten Aktivitäten ist somit von der (möglichen) Verteilung, Überwachung und Steuerung des Prozesses losgelöst. Hinsichtlich der Erfüllung der fachlichen Aufgabe ist daher eine vollständige Realisierung von Verteilungstransparenz, aber auch deren im individuellen Anwendungskontext sinnvolle Einschränkung möglich.

Eine höhere Qualität der Flexibilität in Bezug auf die Vergrößerung des Anpassungsspielraums und der Ermöglichung von proaktiven Anpassungen kann durch die kontextbasierte Anpassung von Benutzungsschnittstellen und durch die Vorhersage von Ausführungskontexten erreicht werden. Durch die vorgeschlagene abstrakte Beschreibung von Benutzungsschnittstellen und de-

ren Abbildung auf konkrete Interaktionselemente zur Laufzeit kann eine fortwährende Anpassbarkeit an den Kontext individueller menschlicher Benutzer erreicht werden. Im Gegensatz zu den zuvor genannten Anpassungen ist hierbei auf der Ebene des einzelnen Prozessmodells eine Vorbereitung auf die Verteilung der prozessorientierten Anwendung notwendig, da etwaige konkrete Benutzungsschnittstellen durch die vorgeschlagene abstrakte Repräsentation ersetzt werden müssen. Sollen inhaltliche Anpassungen an laufenden Prozessen vermieden werden, so ist daher zur Entwicklungszeit des Prozesses zumindest das Vorliegen der Kenntnis einer potentiellen Verteilungsabsicht als Motivation für die abstrakte Beschreibung von Benutzungsschnittstellen und deren Integration in die Prozessbeschreibung erforderlich. Eine konkrete Anpassung des Prozesses auf verschiedene mögliche Verteilungskonfigurationen oder gar deren genaue Kenntnis ist zur Entwicklungszeit jedoch nicht notwendig. Der für die Flexibilisierung notwendige Aufwand begrenzt sich daher auf die (im Idealfall) einmalige Installation und Bereitstellung der unterstützenden dienstbasierten Infrastruktur und auf die Integration der abstrakten Beschreibungen für die Anpassbarkeit der Benutzungsschnittstellen je Prozessmodell.

Die zusätzliche Ermöglichung von proaktiven Anpassungen der Verteilung macht keine Änderung an den ausgeführten Prozessen erforderlich. Im Vergleich zu den zuvor genannten Beiträgen zur Flexibilisierung erfordert die Vorhersage von Ausführungskontexten jedoch einen verhältnismäßig großen zusätzlichen Entwicklungsaufwand durch die Notwendigkeit zur Erstellung individueller Prognosemodelle. Für die spontane Verteilung einzelner Prozessinstanzen ist dieser Aufwand kaum zu rechtfertigen. Ein grundlegendes Prognosemodell zur Verfügbarkeit von Diensten in dynamischen Kontexten kann jedoch für eine größere Klasse von verteilt auszuführenden Prozessen mit speziellem Bedarf für solche Prognosen einen Mehrwert erbringen.

Aus den in diesem Kapitel vorgeschlagenen Konzepten kann insgesamt abgeleitet werden, dass eine höhere Flexibilität für die verteilte Ausführung von Prozessen erlangt werden kann, wenn nicht jeder einzelne Prozess zur Laufzeit inhaltlich angepasst werden muss, sondern die Verteilung und Verwaltung von prozessorientierten Anwendungen durch eine entsprechende Middleware übernommen wird. Hierfür ist im Idealfall eine einmalige Erweiterung der bestehenden Systeme zum Prozessmanagement und insbesondere zur Prozessausführung erforderlich. Eine unterstützende dienstbasierte Infrastruktur hierfür wird im folgenden Kapitel vorgestellt und wurde in wesentlichen Teilen im Rahmen dieser Arbeit prototypisch realisiert. Der entsprechende einmalige Umsetzungsaufwand und der Aufwand zur Anpassung von prozessorientierten Anwendungen zur Laufzeit wird in Kapitel 8 diskutiert und bewertet.

7 Middleware für dynamisch verteilt ausgeführte Prozesse

Ein wichtiges Teilkonzept dieser Arbeit besteht darin, für eine verteilte Ausführung prozessorientierter Anwendungen nicht die Anwendungen selbst anzupassen, sondern die mit der dynamischen Verteilung, Überwachung, Steuerung und der verteilten Ausführung verbundenen Aufgaben für die Anwendungen transparent durch eine unterstützende Softwareschicht zu realisieren. Ziel hierbei ist es, durch eine Bereitstellung geeigneter Softwarekomponenten und eine (im Idealfall einmalige) Anpassung bestehender Prozessmanagementsysteme eine Ausgangsposition zu schaffen, in welcher Prozesse hinsichtlich ihrer Verteilung zur Laufzeit fortwährend anpassbar gehalten werden können. Aus der Sicht der Anwendungen (d. h. der verteilt ausgeführten Prozesse) handelt es sich bei dieser Softwareschicht um eine die prozessausführenden Systeme unterstützende *Middleware* (vgl. Abschnitt 2.7.1). Die Konzeption und Bereitstellung allgemein anwendbarer Middleware-Komponenten zur Erweiterung von Ausführungssystemen als Voraussetzung einer flexiblen verteilten Prozessausführung sind Inhalt dieses Kapitels.

Bei der Analyse der in Kapitel 4 vorgestellten Fallbeispiele wurde zudem deutlich, dass die Möglichkeit zur dynamisch verteilten Prozessausführung insbesondere im Kontext mobiler Ausführungseinheiten und bei deren Zusammenwirken mit klassischen Prozess-Engines als Teil der Middleware stationärer Systeme eine große Relevanz besitzt. In diesem Kapitel wird deshalb eine technische Infrastruktur vorgeschlagen, welche die in Kapitel 6 zur Flexibilisierung der verteilten Prozessausführung vorgeschlagenen Teilansätze auch unter spezieller Berücksichtigung mobiler Ausführungseinheiten umsetzt. Die einzelnen Komponenten der konzeptionellen Ansätze wurden dazu im Rahmen des praktischen Teils dieser Arbeit prototypisch als dienstorientierte Architektur implementiert und stellen somit für bestehende mobile und stationäre Ausführungseinheiten eine komponentenbasierte Middleware zur Unterstützung dynamisch verteilt ausgeführter Prozesse zur Verfügung. Dabei können die vorgeschlagenen Komponenten prinzipiell sowohl einzeln in Hinblick auf ein bestimmtes Ziel der Flexibilisierung als auch gemeinsam zur Unterstützung komplexer Anpassungsvorgänge eingesetzt werden. Die vorgeschlagenen Middleware-Komponenten stellen dabei jeweils eine potentielle *Ergänzung* bestehender Prozessmanagementsysteme und somit die Basis zur Anwendung der beschriebenen Konzepte auf konkrete Technologien zum Prozessmanagement in Kapitel 8 dar.

Die folgenden Abschnitte fassen die Architektur der (ergänzenden) Middleware für dynamisch verteilt ausgeführte Prozesse zusammen. Abschnitt 7.1 gibt einen Überblick über die Komponenten der dienstorientierte Middleware

und deren Einbettung in bestehende Infrastrukturen zum Prozess-, Kontext- und Ereignismanagement. Es folgt eine Betrachtung der einzelnen Komponenten von einer grundlegenden Kommunikationsinfrastruktur zur Unterstützung heterogener mobiler Ausführungseinheiten (vgl. Abschnitt 7.2) bis hin zu den bereitgestellten Diensten für die Repräsentation des Prozessmanagementsystems als verwaltbare Ressource (vgl. Abschnitt 7.3), für die Automatisierung von Anpassungsmaßnahmen (vgl. Abschnitt 7.4), für die Migration von Prozessinstanzen (vgl. Abschnitt 7.5) und für die kontextbasierte Repräsentation von Benutzungsschnittstellen (vgl. Abschnitt 7.6). Im Anschluss wird die Anbindung eines bestehenden Kontextmanagementsystems vorgestellt, welches auch die Integration der Kontextdatenprognose zur proaktiven Anpassung der Verteilung prozessorientierter Anwendungen erlaubt (vgl. Abschnitt 7.7). Das Kapitel schließt in Abschnitt 7.8 mit einer kurzen Zusammenfassung der Ergebnisse.

7.1 Übersicht realisierter Middlewarekomponenten

Entsprechend der in Kapitel 6 vorgestellten Teilansätze zeigt Abbildung 7.1 einen groben Überblick über die komponentenbasierte Architektur der vorgeschlagenen Middleware zur Flexibilisierung verteilter Prozessausführung. Im Zentrum steht die Kapselung von Prozessmanagementsystemen als *verwaltbare Ressourcen*, welche jeweils Schnittstellen zu deren Überwachung und Steuerung anbieten (*Management-Schnittstelle*). Diese können von lokalen oder entfernten Anwendungen genutzt und bei Bedarf zu höherwertigen Diensten komponiert werden. Ganz oder teilweise darauf aufbauend werden hierüber Operationen zur Verteilung von laufenden Prozessinstanzen mittels Migration des Prozessmodells und des Ausführungszustandes realisiert (*Migrationsdienst*). Durch die Integration von Kontextmanagementsystemen und Systemen zur Ereignisverarbeitung können ergänzende Funktionen wie die Berechnung von Prognosen über zukünftige Ausführungskontexte (*Prognosesystem*) oder ein automatisches Management entfernt ausgeführter Prozesse über vordefinierte Regeln zur Ableitung von Anpassungsbedarf realisiert werden (*Managementdienst*). Für die Umsetzung der abstrakten Benutzungsschnittstellen wird ein *Interaktionsdienst* vorgeschlagen, welcher durch die Ausführungsumgebung analog zu den funktionalen Diensten als Ressource der Prozessausführung aufgerufen werden kann.

Die Dienste zur Flexibilisierung verteilter Prozessausführung kapseln innerhalb der prototypischen Realisierung jeweils eine weitestgehend plattformunabhängige Referenzimplementierung zur Anbindung konkreter Prozessmanagementsysteme, können jedoch auch vollständig plattformabhängig implementiert werden, solange ihre öffentlichen Schnittstellen in einer vorgegebenen Syntax und ihre Funktionalitäten in der vorgegebenen Semantik erhalten bleiben. Somit können prinzipiell die technischen *Process-Management-as-a-Service*-Funktionalitäten der hier vorgeschlagenen Middleware-Komponenten

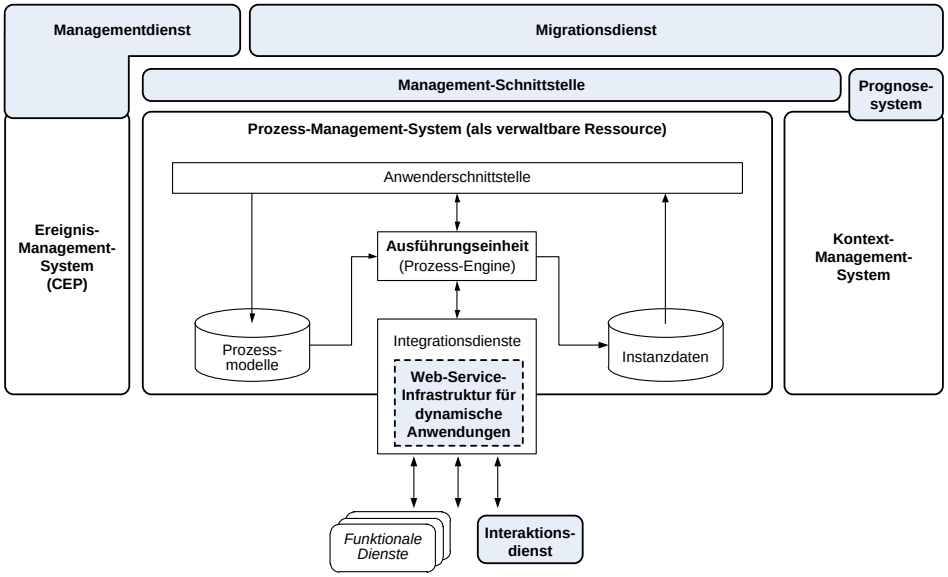


Abbildung 7.1: Überblick über die Middleware-Komponenten zur Flexibilisierung verteilter Prozessausführung auf Basis der idealtypischen Architektur eines Prozessmanagementsystems (vgl. Abbildung 2.17)

analog zu funktionalen Diensten unter Verwendung derselben Technologien angeboten und genutzt werden.

Damit eine dynamische Auswahl und ein dynamisches Einbinden von (funktionalen und/oder technischen) Diensten zur Laufzeit von prozessorientierten Anwendungen jedoch überhaupt möglich ist, muss eine Basisinfrastruktur zur Kommunikation zwischen den beteiligten Ausführungseinheiten bzw. zwischen den Ausführungseinheiten und den jeweils lokal oder global erreichbaren Ressourcen vorhanden sein. Die hier vorgestellte Web-Service-Infrastruktur für die Integration mobiler Systeme dient dazu, einen möglichst großen Teil mobiler Dienste und Ausführungseinheiten für die Ausführung dynamisch verteilt ausgeführter Prozesse nutzbar zu machen. Für eine Anwendung im rein stationären Bereich wird diese Kommunikationsinfrastruktur nicht zwingend benötigt bzw. kann durch andere Ansätze ersetzt oder ergänzt werden. Da ohne eine Möglichkeit zur dynamischen Bindung von Diensten das hier vorgeschlagene Konzept zur Flexibilisierung verteilter Prozessausführung nicht ohne weiteres anwendbar ist, soll im Folgenden zunächst eine exemplarische Umsetzung dieser Infrastruktur als Nachweis der Erfüllbarkeit dieser Grundvoraussetzung vorgestellt werden.

7.2 Dienstorientierte Architektur für dynamische Anwendungen

In einem *Process-Management-as-a-Service*-Szenario ist ebenso wie im Fall einer Dienstkomposition die unterliegende Prozess-Engine sowohl Konsument als auch Anbieter von Diensten. Zum einen werden Dienste als Ressour-

cen zur Durchführung von Aktivitäten der auszuführenden Prozesse aufgerufen und so deren Funktionalität bei der Bearbeitung des Prozesses konsumiert. Zum anderen stellt die Bereitstellung von Funktionalitäten der Prozessausführung mit den ggf. dazugehörigen Managementdiensten ein Anbieten von Diensten dar, welche veröffentlicht, von potentiellen Dienstnutzern aufgefunden und aufgerufen werden können. Gerade in dynamischen Umgebungen – wie zum Beispiel in mobilen (Ad-hoc-) Netzwerken – ist dabei ein Auffinden von Diensten während der Ausführung einzelner Prozessinstanzen und die Möglichkeit zur Einbindung der zur Laufzeit aufgefundenen Dienste besonders wichtig, da nur so zur Entwicklungszeit noch unbekannte Dienstinstanzen zur Erfüllung einer im Prozess benötigten Funktionalität zur Laufzeit integriert werden können. Dies setzt unter anderem die Möglichkeit zur abstrakten Beschreibung von relevanten Diensten und deren Auflösung zur Laufzeit in Abgleich mit den in der lokalen Ausführungsumgebung tatsächlich vorhandenen Diensten voraus. Oft werden im mobilen Bereich zudem andere Technologien verwendet als vom Web-Services-Protokollstapel standardmäßig vorgegeben werden (vgl. Abschnitt 3.4.3), was die Kompatibilität und Interoperabilität mit stationären dienstorientierten Anwendungen stark erschwert. Als Konsequenz ist es oft nicht möglich, heterogene Dienste dynamisch zu integrieren, wie es insbesondere für die Ausführung verteilt ausgeführter Prozesse notwendig ist [ZDL09a, ZDL09b].

Basierend auf dem Konzept der abstrakten Dienstklassen [KZL07a, Kun08] (vgl. auch Abschnitt 6.2.3) und eines verteilten Verzeichnisdienstes [Kun08, ZKL09a] wird im Folgenden zunächst eine für mobile und stationäre Ausführungseinheiten geeignete Architektur zum dynamischen Anbieten und Konsumieren von Web Services beschrieben (vgl. Abschnitt 7.2.1). Im zweiten Teil wird die im Rahmen dieser Arbeit implementierte Beispielkonfiguration vorgestellt (vgl. Abschnitt 7.2.2).

7.2.1 Komponenten für Dienstanbieter und Dienstkonsumenten

Abbildung 7.2 zeigt einen groben Überblick über die hier zugrunde gelegte dienstorientierte Architektur für die Realisierung einer dynamischen Interaktion zwischen mobilen und stationären Dienstanbietern und -konsumenten. Im Gegensatz zu der klassischen dienstorientierte Architektur mit einem zentralen Verzeichnisdienst und der Realisierung durch Standard-Web-Service-Protokolle (vgl. Abschnitte 3.4.2 und 3.4.3) werden die speziellen Anforderungen mobiler Systeme hier durch eine dezentrale Architektur und die Berücksichtigung relevanter mobiler Web-Service-Protokolle adressiert. Jeder teilnehmende Akteur kann dabei nach dem *Peer-to-Peer*-Prinzip als Anbieter oder Konsument von Diensten auftreten und besitzt dazu einen eigenen lokalen Verzeichnisdienst. Nimmt der betrachtete Akteur die Rolle eines Dienstanbieters an, so dient der lokale Verzeichnisdienst zur Verwaltung der Dienstbeschreibungen der durch diesen Akteur aktuell angebotenen Dienstinstanzen. Nimmt der Akteur die Rolle eines Dienstkonsumenten an, so ist der lokale

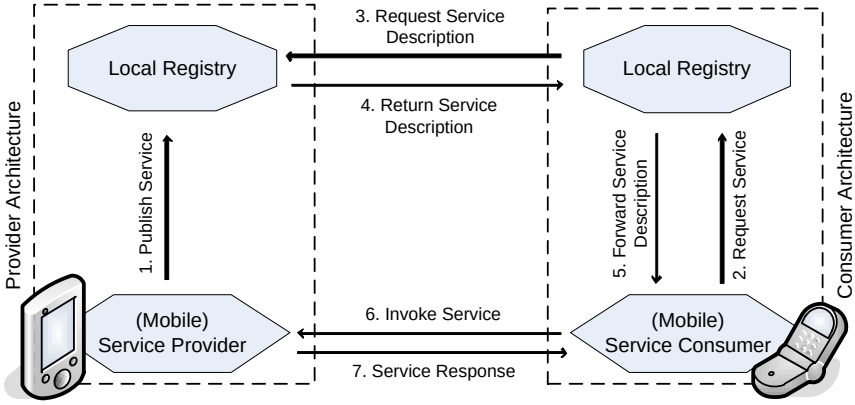


Abbildung 7.2: Dezentrale Dienstarchitektur für die dynamische Interaktion zwischen mobilen und/oder stationären Dienstanbietern und -konsumenten (ZDL09a, ZDL09b)

Verzeichnisdienst für die Suche nach den aktuell benötigten Dienstinstanzen verantwortlich, indem eine Kommunikation mit den Verzeichnisdiensten der anderen Akteure in der (mobilen) Umgebung stattfindet. Da sich der lokale Verzeichnisdienst in diesem Fall wie ein Proxy verhält, können auf diese Weise auch zentrale Verzeichnisdienste (z. B. basierend auf *UDDI*) oder andere verteilte Verzeichnisdienste (wie z. B. *Konark* [HDVL03]) integriert werden, sofern sie sich in Kommunikationsreichweite des Dienstkonsumenten befinden [Dre08, ZDL09a, ZDL09b].

Die Interaktion zwischen einer Anwendung als Dienstanbieter (*Service Provider*) und einer Anwendung als Dienstkonsument (*Service Consumer*) erfolgt wie in Abbildung 7.2 dargestellt. Sofern der Dienstanbieter wenigstens einen Dienst bei seinem lokalen Verzeichnisdienst bekannt gemacht hat (Schritt 1), kann dieser durch einen Dienstkonsumenten dynamisch aufgefunden werden. Die konsumierende Anwendung stellt dazu eine Anfrage an den eigenen lokalen Verzeichnisdienst (Schritt 2). Wird der angefragte Dienst durch die lokale Ausführungseinheit angeboten, so wird direkt die entsprechende Dienstbeschreibung (*Service Description*) zurückgeliefert (Schritt 5). In diesem Fall sind Dienstanbieter und Dienstkonsument identisch bzw. befinden sich in derselben Ausführungsumgebung, z. B. auf demselben mobilen Gerät. Ist dies nicht der Fall, so wird der gesuchte Dienst bei den Verzeichnisdiensten anderer Ausführungsumgebungen angefragt (Schritt 3, vgl. [Kun08, ZKL09a] zu Details). Wird ein passender Dienst gefunden, so wird die Dienstbeschreibung an den anfragenden Verzeichnisdienst (Schritt 4) und darüber an die anfragende Anwendung (Schritt 5) zurückgeliefert. Auf Basis der Dienstbeschreibung kann die konsumierende Anwendung nun analog zur klassischen dienstorientierten Architektur den Dienst beim Dienstanbieter aufrufen (Schritte 6 und 7) [Dre08, ZDL09a, ZDL09b].

Die detaillierte Architektur zur Realisierung von Dienstanbieter und/oder -konsument ist modular aufgebaut, wobei Dienstanbieter und Dienstkonsument grundsätzlich jeweils über dieselben Komponenten verfügen (vgl. Abbildung 7.3). Dies hat den Vorteil, dass grundlegende Funktionalitäten

zur Kommunikation, zur Nachrichtenverarbeitung und zur Interaktion mit dem Verzeichnisdienst nur einmal bereitgestellt werden müssen und von beiden Rollen gemeinsam genutzt werden können (*Shared Components*). Dienstanbieter greifen zusätzlich auf eine Laufzeitumgebung zu, welche die angebotenen Dienstinstanzen verwaltet (*Components of Service Providers*), während Dienstkonsumenten einen Proxy-Generator verwenden, um dynamisch Proxy-Komponenten als Repräsentation lokaler Schnittstellen von (entfernten) Diensten zu erzeugen und um diese schließlich darüber aufzurufen (*Components of Service Consumers*) [Dre08, ZDL09a, ZDL09b]. Sofern nicht beide Rollen benötigt werden, kann die Architektur im Einzelfall um die Komponenten des Dienstanbieters bzw. des Dienstkonsumenten reduziert werden. Da eine Prozess-Engine im Kontext von *Process-Management-as-a-Service*-Szenarien sowohl als Anbieter als auch als Konsument von Diensten auftreten kann, werden im Rahmen dieser Arbeit alle Komponenten der dargestellten Architektur verwendet.

Abhängig von der Konfiguration und den Eigenschaften konkreter (mobiler oder stationärer) Systeme kann diese (bis hierhin abstrakte) Architektur durch die Einbindung von speziellen Adapterkomponenten auf bestimmte Technologien für die Dienstbeschreibung, das Nachrichtenformat und das Kommunikationsprotokoll für den Transport der Nachrichten angepasst werden. Um Kompatibilität zu stationären dienstorientierten Architekturen mit Web Services zu erlangen, kann für den lokalen Verzeichnisdienst und für die Erzeugung von Proxy-Komponenten ein WSDL-Adapter, für die Nachrichtenverarbeitung ein SOAP-Adapter und für die Kommunikation ein HTTP-Adapter integriert werden. Um eine Anpassung auf Technologien zu erreichen, welche im Bereich mobiler Systeme verbreitet sind, kann zum Beispiel eine alternative oder ergänzende Konfiguration aus WSDL, ASN.1 und einer Kommunikation über Bluetooth oder über ein Overlay-Netzwerk realisiert werden [Dre08, ZDL09a, ZDL09b]. Eine entsprechend realisierte Beispielkonfiguration wird in Abschnitt 7.2.2 vorgestellt.

Die Installation und Konfiguration von Adapterkomponenten erfolgt zur Entwicklungszeit der dienstanbietenden bzw. -konsumierenden Anwendungen. Sobald ein neuer Dienst veröffentlicht werden soll (Schritt 1 in den Abbildungen 7.2 und 7.3), kann im Rahmen des Deployments eine Anpassung der Dienstbeschreibung auf die verfügbaren Technologien und Protokolle vorgenommen werden. Stehen mehrere alternative Konfigurationen zur Verfügung, so kann ein Dienst prinzipiell über eine beliebige Kombination der unterstützten Technologien angeboten werden. Dabei muss der Verzeichnisdienst nur dann mehrere Beschreibungen des Dienstes verwalten, wenn auch unterschiedliche Beschreibungssprachen (z. B. ergänzend zu WSDL) unterstützt werden sollen. Das Anbieten des Dienstes ist mit der Übergabe der Dienstbeschreibung(en) an den lokalen Verzeichnisdienst abgeschlossen [Dre08, ZDL09a, ZDL09b].

Ein (potentieller) Dienstkonsument stellt zur dynamischen Einbindung eines Dienstes zunächst eine abstrakte Dienstanfrage an den lokalen Ver-

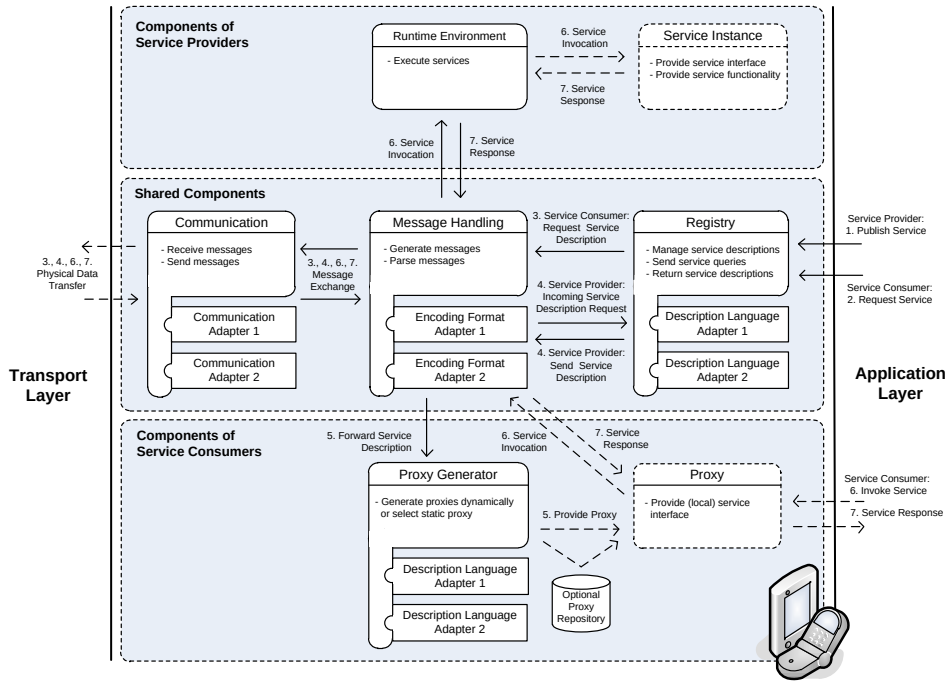


Abbildung 7.3: Komponentenmodell der Dienstarchitektur für Dienstanbieter und Dienstnutzer(ZDL09a, ZDL09b)

zeichnisdienst (Schritt 2). Diese Dienstanfrage enthält die durch die Anwendung spezifizierten Parameter, welche eine Identifikation des gesuchten Dienstes hinsichtlich seiner Funktionalität und optional der gewünschten nicht-funktionalen Eigenschaften erlauben. Die Funktionalität kann dabei zum Beispiel auf Basis eines eindeutigen Identifikators wie eines *Uniform Resource Identifier (URI)*, eines *Universally Unique Identifier (UUID)* (vgl. auch [Kun08, ZKL09b]) oder einer semantischen Beschreibung auf Basis von *Resource Description Framework (RDF)* erfolgen. Für die Integration nicht-funktionaler Aspekte können dabei zusätzliche Beschreibungen verwendet werden (vgl. [HSZ11] zu Details), welche auch die Spezifikation unterstützter oder bevorzugter Nachrichtenformate und/oder Transportprotokolle der hier dargestellten Architektur enthalten können.

Sofern dies mit den nicht-funktionalen Rahmenbedingungen vereinbar ist, überprüft der lokale Verzeichnisdienst zunächst, ob der gesuchte Dienst lokal angeboten wird oder ob sich eine passende Dienstbeschreibung aus früheren Suchanfragen noch im Cache des Verzeichnisdienstes befindet. Ansonsten wird die Anfrage an andere, über die zur Verfügung stehenden Kommunikationsprotokolle erreichbare Verzeichnisdienste weitergeleitet (Schritt 3) [Dre08, ZDL09a, ZDL09b]. Abbildung 7.4 zeigt den groben Aufbau des lokalen Verzeichnisdienstes und die beschriebene Vorgehensweise dieses Schrittes im Detail.

Wurde die Anfrage an andere Verzeichnisdienste weitergeleitet und von einem (potentiellen) Dienstanbieter empfangen, so wird geprüft, ob die für die

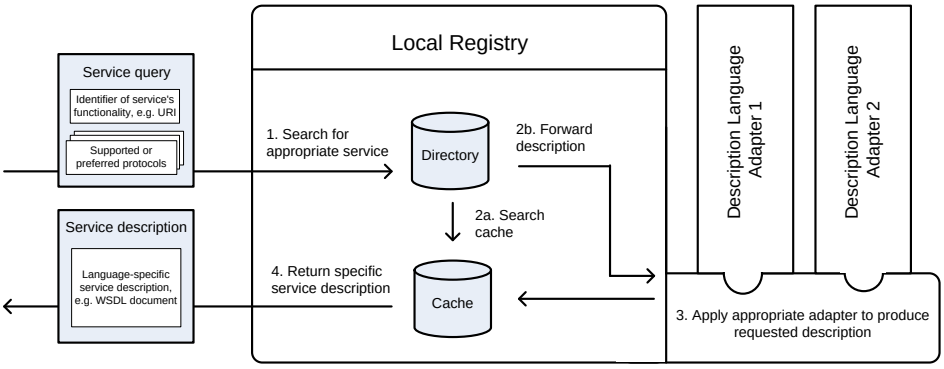


Abbildung 7.4: Bearbeitung von Dienstanfragen durch den lokalen Verzeichnisdienst (Dre08, ZDL09a, ZDL09b)

Bearbeitung der Dienstanfrage benötigten Protokolle in der lokalen Konfiguration zur Verfügung stehen. Im positiven Fall wird die Anfrage an den lokalen Verzeichnisdienst zugestellt, welcher eine passende Dienstbeschreibung der angefragten Dienstinstanz zurückliefert (Schritt 4). Die Dienstbeschreibung wird über die spezifizierten Kommunikationswege an den Dienstkonsumenten gesendet und intern an den Proxy-Generator weitergeleitet (Schritt 5). Abhängig von der konkreten Implementierung kann basierend auf der Dienstbeschreibung entweder automatisch ein neuer Proxy erzeugt werden, oder es kann ein bereits bestehender (statischer) Proxy aus einem Proxy-Repository instantiiert werden. Basierend auf der Zuweisung einer Referenz auf das erzeugte Proxy-Objekt als Repräsentation der entfernten Schnittstelle kann die konsumierende Anwendung nun den gewünschten Dienst aufrufen (Schritt 6). Dabei verwendet der Proxy das in der Dienstbeschreibung vorgegebene Nachrichtenformat und Kommunikationsprotokoll, um die Eingabeparameter zuzuweisen, den Aufruf zu versenden und etwaige Rückgabewerte entgegen zu nehmen sowie diese der Anwendung zur Verfügung zu stellen (Schritt 7). Optional kann die Proxy-Komponente dem Proxy-Repository hinzugefügt werden, um für spätere Dienstaufrufe zur Verfügung zu stehen. Ist der Dienst in der Zwischenzeit nicht mehr verfügbar, so wird die Dienstbeschreibung aus dem Cache des Verzeichnisdienstes gelöscht und eine neue Suchanfrage gestartet [Dre08, ZDL09a, ZDL09b].

Die hier vorgestellte Architektur kann prinzipiell für beliebige dienstanbietende und/oder dienstkonsumierende Anwendungen zugrunde gelegt werden (vgl. [Dre08, ZDL09a, ZDL09b]). Die Art und Anzahl der realisierten Adapter-Komponenten kann flexibel an die Leistungsfähigkeit und die speziellen Eigenschaften der jeweils betrachteten mobilen und stationären Ausführungseinheiten angepasst werden. Je mehr Technologien zur Verfügung stehen, um Dienste in der erreichbaren Umgebung aufzufinden und einzubinden, desto höher ist die Wahrscheinlichkeit, dass ein passender Dienst gefunden werden kann. Die hier vorgestellte Architektur trägt daher zur Interoperabilität zwischen heterogenen Ausführungseinheiten, zur dynami-

schen Ressourcenbindung und somit zur Flexibilisierung der verteilten Prozessausführung auf technischer Ebene bei.

7.2.2 Beispielkonfiguration zur Integration mobiler Systeme

Der Standard-Protokollstapel aus WSDL, SOAP und HTTP ist nur bedingt für das Anbieten und Nutzen von Diensten im Kontext mobiler Systeme geeignet. Zum einen wird durch die Synchronizität von HTTP der Umgang mit Abbrüchen der Netzwerkverbindung erschwert und typische mobile Technologien zur Datenübertragung (wie z. B. Bluetooth) können nicht (ohne Weiteres) unterstützt werden. Zum anderen resultiert der XML-basierte Nachrichtenaustausch aufgrund der geringen Informationsdichte von XML-Beschreibungen im Allgemeinen in einer unnötig hohen Belastung mobiler Netzwerke. Es wurden daher verschiedene alternative Ansätze und Protokolle geschaffen (vgl. z. B. [BMNR03, TVN+04, ABCR06, Oh06]), welche flexibel in die hier vorgestellte Architektur eingebunden werden können. Werden jedoch alle Ebenen des Standard-Protokollstapels durch alternative Technologien ersetzt, so besteht die Gefahr, dass die Interoperabilität mit den in stationären Systemen angebotenen bzw. konsumierten Web Services verloren geht. In diesem Abschnitt wird daher eine Beispielkonfiguration vorgestellt, welche die Interoperabilität zu bestehenden dienstorientierten Architekturen auf der Basis von Web Services erhält und gleichzeitig beispielhaft die Integration mobiler Dienstanbieter und -konsumenten erlaubt.

Abbildung 7.5 zeigt die in Abschnitt 7.2.1 abstrakt vorgestellte Architektur mit einer Prozess-Engine, welche sowohl Dienste in der Ausführungsumgebung als Ressourcen für die Prozessausführung konsumiert als auch die durch Prozesse aggregierten Funktionalitäten wieder als (komplexe) Dienste anbieten kann. Zudem kann im Rahmen des *Process-Management-as-a-Service* die Funktionalität des Prozessmanagements für die dynamische Verteilung von Prozessinstanzen und deren Überwachung und Steuerung als Dienst angeboten werden (vgl. Abschnitte 7.3, 7.4 und 7.5). Für das Anbieten und Konsumieren von Standard-Web-Services steht dabei die Konfiguration aus WSDL, SOAP und HTTP zur Verfügung. Für die zusätzliche Unterstützung mobiler Dienstanbieter und -konsumenten kann z. B. auf *ASN.1* (*Abstract Syntax Notation Number One*) mit DER-Encoding [IT04] zurückgegriffen werden, was eine Reduktion der Nachrichtengröße von bis zu 90% erlaubt (vgl. [ZDL09a, ZDL09b] zu weiteren Details). Als Beispiel für alternative Ansätze zur Datenübertragung wurde die direkte Kommunikation über TCP/IP und die Nutzung eines bestehenden Overlay-Netzwerks realisiert (siehe unten). Gemeinsam ist beiden Teilkonfigurationen nur die Beschreibung von Diensten im WSDL-Format. Dies ist vorteilhaft, da WSDL standardmäßig innerhalb des *Binding*-Elements die Integration beliebiger Nachrichtenformate und Kommunikationsprotokolle erlaubt und somit die Interoperabilität zwischen stationären und mobilen Diensten gewährleistet. Das Übertragen der WSDL-Datei verursacht dabei zwar prinzipiell einen relativ

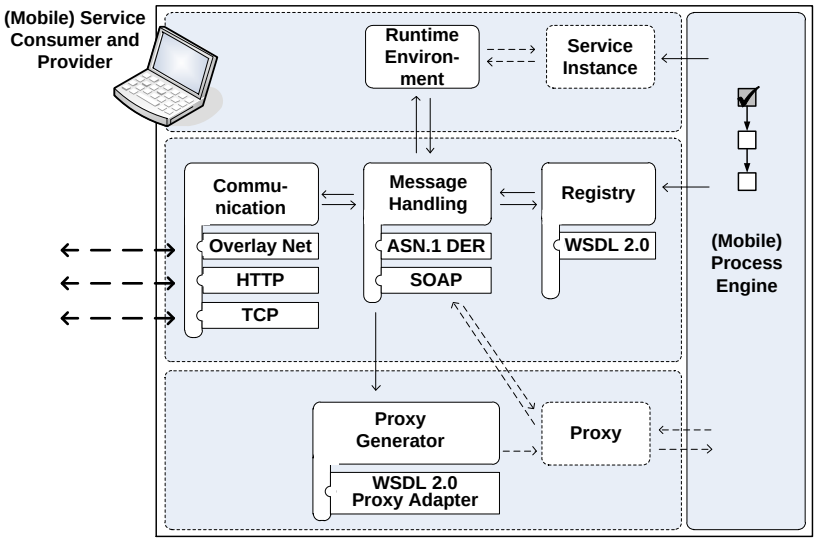


Abbildung 7.5: Beispielkonfiguration für eine (mobile) Prozess-Engine (ZDL09a, ZDL09b)

hohen Übertragungsaufwand, welcher aber für mehrfache Dienstaufrufe in der Regel nur einmalig erfolgen muss [ZDL09a, ZDL09b].

Desweiteren muss ein geeignetes Konzept zum Umgang mit wechselnden Netzwerkadressen in mobilen Umgebungen zugrunde gelegt werden. Dabei muss es insbesondere ermöglicht werden, dass die oben genannten lokalen Verzeichnisdienste eine Möglichkeit besitzen, einander dynamisch aufzufinden. Als Grundlage hierfür wird ein bestehendes leichtgewichtiges Overlay-Netzwerk verwendet (vgl. [Kun05, Kun08]), welches das dynamische Beitreten bzw. Verlassen beliebiger Verzeichnisdienste erlaubt. Dabei wird jeder teilnehmenden Ausführungseinheit mit eigenem Verzeichnisdienst ein eindeutiger Identifikator (in Form einer UUID) zugewiesen, welche zusammen mit einem gemeinsamen Schlüssel (Netz-ID) und der (physikalischen) Adresse des Verzeichnisdienstes in periodischen Abständen über die zur Verfügung stehenden Netzwerkadapter versendet wird. In Folge besitzen alle potentiell an der Prozessausführung teilnehmenden Ausführungseinheiten grundlegendes Wissen über die in der betrachteten Umgebung vorhandenen Systeme und können diese bei Bedarf in eine Dienstanfrage einbeziehen. Das Overlay-Netzwerk steht dabei zudem als zusätzliches Medium für Dienstaufrufe zur Verfügung, sofern dies in der Dienstbeschreibung berücksichtigt wurde.

Abbildung 7.6 zeigt abschließend eine mögliche Interaktion der Prozess-Engine (Gerät 3) über die bereitgestellte Konfiguration. Dabei können durch die Prozess-Engine sowohl Dienste von leistungsbeschränkten mobilen Dienstanbietern (Gerät 1) als auch von (stationären) Dienstanbietern mit Standard-Web-Service-Protokollen (Gerät 2) konsumiert werden. Ebenfalls können die durch die Prozess-Engine angebotenen Dienste von Konsumenten mit einer kompatiblen Konfiguration (Gerät 4) dynamisch eingebunden werden. Geräte mit einer zu geringen oder nicht kompatiblen Menge von Adapter-Komponenten können in diesem Szenario nicht unterstützt werden

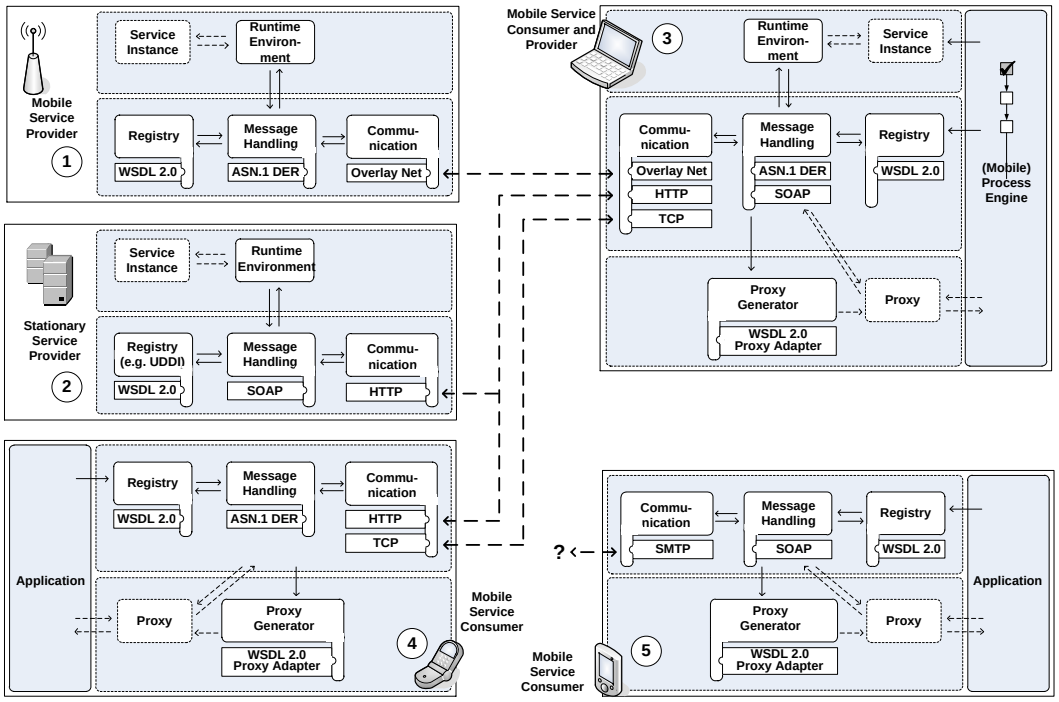


Abbildung 7.6: Interaktion mit heterogenen Dienstanbietern und -konsumenten (ZDL09a, ZDL09b)

(Gerät 5), sind aber aufgrund ihrer Mobilität und der bereitgestellten Infrastruktur unter Umständen in der Lage, die gesuchten Dienste in anderen Ausführungsumgebungen aufzufinden [ZDL09a, ZDL09b].

7.3 Prozessmanagementsystem als verwaltbare Ressource

Um die Modifikation von einzelnen Prozessinstanzen für deren (entfernte) Überwachung und Steuerung zu vermeiden und aufbauend auf grundlegenden Informations- und Steuerungsfunktionalitäten auch komplexere Anpassungen der Prozessausführung zu ermöglichen, wurde in Abschnitt 6.3 die Kapselung und Repräsentation von Prozessmanagementsystemen als *verwaltbare Ressourcen* vorgeschlagen. Dabei wurde der Gedanke, die speziellen Eigenschaften beliebiger Objekte als verwaltbare Ressourcen über eine dienstbasierte Schnittstelle anzubieten, bereits durch das *Web Services Distributed Management* (*WSDM*) aufgegriffen (vgl. Abschnitt 3.8.3). Voraussetzung für die Anwendung von WSDM ist ein Modell der zu verwaltenden Ressource als gemeinsames Verständnis der im Rahmen von WSDM ausgetauschten Informationen und Steuerungsanweisungen. Durch WSDM selbst wird jedoch lediglich ein grundlegendes Modell zum Management von Web Services (*Management of Web Services*, kurz *MOWS*) als verwaltbare Ressourcen spezifiziert [WS06]. Auf der Grundlage der in Kapitel 2 erarbeiteten Eigenschaften von Prozessmanagementsystemen im Allgemeinen und unter spezieller Berücksichtigung der in Kapitel 4 identifizierten Anforderungen an eine dynamisch verteilte

Ausführung wurde daher in Abschnitt 6.3.2 ein Referenzmodell eines Prozessmanagementsystems als verwaltbare Ressource vorgestellt. In Zusammenhang mit der Spezifikation zum Anbieten einer verwaltbaren Ressource über eine Web-Service-Schnittstelle (*Management using Web Services*, kurz *MUWS*) [BV06a, BV06b] können daher nun auf Basis von WSDM prinzipiell beliebige auf dem Referenzmodell basierende Prozessmanagementsysteme als verwaltbare Ressourcen abgebildet werden.

Zunächst wird in diesem Abschnitt eine mögliche Umsetzung der grundlegenden Informations- und Steuerungsfunktionalitäten eines Prozessmanagementsystems mittels WSDM vorgestellt (vgl. Abschnitt 7.3.1). Im Anschluss wird eine Auswahl inhaltlicher Funktionalitäten zur Verwaltung von Prozessmanagementsystemen beschrieben, welche sich auf das vorgeschlagene Referenzmodell begründen (vgl. Abschnitt 7.3.2). Da für die Bereitstellung und Manipulation von Eigenschaften eines Prozessmanagementsystems Anpassungen an selbigem notwendig sind, wird schließlich eine mögliche Integration der genannten Funktionalitäten in das Prozessmanagementsystem diskutiert (vgl. Abschnitt 7.3.3). Die hier spezifizierten Funktionen stellen zudem die Grundlage für höherwertige Dienste für die Unterstützung verteilter Prozessausführung dar, welche in den nachfolgenden Abschnitten 7.4 und 7.5 genauer beschrieben werden.

7.3.1 Abbildung mit WSDM

Die Repräsentation eines Prozessmanagementsystems als verwaltbare Ressource mit WSDM erlaubt es, die im Rahmen des Referenzmodells festgelegten Elemente und Eigenschaften über eine standardisierte Web-Service-Schnittstelle abzufragen oder zu manipulieren und somit die dadurch bereitgestellten Management-Funktionalitäten über Verzeichnisdienste in bestehende oder zukünftige dienstorientierte Architekturen und Anwendungen zu integrieren. Dabei können die seitens WSDM bestehende Rahmenfunktionen und die Möglichkeiten zur Beschreibung von Meta-Eigenschaften für die Abbildung von Prozessmanagementsystemen als verwaltbare Ressourcen weiterverwendet werden. Dieser Abschnitt beschreibt die Einbettung des Referenzmodells aus Abschnitt 6.3.2 in das bestehende Rahmenwerk von WSDM.

Abbildung 7.7 zeigt das grundlegende Modell von WSDM (vgl. Abbildung 3.23 in Abschnitt 3.8.3) in Hinblick auf die Kapselung eines Prozessmanagementsystems als verwaltbare Ressource. Dabei kann ein potentieller Konsument der Management-Funktionalität (*Manageability Consumer*) über ein *Discovery*-Verfahren (z. B. über einen Verzeichnisdienst) die Dienstbeschreibung (*WSDL*) und darüber die Adresse des Dienstzugriffspunktes (*Endpoint Reference*, kurz *ERP* [BCC+04]) des gekapselten Prozessmanagementsystems in Erfahrung bringen. Auf Basis dieser Information können nun weitere Informationen zur dienstbasierten Management-Schnittstelle abgefragt werden, fachliche Werte der im Referenzmodell zugrunde gelegten Eigenschaften abgerufen bzw. modifiziert werden oder Ereignisse dieses Prozessmanagement-

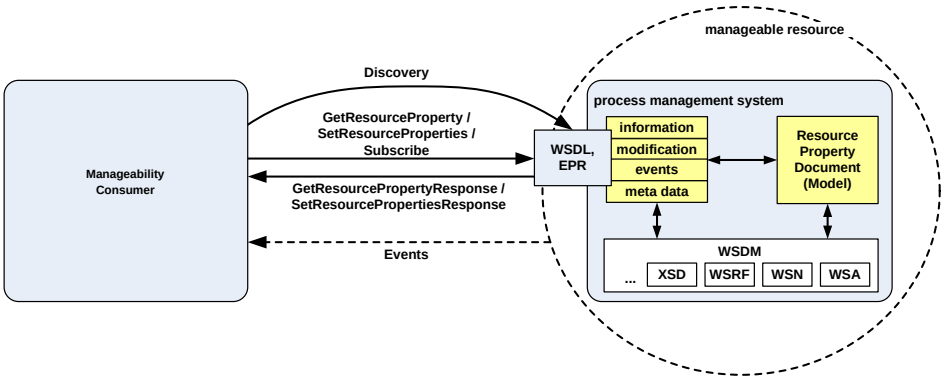


Abbildung 7.7: Umsetzung einer Prozess-Engine als verwaltbare Ressource im Rahmen des Web Services Distributed Management (WSDM) (BV06a, ZBH+10)

systems abonniert werden. Grundlage dieser Funktionalität ist die im Rahmenwerk von WSDM zentrale Ressourcenbeschreibung (*Resource Property Document*), welche das Modell der zu verwaltenden Ressource in einer XML-Repräsentation enthält. Jede Eigenschaft dieses Modells wird dabei als sogenanntes *Resource Property* abgebildet. Alle die Ressource betreffenden Eigenschaften können damit über generische Operationen gemäß der Spezifikation *WS-Resource Properties* (*WSRP*) [GT06] abgefragt bzw. verändert werden. Dabei stehen im Kontext dieser Arbeit insbesondere die Operationen *GetResourceProperty* zur Abfrage von Informationen und *SetResourceProperties* zur Manipulation von Werten der definierten Eigenschaften im Vordergrund. Der Konsument der Management-Funktionalität erhält entsprechend seiner Anfrage eine *GetResourcePropertyResponse* bzw. eine *SetResourcePropertiesResponse* oder eine Fehlermeldung für den Fall, dass die Anfrage nicht erfolgreich ausgeführt werden konnte. Das Abonnement von Ereignissen wird über die Integration des Standards *WS-Notification* (*WSN*) [GHM06] mit den entsprechenden Operationen zur Verwaltung von Abonnements (insbesondere der Operation *Subscribe* zum Abonnieren eines Ereignisses) verwaltet. Die Art und Weise des Aufrufs der bereitgestellten Operationen sowie die zu erwartenden Fehlermeldungen werden entsprechend der inkludierten Standards im WSDL-Dokument der verwaltbaren Ressourcen spezifiziert [BV06a, BV06b, Wun09, ZBH+10].

Das *Resource Property Document* erlaubt die Abbildung des in Abschnitt 6.3.2 vorgestellten Referenzmodells als XML-Schema. Dabei können auch die dort spezifizierten hierarchischen Beziehungen übertragen werden. Entsprechend des Referenzmodells besteht die Ressource *Process Management System* daher aus den Subressourcen *Process Models*, *Process Instances* und *Process Histories* sowie auf der nächsten Ebene aus deren im Referenzmodell spezifizierten Entitäten und deren jeweiligen Eigenschaften (*Properties*). Zusätzlich können bei Bedarf die relevanten Entitäten des Kontextes als Subressourcen definiert werden. Eine exemplarische Umsetzung des Referenzmodells als

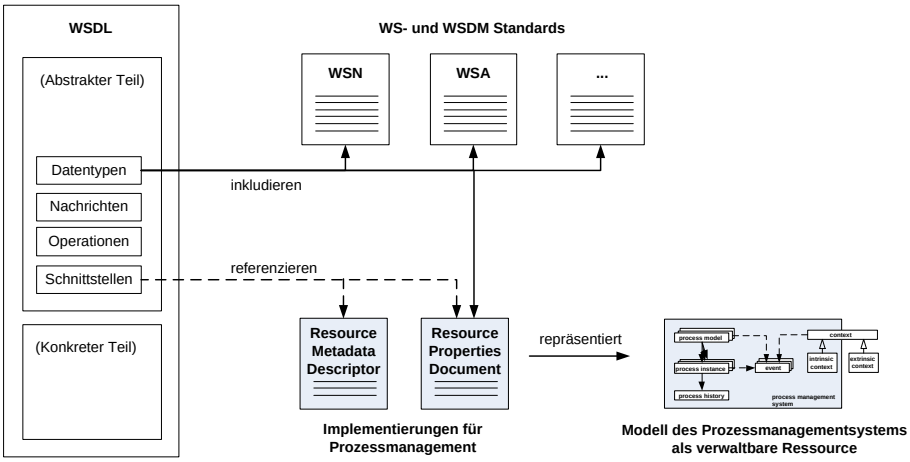


Abbildung 7.8: Zusammenspiel der verwendeten Standards und Ergänzungen im Rahmen dieser Arbeit (Wun09)

XML-Schema kann der Arbeit von WUNDERLICH [Wun09] entnommen werden.

Die im *Resource Property Document* beschriebenen Eigenschaften (*Properties*) können nun mit Hilfe von Metadaten genauer spezifiziert werden. Eine Grundlage hierfür stellt der zu WSRF gehörende Standard *WS-Resource Metadata* [Jem06] dar, welcher Inhalt und Aufbau der Beschreibung eines *Resource Property Document* innerhalb eines ergänzenden *Descriptor*-Dokuments regelt. Es enthält eine Liste von *Properties*, wobei jedem *Property* ein Wertebereich sowie die Attribute *Modifiability* (Änderbarkeit) und *Mutability* (Veränderlichkeit) zugeordnet werden können. Dem Attribut *Modifiability* werden die in Abschnitt 6.3.2 identifizierten Arten von Änderbarkeit (*Read*, *Read-Write* und *Read-Limited-Write*) zugeordnet. Dem Attribut *Mutability* werden die von WSDM definierten Arten von Veränderlichkeit (*Constant*, *Appendable* und *Mutable*) zugewiesen. Aufgrund der Relevanz für dynamische Umgebungen werden diese Attribute noch um die in Abschnitt 6.3.2 identifizierten Attribute *Mutability-Frequency* (Änderungshäufigkeit) mit seinen möglichen Werten (*Always*, *Seconds*, Minutes, *Hours*, *Days*, *Months*, *Years*, *Never* und *Unknown*) und *Availability* (Verfügbarkeit) mit seinen möglichen Werten (*Preparation*, *Wait*, *Execution*, *Postprocessing* und *Always*) ergänzt (vgl. Abschnitt 6.3.2). Dazu wird ein zusätzlicher Namensraum in den *WS-Resource Metadata*-Deskriptor integriert.

Abbildung 7.8 verdeutlicht die dargelegten Zusammenhänge zwischen den einzelnen WS*-Spezifikationen und dem in dieser Arbeit erarbeiteten Referenzmodell für Prozessmanagementsysteme graphisch. Die Trennung der Metadaten-Beschreibung von der Repräsentation des Modells erlaubt dabei, dass verschiedenen Ressourcen (d. h. in diesem Fall verschiedenen Prozessmanagementsystemen mit ihren Eigenschaften) ein gemeinsames Modell zugrunde gelegt werden kann, jedoch die Meta-Eigenschaften *Änderbarkeit*, *Veränderlichkeit*, *Änderungshäufigkeit* und *Verfügbarkeit* für jede Ressource

und ihre jeweils untergeordneten *Ressource Properties* individualisierbar sind. Sowohl die Werte der Meta-Eigenschaften als auch die Werte der *Ressource Properties* können über die WSDL-Schnittstelle der verwaltbaren Ressource zugegriffen werden.

7.3.2 Überwachungs- und Steuerungsfunktionalität

Die für die Überwachung und Steuerung verteilt ausgeführter Prozesse benötigte Management-Schnittstelle kann auf Basis des Referenzmodells, einer Zuordnung zu bereits durch WSDM bereitgestellten allgemeinen Managementfunktionalitäten, der durch *WS-Resource Properties* (*WSRP*) [GT06] definierten generischen Operationen sowie einer Reihe von anwendungsspezifischen Funktionalitäten zusammengesetzt werden. Ergänzend wird eine Ereignisschnittstelle mit *WS-Notification* (*WSN*) [GHM06] auf Basis der Definition von Themenbäumen mit *WS-Topics* [VGN06] umgesetzt. Für den Einsatz auf mobilen Geräten mit Leistungseinschränkungen, die den Einsatz des eher schwergewichtigen Rahmenwerks von WSDM und den inkludierten Web-Service-Standards unvorteilhaft oder unmöglich machen, wird eine Abbildung der resultierenden Web-Service-Schnittstelle auf äquivalente Funktionen der mobilen Ausführungsumgebung vorgenommen. Die Schnittstelle und die konzeptionellen Überlegungen können jedoch beiden Arten der Implementierung zugrunde liegen, so dass die angestrebte Interoperabilität zwischen verwaltbarer Ressource und Konsument der Management-Funktionalität auch hier weiterhin gegeben ist.

Standard-Manageability-Capabilities

Als Zusammenfassung von Eigenschaften, Operationen, Ereignissen und Metadaten sind durch WSDM bereits sogenannte *Manageability Capabilities* definiert, welche eine allgemeine Gruppierung von Managementfunktionalitäten einer verwaltbaren Ressource zur Verfügung stellen. Um einen WSDM-MUWS-kompatiblen Dienstzugriffspunkt zu definieren, muss dabei zumindest die Capability *Identität* (*Identity*) durch die verwaltbare Ressource angeboten werden (vgl. [BV06a]). Diese liefert auf Anfrage einen eindeutigen und unveränderbaren Identifikator der Ressource zurück (*RessourceID*) und ist Voraussetzung wichtiger anderer Funktionalitäten, z. B. zur Prüfung auf Gleichheit oder zur Korrelation. Die Liste der verwendbaren *WSDM Manageability Capabilities* ist zur Veranschaulichung in Tabelle 7.1 zusammengefasst. Dabei wird die Semantik der einzelnen Capabilities jeweils über eine eindeutige URI festgelegt (vgl. [BV06a, BV06a]).

Die Nutzung der definierten WSDM Manageability Capabilities ist (mit Ausnahme von *Identity*) nicht zwingend notwendig, erhöht jedoch durch die gemeinsame bereits festgelegte Semantik die Interoperabilität zwischen den verwaltbaren Ressourcen und den Konsumenten der Managementfunktionalität. Es sollte daher stets geprüft werden, welche Managementfunktionalitäten sich

WSDM Manageability Capabilities	Beschreibung
Identity	Eindeutiger, nicht veränderbarer bzw. nicht veränderlicher Identifikator der Ressource
Manageability characterisics	Liste der von der Ressource unterstützten Capabilities
Correlatable properties	Korrelationseigenschaften zur Prüfung auf Gleichheit zweier Ressourcen
Description	Beschreibung und Version der Ressource
State	Aktueller bzw. letzter Zustand der Ressource auf der Grundlage eines eigenen Zustandsmodells
Operational status	Abstrakter Zustand der Ressource hinsichtlich ihrer Verfügbarkeit
Metrics	Metrische Informationen zur Leistung und zum Betrieb der Ressource
Configuration	Eigenschaften, welche das Verhalten der Ressource beeinflussen und durch den Konsumenten der Managementfunktionalität verändert werden können
Relationships	Beziehungen, an denen die Ressource beteiligt ist
Relationship resource	Eigenschaften der Ressource, welche eine Beziehung ausdrücken bzw. repräsentieren
Advertisement	Benachrichtigungen im Fall der Erzeugung oder bei der Beendigung der Existenz von Ressourcen

Tabelle 7.1: WSDM Manageability Capabilities (BV06a, BV06a)

bereits den bestehenden Capabilities zuordnen lassen [BV06a, BV06a]. Dabei können die bestehenden Capabilities mit teilweise bestehenden Kombinationen von allgemeinen Eigenschaften verwaltbarer Ressourcen auch durch ressourcenspezifische Eigenschaften mit ihren Metadaten ergänzt werden. Die grundlegenden Eigenschaften des Prozessmanagementsystems, wie dessen Herstellerbezeichnung, die Version und die unterstützte(n) Prozessbeschreibungssprache(n), können zum Beispiel der Capability *Description* zugeordnet werden. Für die hierarchisch untergeordneten Prozessinstanzen kann zum Beispiel das entsprechende Zustandsmodell (vgl. Abschnitte 2.7.2 bzw. 6.2.2) auf die Capability *State* abgebildet werden. Ist keine geeignete Abbildung aller benötigten Managementfunktionalitäten bzw. Eigenschaften des Ressourcenmodells auf die genannten Capabilities möglich, können weitere eigene Capabilities definiert werden. Im Folgenden wird eine exemplarische Abbildung von Eigenschaften des Referenzmodells vorgenommen, welche mit den Standard-Operationen von *WS-Resource Properties* zugegriffen werden können. Die in Hinblick auf das Prozessmanagement spezifischen Managementfunktionalitäten werden anschließend unter der neuen Capability *Process Management* zusammengefasst.

Standard-Operationen von *WS-Resource Properties*

Der Abruf von Werten einer Eigenschaft bzw. die Änderung von Werten erfolgt auf Basis von *WS-Resource Properties* in Form von generischen Operationen, welchen als Parameter der Name der angeforderten Eigenschaft und ggf. die zu manipulierenden Werte übergeben werden. Als Ergebnis wird im Erfolgsfall die Antwort in dem im *Resource Property Document* spezifizierten Datenformat zurückgeliefert. Für die Verwendung mit WSDM stehen hierfür die Standard-Datentypen von XML-Schema zur Verfügung. Desweiteren können beliebige individuelle oder komplexe Datentypen definiert und den Eigenschaften zugeordnet werden. Im Fehlerfall können Fehlernachrichten geliefert wer-

Operation	Beschreibung
GetResourcePropertyDocument	Liefert das *Resource Property Document* bzw. alle dort enthaltenen Eigenschaften mit ihren Werten zurück
GetResourceProperty	Gibt den aktuellen Wert der angefragten Eigenschaft zurück
GetMultipleResourceProperties	Gibt die aktuellen Werte mehrerer angefragter Eigenschaften zurück
QueryResourceProperties	Gibt die Werte von Eigenschaften zurück, welche einem angegebenen Auswahlkriterium entsprechen
PutResourcePropertyDocument	Ersetzt das *Resource Property Document* bzw. alle Werte der dort enthaltenen Eigenschaften
SetResourceProperties	Modifiziert die angegebenen Eigenschaften bzw. deren Werte in der angegebenen Form (*Insert*, *Update* oder *Delete*)
InsertResourceProperties	Einfügen von neuen Werten einer Eigenschaft
UpdateResourceProperties	Ersetzen von bestehenden Werten einer Eigenschaft, welche von Konsumenten der Managementfunktionalität verändert werden können
DeleteResourceProperties	Löschen aller Werte einer Eigenschaft

Tabelle 7.2: Standard-Operationen von *WS-Resource Properties* (GT06)

den, welche weitere Informationen über den aufgetretenen Fehler enthalten. Ist eine angefragte Eigenschaft undefiniert, wird eine leere Antwortnachricht zurückgeliefert.

Um mehrere Eigenschaften mit einer Nachricht anzufragen bzw. zu bearbeiten, können Listenoperationen verwendet werden. Diese können entweder das gesamte *Resource Property Document* oder nur eine Teilmenge an Eigenschaften betreffen. Bei nicht verändernden Anfragen wird hierbei eine Liste von angeforderten Eigenschaften übergeben und entsprechend eine Liste mit den Werten der angefragten Eigenschaften zurückgeliefert. Bei verändernden Anfragen können analog mehrere Manipulationsoperationen in einer (komplexen) Änderungsoperation zusammengefasst werden. In Ergänzung zu einem reduzierten Übertragungsaufwand durch die Zusammenfassung zu einer Anfrage- bzw. Antwortnachricht ist hierdurch auch die Durchführung von transaktional zusammenhängenden Änderungsoperationen möglich. Es ist dabei jedoch generell nur die Änderung von Eigenschaften erlaubt, welche durch eine entsprechende Konfiguration der Änderbarkeitseigenschaft gekennzeichnet sind. Bei unerlaubtem Zugriff auf eine geschützte Eigenschaft wird eine Fehlermeldung zurückgegeben. Tabelle 7.2 zeigt die zur Verfügung stehenden Operationen in einer Übersicht.

Exemplarische Eigenschaften der verwaltbaren Ressource

Das in Abschnitt 6.3.2 dargestellte Modell des Prozessmanagementsystems verfügt über drei hierarchisch gegliederte Arten von Ressourcen, für welche prinzipiell beliebige individuelle Eigenschaften definiert werden können. Als Referenzkonfiguration wurde im Rahmen dieser Arbeit nur eine Auswahl von Eigenschaften definiert, welche insbesondere für die in Abschnitt 4.2 betrachteten Anwendungsfälle relevant sind und eine geeignete Grundlage für die Umsetzung der Migration laufender Prozessinstanzen zur Flexibilisierung der Verteilung darstellen. Die Eigenschaften sind zur Veranschau-

Eigenschaft (*Property*)	Beschreibung
Ressource ID	Obligatorischer Identifikator der Ressource in WSDM (Capability *Identity*)
Caption	Herstellerbezeichnung des Prozessmanagementsystems
Version	Software-Version des Prozessmanagementsystems
Description	Ausführliche menschenlesbare Beschreibung
Supported PDLs	Liste der unterstützten Prozessbeschreibungssprachen
Owner	Organisatorische Zugehörigkeit des Prozessmanagementsystems
Operational status	Verfügbarkeit des Prozessmanagementsystems (Capability *Operational Status* mit den möglichen Werten *available, unavailable, partially available, unknown*)
Position	Aktuelle geographische Position des Prozessmanagementsystems (Ergänzung der Capability *Metrics*)
Local time	Aktuelle Uhrzeit an der Position des Prozessmanagementsystems (Ergänzung der Capability *Metrics*)
Workload	Aktuelle Auslastung der Prozess-Engine (Ergänzung der Capability *Metrics*)
Process models	Liste aller installierten Prozessmodelle
Process instances	Liste aller laufenden Prozessinstanzen
Process histories	Liste aller Prozess-Historien

Tabelle 7.3: Eigenschaften der Ressource *Prozessmanagementsystem*

lichung im Folgenden in jeweils einer Tabelle zusammengefasst. Metadaten werden an dieser Stelle nicht diskutiert, da sie im Speziellen der Dynamik der Ausführungsumgebung und deren individuellen Zugriffsbeschränkungen unterliegen. Für Details zu einer möglichen Abbildung auf konkrete Datentypen, Parameter und Rückgabewerte wird auf die Arbeit von WUNDERLICH [Wun09] verwiesen.

Tabelle 7.3 stellt die Eigenschaften der obersten Hierarchieebene in Form des Prozessmanagementsystems dar. Dieses verfügt über grundlegende Eigenschaften und Kontextinformationen und lässt Aussagen über installierte Prozessmodelle, laufende Prozessinstanzen und abgeschlossene Prozesshistorien zu. Durch die Bereitstellung der Eigenschaft *Ressource ID* wird die in WSDM obligatorische Capability *Identity* unterstützt. Eine Kombination der Eigenschaften *Caption*, *Version* und *Description* erfüllt die WSDM-Capability *Description*, welche noch durch die Eigenschaften *Supported PDLs* und *Owner* ergänzt werden kann. Die Eigenschaft *Operational status* dient der Angabe der Verfügbarkeit nach der gleichnamigen WSDM-Capability. Die durch WSDM vorgegebene generischen Zustände *available* unf *unavailable* lassen sich dabei direkt auf die Betriebsfähigkeit des Prozessmanagementsystem abbilden. Der Zustand *partially available* drückt aus, dass die Betriebsfähigkeit des Prozessmanagementsystem momentan eingeschränkt ist, z. B. da das System gerade hochgefahren wird oder eine Auslastungsgrenze erreicht ist. Weitere Details können in diesem Fall über die Abfrage weiterer Eigenschaften, wie z. B. der Eigenschaft *Workload*, bezogen werden.

Die Eigenschaften *Position*, *Local Time* und insbesondere *Workload* können als mögliche Erweiterungen der Capability *Metrics* modelliert werden. Hierbei werden konkrete Messwerte der verwalteten Ressource angegeben und mit Metadaten versehen, so dass eine korrekte Interpretation des Messwerts möglich ist. Zum Beispiel kann für die Eigenschaft *Workload* angegeben wer-

den, dass diese in Prozent angegeben werden soll und welches die Bezugsgrößen sind. Metrische Daten können in WSDM zudem mit den Attributen *ResetAt* (letzte Rücksetzung des Wertes), *Last Updated* (letzter Aktualisierungszeitpunkt) und *Duration* (Dauer der Datenerhebung) versehen werden, um Daten verschiedener Ressourcen besser vergleichbar zu machen. An dieser Stelle sind neben den drei genannten Eigenschaften viele weitere Eigenschaften denkbar, welche Informationen über das Prozessmanagementsystem liefern. Im Kontext verteilt ausgeführter Prozesse sei dabei jedoch angemerkt, dass die meisten dieser das gesamte Prozessmanagementsystem betreffenden Eigenschaften in der Regel nicht durch den Initiator eines einzelnen Prozesses änderbar sind. Aus diesem Grund sind in diesem Beispiel keine Eigenschaften der Capability *Configuration* modelliert. Dies soll jedoch nicht heißen, dass Änderungen an Eigenschaften auf dieser Ebene prinzipiell ausgeschlossen sind.

Die für die verteilte Prozessausführung interessanten Eigenschaften umfassen die Informationen über die auf dem Prozessmanagementsystem laufenden Prozesse. Durch die Möglichkeit, die Prozessmodelle, Prozessinstanzen und Prozesshistorien als jeweils eigene verwaltbare Subressource nach dem Prinzip von WSDM zu modellieren, ergibt sich der Vorteil, dass jeder relevante Prozess mit einem eigenen Dienstzugriffspunkt versehen werden kann. Im Fall von verteilt bzw. entfernt ausgeführten Prozessen verschiedener Auftraggeber können somit jedem Auftraggeber die Zugriffspunkte „seiner“ individuellen Prozesse zu deren Überwachung und Steuerung mitgeteilt werden. Dies erleichtert insbesondere die Anwendung von Mechanismen der Zugriffskontrolle und die Verwaltung der Ressourcen von Seiten des Auftraggebers, da beim Deployment bzw. der Instantiierung von Prozessen direkt der Zugriffspunkt zum Management des entfernt ausgeführten Prozesses zurückgegeben werden kann und nicht umständlich über das Prozessmanagementsystem des Dienstanbieters erfragt werden muss. Der Nachteil dieser Variante ist, dass dabei für jede relevante verwaltbare Subressource in der Regel auch eine eigene Schnittstelle bereitgestellt werden muss. Da bei der Anwendung von WSDM jedoch ein Zusammenspiel von verwaltbaren und nicht verwaltbaren Ressourcen unterstützt wird, müssen im Einzelfall nur Schnittstellen für Subressourcen zur Verfügung gestellt werden, welche auch tatsächlich durch einen Konsumenten der Managementfunktionalität verwaltet werden sollen (z. B. im Rahmen einer vorhergehenden Registrierung).

Die drei genannten Eigenschaften *Process models*, *Process instances* und *Process histories* können als Definition für WSDM-*Relationship Types* angesehen werden, wobei jede Beziehung zwischen den Entitäten als WSDM-*Relationship* abgebildet werden kann. Alternativ zur Modellierung solcher Beziehungen können die Prozessmodelle, Prozessinstanzen und Prozesshistorien jedoch auch als komplexe Eigenschaften des Prozessmanagementsystems abgebildet werden. Das Ergebnis der Anfrage wäre in diesem Fall ein entsprechender Auszug aus dem *Resource Property Document* mit den Prozessen als komplexer Ergebnistyp. Die Anfragen bzw. Ergebnislisten lassen sich in die-

Eigenschaft (*Property*)	Beschreibung
Ressource ID	Obligatorischer Identifikator der Ressource (Capability *Identity*)
Caption	Name des Prozessmodells
Version	Version des Prozessmodells
Description	Beschreibung des Prozesses
PDL	Prozessbeschreibungssprache des Prozessmodells (*Process Description Language*)
Owner	Organisatorischer Ursprung des Prozessmodells
Operational status	Verfügbarkeit des Prozessmodells (Capability *Operational Status* mit den möglichen Werten *available, unavailable, partially available, unknown*)
Deployment date	Zeitpunkt des Deployments
Process definition	Gesamte Prozessbeschreibung des Prozessmodells
Activities	Liste aller Aktivitäten des Prozessmodells
Data fields	Liste aller Variablen des Prozessmodells
Performers	Liste aller zugewiesenen Prozessbeteiligten
Events	Liste aller fachlichen Ereignisse des Prozessmodells
Process instances	Liste aller laufenden Prozessinstanzen dieses Prozessmodells
Process histories	Liste aller Prozess-Historien dieses Prozessmodells
Average duration	Durchschnittliche Ausführungszeit der von diesem Prozessmodell abgeleiteten Prozessinstanzen (Ergänzung der Capability *Metrics*)
Maximum duration	Längste Ausführungszeit der von diesem Prozessmodell abgeleiteten Prozessinstanzen (Ergänzung der Capability *Metrics*)
Instances count	Anzahl der von diesem Prozessmodell abgeleiteten Prozessinstanzen (Ergänzung der Capability *Metrics*)
Failed instances	Anzahl der nicht erfolgreich ausgeführten, von diesem Prozessmodell abgeleiteten Prozessinstanzen (Ergänzung der Capability *Metrics*)

Tabelle 7.4: Eigenschaften der Subressource *Prozessmodell*

sem Fall filtern oder spezialisieren, so dass auch hier eine Abfrage bzw. Manipulation von untergeordneten Eigenschaften möglich ist (z. B. durch die Operation *QueryRessourceProperties*, vgl. Tabelle 7.2). Eine weitere Möglichkeit zum Zugriff auf die Subressourcen ist die Definition spezieller Operationen zum Zugriff der genannten Entitäten. Ein Beispiel hierfür wird im nächsten Unterabschnitt erläutert.

Tabelle 7.4 zeigt eine Übersicht der Eigenschaften der Subresssource für die Abbildung eines Prozessmodells. Die Eigenschaften *Ressource ID*, *Caption*, *Version* und *Description* gelten analog zur Ressource des übergeordneten Prozessmanagementsystems zur Unterstützung der dort genannten Standard-Capabilities. Die ergänzende Eigenschaft *PDL* gibt die Prozessbeschreibungssprache an, in welcher das Prozessmodell verfasst ist. Die ergänzende Eigenschaft *Owner* spezifiziert zudem die Person oder Organisation, welche das Prozessmodell auf diesem Ausführungssystem installiert hat.

Die Eigenschaft *Operational status* dient der Angabe der Verfügbarkeit eines Prozessmodells nach der gleichnamigen WSDM-Capability (siehe oben). Ein Prozessmodell ist dabei verfügbar, solange es auf der betrachteten Prozess-Engine installiert ist und Prozessinstanzen davon abgeleitet werden können. Die ressourcenspezifische Eigenschaft *Deployment date* gibt dabei den Zeitpunkt des Deployments an. Da im Kontext verteilt ausgeführter Prozesse Prozessmodelle verschiedener Auftraggeber auf einem einzigen Prozessmanagementsystem ausgeführt werden können und nicht zwingend jedes Prozessmo-

Eigenschaft (*Property*)	Beschreibung
Ressource ID	Obligatorischer Identifikator der Ressource (Capability *Identity*)
Process model	Prozessmodell, von dem diese Prozessinstanz abgeleitet wurde
Owner	Initiator der Prozessinstanz
Creation date	Zeitpunkt der Instantiierung
Operational status	Verfügbarkeit der Prozessinstanz (Capability *Operational Status* mit den möglichen Werten *available, unavailable, partially available, unknown*)
State	Aktueller Zustand der Prozessinstanz nach dem definierten Zustandsmodell für migrierbare Prozesse mit den mögliche Werten *created, (option), (transferring), running, in error, suspended, finished, terminating, terminated, deleted*
Activities	Liste aller Aktivitäten der Prozessinstanz
Data fields	Liste aller Variablen der Prozessinstanz
Performers	Liste aller aktuell Prozessbeteiligten
Events	Liste aller Ereignisse der Prozessinstanz
Progress	Fortschritt der Prozessausführung (Ergänzung der Capability *Metrics*)
Time remaining	Prognose über verbleibende Ausführungszeit bis zur Terminierung der Prozessinstanz (Ergänzung der Capability *Metrics*)

Tabelle 7.5: Eigenschaften der Ressource *Prozessinstanz*

dell einen global eindeutigen Identifikator besitzt, dient die Kombination aus den Eigenschaften *Caption*, *Version*, *Owner* und *Deployment date* als Korrelationseigenschaft (Capability *CorrelatableProperties*), wobei alle genannten Eigenschaften übereinstimmen müssen (*matchAll*).

Dem Prozessmodell sind wiederum die von ihm abgeleiteten Prozessinstanzen und -historien in Form einer Beziehung untergeordnet. Die Eigenschaften *Activities*, *Data fields*, *Performer* und *Events* stellen auf der Ebene des Prozessmodells keine verwaltbaren Ressourcen im Sinne von WSDM dar, sondern geben lediglich die in der Prozessbeschreibung definierten Objekte zurück. Da es sich hierbei nicht um metrische Informationen zur Leistung und zum Betrieb der Ressource handelt, sondern spezielle Aspekte des Prozessmanagements ausgedrückt werden, werden diese Eigenschaften unter der ressourcenspezifischen Capability *Process Management* zusammengefasst. Das gleiche gilt für den Zeitpunkt des Deployments (*Deployment date*) und für die Prozessbeschreibung als Ganzes (*Process description*). Eine prinzipielle Änderbarkeit dieser Eigenschaften bleibt von dieser Einordnung unberührt. So ist es zum Beispiel denkbar, die Prozessbeschreibung (bzw. Teile davon) als änderbar zu deklarieren und damit auch eine inhaltliche Anpassung des Prozesses zu erlauben (vgl. Abschnitt 3.3.1).

Exemplarisch sind für das Prozessmodell weitere für eine Überwachung potentiell interessante Eigenschaften angegeben, welche analog zur Ressource *Process Management System* die Capability *Metrics* erweitern. Details hierzu könnend der Arbeit von WUNDERLICH [Wun09] entnommen werden.

Tabelle 7.5 zeigt die Eigenschaften der Subresssource für die Abbildung einer Prozessinstanz. Analog dazu stellt Tabelle 7.6 die Eigenschaften der entsprechenden Prozesshistorie dar. Obwohl beide Entitäten des Modells über einen ähnlichen Aufbau verfügen, wird hierbei deutlich, dass sie über unterschiedliche interessante Eigenschaften verfügen können, z. B. hinsichtlich des

Eigenschaft (*Property*)	Beschreibung
Ressource ID	Obligatorischer Identifikator der Ressource (Capability *Identity*)
Process model	Prozessmodell, von dem diese Prozessinstanz abgeleitet wurde
Owner	Initiator der Prozessinstanz
Creation date	Zeitpunkt der Instantiierung
Termination date	Zeitpunkt der Beendigung der Ausführung
Operational status	Verfügbarkeit der Prozesshistorie(Capability *Operational Status* mit den möglichen Werten *available, unavailable, partially available, unknown*)
State	Letzter Zustand der Prozessinstanz nach dem definierten Zustandsmodell für migrierbare Prozesse mit den möglichen Werten *created, (option), (transferring), running, in error, suspended, finished, terminating, terminated, deleted*
Activities	Liste aller Aktivitäten der beendeten Prozessinstanz
Data fields	Liste aller Variablen der beendeten Prozessinstanz
Performers	Liste aller Prozessbeteiligten der beendeten Prozessinstanz
Events	Liste aller Ereignisse der beendeten Prozessinstanz
Duration	Gesamte Ausführungsdauer der beendeten Prozessinstanz (Ergänzung der Capability *Metrics*)

Tabelle 7.6: Eigenschaften der Ressource *Prozesshistorie*

Zustandes und der Verfügbarkeit von Daten zur Auswertung. Prozessinstanzen verfügen im Referenzmodell zudem über weitere Entitäten, welche nicht nur als Eigenschaften, sondern auch als eigene verwaltbare Subressourcen angesehen werden können. An dieser Stelle wurden die Entitäten *Aktivität*, *Variable* und *Ereignis* herausgegriffen.

Neben der obligatorischen *Ressource ID* ist die Beziehung zum zugehörigen Prozessmodell eine wesentliche Information. Da Prozessinstanzen in der Regel automatisiert erzeugt werden, existieren normalerweise keine individuellen menschenlesbaren Beschreibungen, welche der Standard-Capability *Description* zugeordnet werden können. Die allgemeinen Informationen können daher über die Beziehung zum Prozessmodell abgerufen werden. Als individuelle Informationen sind jedoch für Prozessinstanzen der Initiator der Prozessinstanz (*Owner*) und der Zeitpunkt der Erzeugung (*Creation date*) bzw. bei einer Prozesshistorie auch der Zeitpunkt der Terminierung (*Termination date*) interessant. Zur Unterstützung der Standard-Capability *CorrelatableProperties* dient die Kombination aus den Eigenschaften *Process model*, *Owner* und *Creation date* als Korrelationseigenschaft, wobei alle genannten Eigenschaften übereinstimmen müssen (*matchAll*).

Die Eigenschaft *Operational status* dient der Angabe der Verfügbarkeit einer Prozessinstanz nach der gleichnamigen WSDM-Capability (siehe oben). Eine Prozessinstanz ist dabei verfügbar, solange sie auf der betrachteten Prozess-Engine ausgeführt wird, d. h. vom Zeitpunkt ihrer Instantiierung bis zu ihrer Terminierung. Eine Prozesshistorie ist verfügbar, sofern die Prozessinstanz abgeschlossen ist und Prozesshistorien auf dem betrachteten System gespeichert werden. Der aktuelle bzw. letzte Zustand wird durch die WSDM-Capability *State* angegeben, welche die Integration eines eigenen Zustandsmodells erlaubt (vgl. Tabelle 7.1). Es kann hierbei der aktuelle Zustand, der vorhergehende Zustand und der Zustandsübergang mit Zeitangabe abgefragt werden.

Eigenschaft (*Property*)	Beschreibung
Ressource ID	Obligatorischer Identifikator der Ressource (Capability *Identity*)
Caption	Name der Aktivität
Process instance	Zugehörige Prozessinstanz
Start date	Zeitpunkt des Beginn der Ausführung
End date	Zeitpunkt der Beendigung der Ausführung
State	Aktueller Zustand der Aktivität nach dem definierten Zustandsmodell für Aktivitäten migrierbarer Prozesse mit den möglichen Werten *inactive, ready, executing, executed, skipped, expired, in error, finished, terminated*
Input data fields	Liste der Variablen, welche von dieser Aktivität gelesen werden
Output data fields	Liste der Variablen, welche von dieser Aktivität geschrieben werden
Performer	Für die Ausführung verantwortlicher Prozessbeteiligter
Duration	Bisherige bzw. gesamte Ausführungsdauer der Aktivität (Ergänzung der Capability *Metrics*)

Tabelle 7.7: Eigenschaften der Ressource *Aktivität (Instanzebene)*

In dieser Arbeit wurde als Beispiel das Zustandsmodell für migrierbare Prozesse hinterlegt (vgl. Abschnitt 6.2.2), wobei der aktuelle Zustand über die hier gleichnamige Eigenschaft *State* ausgedrückt wird. Nach WSDM ist diese Eigenschaft ist durch das Metadatum *read only* immer schreibgeschützt. Daher sind keine Manipulationen der Ressource über das Verändern ihres Zustands möglich. Im Kontext verteilt ausgeführter Prozesse ist dies jedoch für die Steuerung des Prozesses, z.B. für dessen Abbruch, relevant. Es müssen daher hierfür entsprechende ergänzende Operationen im Rahmen der ressourcenspezifischen Capability *Process management* bereitgestellt werden.

Entsprechend des in Abschnitt 6.3.2 vorgeschlagenen Modells ist jede Prozessinstanz weiter in die (multiplen) Entitäten der Arten *Aktivität* (*Activity*), *Prozessteilnehmer* (*Performer*), *Variable* (*Data field*) und *Ereignis* (*Event*) gegliedert. Aufgrund ihrer potentiell unterschiedlichen Beschaffenheit werden Prozessteilnehmer in dieser Arbeit nicht als verwaltbare Ressourcen modelliert. Wird eine Aktivität jedoch durch einen WSDM-MOWS-kompatiblen Web Service ausgeführt (vgl. [WS06]), so ist es möglich, an dieser Stelle über eine entsprechende Beziehung direkt den Zugriffspunkt für die Managementfunktionalität des Dienstes einzubinden. Somit können nahtlos auch die dort bereitgestellten Eigenschaften und Operationen (z. B. zur Verfügbarkeit des Dienstes) mit den Managementfunktionalitäten des Prozessmanagementsystems integriert werden.

Exemplarisch sind für Prozessinstanzen und -historien weitere für eine Überwachung und die Anfertigung von Statistiken potentiell interessante Eigenschaften angegeben, welche analog zu den zuvor genannten Ressourcen die Capability *Metrics* erweitern. Details hierzu können tlw. der Arbeit von WUNDERLICH [Wun09] entnommen werden.

Tabelle 7.7 zeigt die Eigenschaften der Subresssource für die Abbildung einer Aktivität auf Instanzebene. Diese ist von einer Aktivität auf Modellebene zu unterscheiden, da im Rahmen von Schleifen innerhalb des Kontrollflusses des Prozesses eine Mehrfachausführung von Aktivitäten möglich ist. Dabei

Eigenschaft (*Property*)	Beschreibung
Ressource ID	Obligatorischer Identifikator der Ressource (Capability *Identity*)
Caption	Variablenbezeichner
Process instance	Zugehörige Prozessinstanz
Data class	Angabe einer Datenklasse zur Synchronisation
Last update	Zeitpunkt der letzten Änderung
Locked	Angabe über Sperren
Value	Aktueller Wert der Variablen (Ergänzung der Capability *Metrics*)

Tabelle 7.8: Eigenschaften der Ressource *Prozessvariable*

wird bei jeder Iteration eine neue Instanz der Aktivität erzeugt, welche z. B. jeweils einen eigenen Start- und Endzeitpunkt für ihre Ausführung besitzt.

Wie bei den zuvor beschriebenen Ressourcen werden die Eigenschaften *Ressource ID* und *Caption* auf die Capabilities *Identity* bzw. *Description* abgebildet. Die Beziehung zur Prozessinstanz wird über die Eigenschaft *Process instance* hergestellt. Zudem wird der Zeitpunkt des Beginns (*Start date*) und der Beendigung (*End date*) der Ausführung angegeben. Der Zustand der Aktivität wird in dieser Arbeit über das Zustandsmodell für Aktivitäten migrierbarer Prozesse hinterlegt (vgl. Abschnitt 6.2.2), wobei der aktuelle Zustand über die Eigenschaft *State* der gleichnamigen Standard-Capability ausgedrückt wird. Desweiteren führt die Aktivität die Liste der während der Ausführung gelesenen Variablen (*Input data fields*) als Repräsentation des Input-Containers und analog eine Liste der geschriebenen Variablen (*Output data fields*) als Repräsentation des Output-Containers (vgl. auch Abschnitt 2.5). Desweiteren können beliebige weitere Eigenschaften als Erweiterung der Capability *Metrics* angegeben werden. Exemplarisch ist hierfür die Ausführungsdauer der Aktivität (*Duration*) angegeben.

Als Beispiel für eine änderbare Eigenschaft im Rahmen der Capability *Configuration* kann z. B. der für die Ausführung verantwortliche Prozessteilnehmer (*Performer*) angesehen werden. Dieser kann prinzipiell angepasst werden, bevor die Ausführung der Aktivität beginnt, was durch die Metadaten der Eigenschaft ausgedrückt werden kann. Die Änderung des Prozessteilnehmers ist insbesondere für die Verteilung laufender Prozessinstanzen relevant, wo Ressourcen nach erfolgter Migration auf eine andere Ausführungseinheit unter Umständen neu gebunden werden müssen.

Tabelle 7.8 zeigt die Eigenschaften der Subresssource für die Abbildung einer Variablen (*Data field*). Hier wurde bereits ein besonderes Augenmerk auf eine möglicherweise notwendige Synchronisation bei der verteilten Ausführung des Prozesses gelegt. Ergänzend zu den bereits im Kontext der anderen Ressourcen erläuterten Eigenschaften betrifft dies die Eigenschaften *Data class* als Repräsentation der Datenklasse, welche Informationen über die Synchronisation der Variablen enthält, *Last update* als Zeitpunkt der letzten Änderung des Variablenwertes und *Locked* als Angabe, ob die Variable aktuell für Lese- und/oder Schreibzugriffe gesperrt ist. Schließlich gibt *Value* den aktuellen Wert der Variablen an.

Eigenschaft (*Property*)	Beschreibung
Ressource ID	Obligatorischer Identifikator der Ressource (Capability *Identity*)
Caption	Name des (fachlichen) Ereignisses
Process instance	Zugehörige Prozessinstanz
Event source	Quelle des Ereignisses
State	Aktueller Zustand des Ereignisses nach dem definierten Zustandsmodell
Last occurrence	Zeitpunkt des letzten Eintritts

Tabelle 7.9: Eigenschaften der Ressource *(fachliches) Ereignis*

Messbare Eigenschaften im Sinne der Capability *Metrics* werden für Prozesvariablen nicht definiert. Im Rahmen von inhaltlichen Anpassungen und Synchronisationen ist es aber denkbar, dass Variablenwerte zur Laufzeit geändert werden sollen. Es kann daher die Eigenschaft *Value* im Rahmen der Capability *Configuration* bereitgestellt und so bei Bedarf über die Methode *SetRessourceProperties* geändert werden. Dabei müssen natürlich etwaige Seiteneffekte beachtet werden.

In Tabelle 7.9 werden schließlich mögliche Eigenschaften der Subresssource für die Abbildung eines innerhalb der Prozessbeschreibung definierten Ereignisses dargestellt. Dieses fachliche Ereignis ist dabei von den (technischen) Ereignissen abzugrenzen, welche von der verwaltbaren Ressource selbst hinsichtlich der Änderung von Eigenschaften ausgelöst werden. Für eine Überwachung sind insbesondere der Zeitpunkt des letzten Eintritts (*Last occurrence*) und die Quelle des Ereignisses (*Event source*) interessant. Als mögliche Änderungsoperation kann die Quelle des Ereignisses auch durchaus zur Laufzeit des Prozesses angepasst werden (Capability *Configuration*) und so z. B. das Rebinding von Ereignisquellen bei einer Anpassung der Verteilung des Prozesses unterstützen.

Prozessmanagement-spezifische Operationen

Im Rahmen der (verteilten) Prozessausführung werden unter Umständen weitere Managementfunktionalitäten benötigt, welche sich nicht durch ein Auslesen oder Manipulieren der genannten Eigenschaften oder durch Standard-Capabilities realisieren lassen. Dies ist insbesondere für Steuerungsfunktionalitäten der Fall, welche den Zustand von Prozessinstanzen und Aktivitäten beeinflussen, da dieser in WSDM standardmäßig als schreibgeschützte Eigenschaft geführt wird. WSDM erlaubt jedoch die Einführung eigener Capabilities und damit die Definition von ergänzenden Operationen, welche der Managementschnittstelle der verwaltbaren Ressource hinzugefügt werden, ohne den Standard dadurch zu verletzen. Im Tabelle 7.10 werden exemplarisch eine Reihe von Operationen aufgezeigt, welche der neuen ressourcenspezifischen Capability *Process Management* und den dort bereits definierten Eigenschaften zugeordnet werden. Sie stellen grundlegende Funktionalitäten zum Deployment von Prozessmodellen sowie zur Verwaltung des Ausführungszustands von Prozessinstanzen und Aktivitäten dar.

Operation	Beschreibung
Deploy process	Installiert das angegebene Prozessmodell auf dem Prozessmanagementsystem
Undeploy process	Deinstalliert das angegebene Prozessmodell von dem Prozessmanagementsystem
Instantiate process	Instantiiert das angegebene Prozessmodell mit angegebenen Parametern und abonniert optional angegebene Ereignisse der resultierenden Prozessinstanz. Die Prozessinstanz wird nach dem Aufruf dieser Operation sofort ausgeführt.
Instantiate suspended process	Instantiiert das angegebene Prozessmodell mit angegebenen Parametern und abonniert optional angegebene Ereignisse der resultierenden Prozessinstanz. Die Prozessinstanz befindet sich nach dem Aufruf dieser Operation im Zustand *suspended*.
Suspend process	Hält die Ausführung der angegebenen Prozess-Instanz an, sobald der Ausführungszustand dies erlaubt und versetzt den Prozess entsprechend des definierten Zustandsmodells in den *Suspended*-Zustand
Resume process	Führt die Ausführung der angehaltenen Prozess-Instanz entsprechend des definierten Zustandsmodells fort
Restart process	Abbruch und Neustart der angegebenen Prozessinstanz mit den ursprünglichen Eingabe-Parametern
Cancel process	Abbruch der Ausführung der angegebenen Prozessinstanz
Restart activity	Abbruch und Neustart der angegebenen Aktivität
Cancel activity	Abbruch der Ausführung der angegebenen Aktivität

Tabelle 7.10: Prozessmanagement-spezifische Operationen (tlw. nach (Wun09))

Ereignisschnittstelle

Für die Umsetzung der Ereignisschnittstelle können im Rahmen von *WS-Notification* (*WSN*) [GHM06] Themenbäume spezifiziert werden, welche die Abbildung der Hierarchie von Entitäten aus dem Referenzmodell auf Themen (*Topics*) und Unterthemen (*Subtopics*) für das Abonnement von Ereignissen erlauben. Zusammengehörige Bäume werden zu einem gemeinsamen Namensraum (*Topic Namespace*) zusammengefasst [VGN06]. Für neu installierte Prozessmodelle und neu gestartete Prozessinstanzen können dabei auch dynamisch neue Themen vom Typ *Prozessmodell* bzw. *Prozessinstanz* als Unterthemen erzeugt werden, so dass eine Registrierung auch für einzelne Entitäten dieser Art möglich ist. Dies ist insbesondere in Hinblick auf die Durchsetzung von Zugriffsbeschränkungen wichtig, da bei der Überwachung verteilt ausgeführter Prozesse in der Regel nur die Benachrichtigung über Ereignisse eigener Prozesse erlaubt sein soll, deren Instanzen jedoch zum Zeitpunkt des Ereignisabonnements unter Umständen noch gar nicht existieren [Wun09].

WSDM unterstützt die Bildung der genannten Themenbäume, indem für jede im *Resource Property Document* definierte Eigenschaft ein Thema erzeugt werden kann. Ergänzend können eigene Themen definiert werden. Basierend auf WS-Notification wird die WSDL-Schnittstelle der verwaltbaren Ressource um die Operation *Subscribe* zum Abonnieren von Ereignissen zu einem bestimmten Thema ergänzt. Wird dann der Wert einer Eigenschaft im *Resource Property Document* geändert, so erfolgt eine Benachrichtigung an alle Abonnenten dieses Themas, wobei als Inhalt der WS-Notification-Nachricht ein WSDM-*Management-Event* versendet wird. Standardmäßig können dabei die Änderung des Wertes (d. h. vorhergehender Wert und aktueller Wert) und der Zeitpunkt der Änderung sowie die Ergebnisse weitere Klassifizierungen

des Ereignisses, wie z. B. die Einordnung des Ereignisses in eine bestimmte Priorität, angegeben werden.

Betreffen Ereignisse eine bestimmte Prozessinstanz, so ist es unter Umständen hinderlich, wenn die Prozessinstanz nach dem Auftreten des Ereignisses direkt fortgesetzt wird, da hierdurch angepasste Reaktionen auf die Situation erschwert oder sogar verhindert werden. Aus diesen Grund kann ergänzend zu den durch WSDM definierten Standardfunktionalitäten beim Abonnement eines Ereignisses einer Prozessinstanz und der ihr untergeordneten Entitäten angegeben werden, ob der Prozess nach dem Eintreten des Ereignisses angehalten werden soll. Dabei werden die aktuell ausgeführten atomaren Aktivitäten beendet, aber keine neuen Aktivitäten mehr gestartet, so dass der Prozess in den *Suspended*-Zustand versetzt werden kann. Die Ausführung des Prozesses wird dann erst nach einer Antwort des Ereigniskonsumenten fortgesetzt. Unterbrechende Ereignisse dieser Art werden in dieser Arbeit als *blockierende Ereignisse* bezeichnet [Wun09, ZBH+10] (vgl. auch Abschnitt 6.3.3 und [vLLM+08]). Blockierende Ereignisse können zum Beispiel im Rahmen der Operation *Instantiate process* angegeben werden (vgl. Tabelle 7.10), so dass das Abonnement der Ereignisse für jede einzelne Prozessinstanz direkt bei deren Instantiierung vorgenommen werden kann.

Werden blockierende Ereignisse von mehr als einem Ereigniskonsumenten abonniert, so kann mit der Prozessausführung erst fortgefahren werden, nachdem alle Ereigniskonsumenten geantwortet haben. Damit die Prozessausführung nicht unnötig lange verzögert wird, kann für das Eingehen der Antwortnachrichten eine Zeitbeschränkung definiert werden. Bei Ausbleiben einer Antwortnachricht eines Ereigniskonsumenten innerhalb des individuell angegebenen Zeitraums wird die Prozessausführung fortgesetzt. Prinzipiell können dabei auch gleichzeitig unterschiedliche Reaktionen als Anpassungsmaßnahmen auf die Prozessausführung ausgelöst werden [Wun09]. Da jedoch in der Praxis in der Regel nur eine Partei die Rechte an der Steuerung der Prozessausführung besitzt, ist die Auflösung etwaiger durch nebenläufigen Zugriff resultierender Konflikte nicht Teil dieser Arbeit.

Tabelle 7.11 fasst die wesentlichen Ereignisse der exemplarischen Managementschnittstelle zusammen. Das Ereignis *System state change* zeigt eine Änderung der Eigenschaft *Operational status* des betrachteten Prozessmanagementsystems an. Es kann daher verwendet werden, um eine Liste der verfügbaren Prozessmanagementsysteme zu pflegen, welche im Fall einer verteilten Prozessausführung zur Verfügung stehen. Ereignisse über die Installation (*Process model deployed*) bzw. Deinstallation (*Process model undeployed*) von Prozessmodellen können in WSDM automatisch ausgelöst werden, sobald eine neue verwaltbare Ressource erzeugt bzw. deren Existenz beendet wird. Dasselbe gilt für die Initialisierung und Beendigung von Prozessinstanzen sowie für die Erzeugung von Prozesshistorien. Die Ereignisse *Process state changed* und *Activity state changed* beziehen sich auf Änderungen der jeweiligen *State*-Eigenschaften von Prozessinstanz bzw. Aktivität, während das Ereignis *Value changed* sich auf die *Value*-Eigenschaft einer Prozessva-

Ressource	Ereignis	Beschreibung
Process management system	System state change	Änderung der Verfügbarkeit der Prozess-Engine
	On error	Auftreten eines Fehlers auf der Ebene des Prozessmanagementsystems
Process model	Process model deployed	Installation eines neuen Prozessmodells
	Process model undeployed	Deinstallation eines Prozessmodells
	On error	Auftreten eines Fehlers auf der Ebene eines Prozessmodells
Process instance	Process initiated	Initialisierung einer neuen Prozessinstanz
	Process state changed	Änderung des Zustands der angegebenen Prozessinstanz
	Process finished	Ordentliche Beendigung der Ausführung der angegebenen Prozessinstanz
	On error	Auftreten eines Fehlers auf der Ebene einer Prozessinstanz
Activity	Activity state changed	Änderung des Zustands der angegebenen Aktivität
	Activity started	Start der Ausführung einer Aktivität
	Activity finished	Ordentliche Beendigung der Ausführung der angegebenen Aktivität
	On error	Auftreten eines Fehlers auf der Ebene einer Aktivität
Data field	Value changed	Änderung des Variablenwertes
Process Event	Event Occured	Eintritt eines innerhalb des Prozesses definierten fachlichen Ereignisses
Process history	Process history created	Erzeugung einer Prozesshistorie

Tabelle 7.11: Basisereignisse der verwaltbaren Ressource (tlw. nach (Wun09))

riablen bezieht. Analog dazu wird bei Eintreten eines fachlichen Ereignisses das Management-Ereignis *Event occured* auf Basis einer Änderung der Eigenschaft *Last occurrence* des fachlichen Ereignisses ausgelöst. Bis hierhin sind daher alle benötigten Ereignisse mit Themen auf Basis des definierten *Resource Property Document* zu realisieren. Die Ereignisse *On error* bei den unterschiedlichen Ressourcen wurden exemplarisch jeweils als zusätzliches Thema ergänzt und werden im Fall von individuellen Fehlern ausgelöst, wobei jeweils eine Beschreibung des Fehlers zurückgeliefert wird.

7.3.3 Integration von Prozessmanagementsystemen

Die konkrete Art und Weise der Anbindung der Managementschnittstelle an das gekapselte Prozessmanagementsystem ist plattformabhängig und kann in der Regel nur durch eine individuelle Anpassung des Programmcodes erfolgen, welche die Integration von geeigneten Methoden zur Extraktion der benötigten Informationen, zur Manipulation von Eigenschaften und zur Benachrichtigung bei Auftreten der definierten Ereignisse umfasst. Werden diese Anpassungen für jede Managementfunktionalität einzeln durchgeführt, können dadurch ein relativ hoher Änderungsaufwand und ggf. eine Beeinflussung der Laufzeiteigenschaften des Prozessmanagementsystems verursacht werden. Es ist daher zu diskutieren, wie die Kapselung von Prozessmanagementsystemen mit möglichst wenigen Eingriffen in den (bestehenden) funktionalen Programmcode effizient gestaltet werden kann. Eine Möglichkeit hierfür ist der Einsatz von Adapterkomponenten, welche grundlegende Managementfunktionalitäten als Ausgangsbasis nehmen, um daraus komplexere Funktionalitäten zusammenzusetzen und diese der Managementschnittstelle anzubieten.

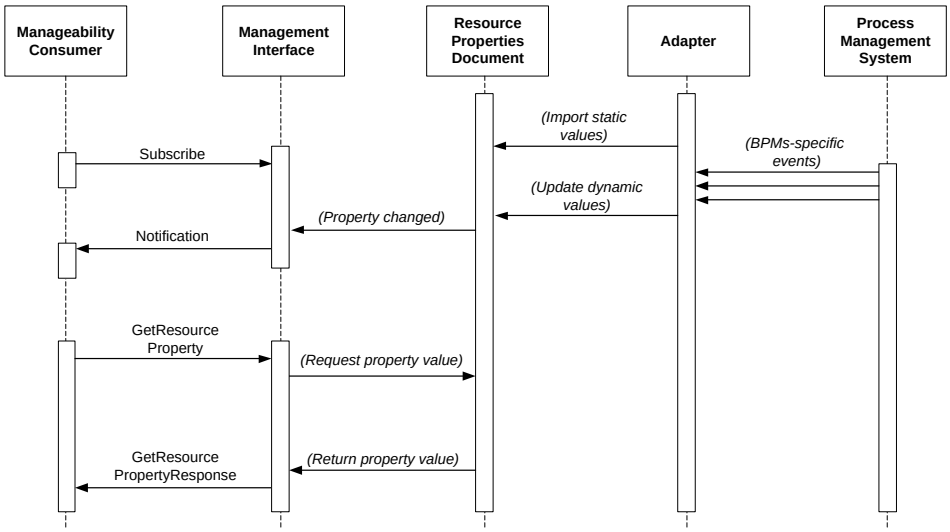


Abbildung 7.9: Interaktionsdiagramm zur Veranschaulichung der Beziehungen zwischen Managementschnittstelle und Prozessmanagementsystem: Lesende Zugriffe (tlw. nach (Wun09))

Analog zu den definierten Managementfunktionalitäten und deren Unterteilung in lesenden und schreibenden Zugriff auf die definierten Eigenschaften der verwaltbaren Ressource lassen sich hinsichtlich der Entwicklung von Adapterkomponenten die Implementierung beobachtender und modifizierender Operationen unterscheiden. Die Abbildungen 7.9 und 7.10 zeigen den möglichen Ablauf von Interaktionen zwischen dem Konsumenten der Managementfunktionalität, der Managementschnittstelle, der (Objekt-)Repräsentation des *Resource Properties Document*, einer Adapterschicht und dem gekapselten Prozessmanagementsystem. Abbildung 7.9 umfasst dabei lesende Zugriffe auf Eigenschaften des Prozessmanagementsystems am Beispiel eines Ereignisabonnements (als Umsetzung des *Push*-Interaktionsmodells) und am Beispiel der Operation *GetRessourceProperty* (als Umsetzung des *Pull*-Interaktionsmodells). Abbildung 7.10 verdeutlicht einen modifizierenden Zugriff am Beispiel der Operation *SetRessourceProperties*.

Für die Realisierung nur lesender Zugriffe (vgl. Abbildung 7.9) übernimmt die Adapterkomponente zwei wesentliche Aufgaben. Zum einen werden bei der Anbindung der Ressource einmalig statische Informationen über das gekapselte Prozessmanagementsystem in das *Resource Property Document* übertragen. Beispiele für statische Informationen sind die Werte der Eigenschaften *Caption*, *Owner* oder *Supported PDLs*, welche sich in der Regel nicht ändern oder nur angepasst werden müssen, wenn das Prozessmanagementsystem oder wesentliche Teile davon ersetzt werden. Zum anderen empfängt die Adapterkomponente plattformabhängige Ereignisse des Prozessmanagementsystems und bildet diese auf die hiervon betroffenen Eigenschaften des *Resource Property Document* ab. Wesentliche Ereignisse stellen hierbei insbesondere die Installation bzw. Deinstallation von Prozessmodellen, das Erzeugen neuer Prozessinstanzen sowie die Informationen über Zustandsübergänge der einzelnen

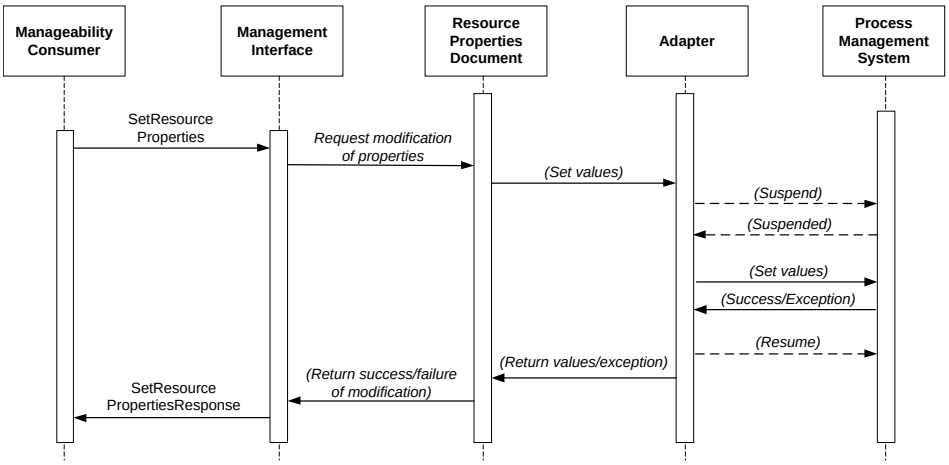


Abbildung 7.10: Interaktionsdiagramm zur Veranschaulichung der Beziehungen zwischen Managementschnittstelle und Prozessmanagementsystem: Schreibende Zugriffe (tlw. nach (Wun09))

Prozessinstanzen und die Zuweisung von Prozessdaten zu Variablenwerten dar. Zusätzlich zum Prozessmanagementsystem können an dieser Stelle aber auch weitere Informationsquellen (wie zum Beispiel ein Kontextmanagementsystem) angebunden werden, um zum Beispiel Informationen über den geographischen Aufenthaltsort des Prozessmanagementsystems zu integrieren. Im Idealfall ist das *Resource Property Document* auf diese Weise immer auf dem aktuellen Stand und kann damit sowohl als Grundlage für die Weiterleitung von Ereignissen an registrierte Beobachter als auch als Informationsquelle für individuelle Anfragen nach dem momentanen Wert von Eigenschaften dienen.

Die Realisierung schreibender Zugriffe auf das Prozessmanagementsystem ist mit größerem Aufwand verbunden (vgl. Abbildung 7.10). Insbesondere müssen hierbei die in Tabelle 7.10 genannten prozessmanagementspezifischen Operationen umgesetzt werden, da für die Modifikation von Eigenschaften laufender Prozessinstanzen das temporäre Anhalten der betreffenden Prozessinstanz notwendig ist. Dies wirkt sich auch auf das Empfangen von blockierenden Ereignissen des Prozessmanagementsystems durch die Adapterkomponente aus, da diese programmiertechnisch nicht in Form einer typischen ereignisbasierten Kommunikation innerhalb eines neuen Threads erfolgen kann, sondern stattdessen in Form eines Methodenaufrufs mit späteren Rückgabe des Programmkontrollflusses realisiert werden muss.

Die Umsetzung von Adapterkomponenten zur Kapselung von verwaltbaren Ressourcen wird bereits durch bestehende Implementierungen des WSDM-Standards unterstützt. Ein Beispiel hierfür ist das Projekt *Muse* der *Apache Software Foundation* [Apa07], welches ein Java-basiertes Rahmenwerk und entsprechende Werkzeuge zur Verfügung stellt, um auf Basis der WSDL-Beschreibung der Managementschnittstelle und des konkreten *Resource Property Document* Java-Klassen mit (leeren) Methodenrümpfen zur individuellen Integration der zu kapselnden Ressource zu generieren [Apa07,

Wun09]. In Bezug auf Prozessmanagementsysteme als zu kapselnde Ressource können drei mögliche Integrationsstrategien identifiziert werden, welche eine möglichst lose Kopplung der Adapterkomponente zum Prozessmanagementsystem ermöglichen und welche bei Vorliegen der systemspezifischen Voraussetzungen auch in Kombination eingesetzt werden können:

- **Nutzung bestehender APIs:** Bestehen auf der Seite des Prozessmanagementsystems bereits Schnittstellen auf der Ebene der Anwendungsprogrammierung (*Application Programming Interface*, kurz *API*), um grundlegenden Funktionalitäten des Prozessmanagements (vgl. Tabelle 7.10) zur individuellen Vor- und Weiterverarbeitung nutzbar zu machen, so können diese direkt durch die Adapterkomponente verwendet werden.
- **Ereignisbasierte Integration:** Ein Großteil der genannten Eigenschaften eines Prozessmanagementsystems und der ausgeführten Prozesse können durch das Abonnement relevanter interner Ereignisse des Prozessmanagementsystems beobachtet werden. Dabei können durch die Adapterkomponente entweder bestehende Ereignisse des Prozessmanagementsystems abonniert und weiterverarbeitet werden oder die benötigte Ereignisstruktur muss explizit in das Prozessmanagementsystem hinein implementiert werden. Das Empfangen der Ereignisse ermöglicht der Adapterkomponente in beiden Fällen das Spiegeln der für das Management relevanten Eigenschaften und deren Aufbereitung, so dass deren Abruf über die Managementschnittstelle ermöglicht wird. Modifikationen sind über diese Integrationsstrategie jedoch nicht möglich.
- **Datenbankbasierte Integration:** Prozessmanagementsysteme verwenden in der Regel ein Datenbanksystem, um Prozessmodelle, Prozessinstanzen und die bei der Ausführung anfallenden Prozessdaten zu speichern und zur Laufzeit zu verwalten. Ein Zugriff auf das Datenbanksystem erlaubt ein transaktionssicheres Lesen von relevanten Parametern ohne umfangreiche Änderungen am Prozessmanagementsystem selbst durchführen zu müssen. Neben der (periodischen) Abfrage von Datensätzen zur Weiterverarbeitung in der Adapterkomponente können in Abhängigkeit des verwendeten Datenbanksystems auch höherwertige Funktionen, wie zum Beispiel *Trigger* oder *Stored Procedures* verwendet werden, um komplexere Funktionalitäten der Adapterkomponente zu realisieren. Für die Modifikation von Parametern zur Laufzeit von Prozessinstanzen ist jedoch in der Regel ergänzend das geordnete Anhalten bzw. die Wiederaufnahme der Prozessausführung notwendig.

Die geeignete Form der Anbindung ist sowohl von der konkreten Implementierung des Prozessmanagementsystems als auch von der für das Management benötigten Funktionalität abhängig und soll daher an dieser Stelle nicht weiter thematisiert werden. Im Folgenden wird davon ausgegangen, dass die für das in Abschnitt 6.3.2 vorgestellte Referenzmodell spezifizierten Eigenschaften und Managementfunktionalitäten abgebildet werden können und für die

Überwachung und Steuerung entfernt ausgeführter Prozesse bzw. Prozesspartitionen zur Verfügung stehen. Aufbauend auf dieser Basisfunktionalität können die nachfolgend vorgestellten Komponenten zum Offline-Management und zur dynamischen Verteilung der Prozessausführung realisiert werden. Eine konkrete Anwendung der hier aufgezeigten Integrationsstrategien wird am Beispiel verschiedener Prozessmanagementsysteme in Kapitel 8 wieder aufgegriffen.

7.4 Managementdienst

Spontane Anpassungen entfernt ausgeführter Prozesspartitionen können über die Kapselung des Prozessmanagementsystems als verwaltbare Ressource und die entsprechende Bereitstellung von grundlegenden Managementfunktionalitäten als Dienstinfrastruktur auf Basis von WSDM ermöglicht werden (vgl. 7.3. Hingegen wird für die Durchführung vordefinierter Reaktionen auf bestimmte (Ausnahme-)Situationen eine zusätzliche Funktionalität benötigt, welche benutzerdefinierte Rahmenbedingungen entgegennimmt, auf ihre Einhaltung überwacht und bei Erkennen einer handlungsbedürftigen Situation die entsprechend definierten Maßnahmen veranlasst. Als Beispiel für die in Abschnitt 6.3.5 vorgeschlagene Komponente zum regel- und ereignisbasierten Management wird in diesem Abschnitt die Realisierung eines *Managementdienstes* vorgestellt, welcher sowohl auf der Seite des Auftraggebers der verteilten Prozessausführung im Rahmen eines *Online-Managements* als auch auf Seiten des Anbieters der Managementfunktionalität für ein *Offline-Management* von entfernt ausgeführten Prozesspartitionen verwendet werden kann. Letzteres ist insbesondere im Kontext mobiler Ausführungseinheiten gut geeignet, da hierdurch die Notwendigkeit zum Nachrichtenaustausch minimiert werden kann und auch temporäre Unterbrechungen der Netzwerkverbindung berücksichtigt werden können. In beiden Fällen stellt der Managementdienst die von ihm gekapselte Funktionalität über eine eigene Dienstschnittstelle zur Verfügung (vgl. Tabelle 7.12). Diese kann entsprechend der in Abschnitt 7.2 vorgeschlagenen Infrastruktur als Standard-Web-Service oder als spezialisierte Schnittstelle für mobile Dienstkonsumenten repräsentiert werden.

Der hier vorgeschlagene Managementdienst konsumiert die vom Prozessmanagementsystem angebotenen Managementfunktionalitäten entweder über eine WSDM-basierte Web-Service-Schnittstelle oder über eine entsprechende lokale Schnittstelle. Abbildung 7.11 zeigt analog zu Abbildung 6.37 die Beziehungen zwischen Managementdienst (*Management Service*) und der durch die verwaltbare Ressource bereitgestellten Schnittstelle, wobei der Managementdienst im Kontext von WSDM als *Manageability Consumer* auftritt. Der Managementdienst kann somit im Idealfall mit jedem nach dem Referenzmodell gekapselten Prozessmanagementsystem verwendet werden.

Für die Umsetzung des Managementdienstes wurden eine Reihe von plattformunabhängigen Komponenten entwickelt, welche die Verwaltung der be-

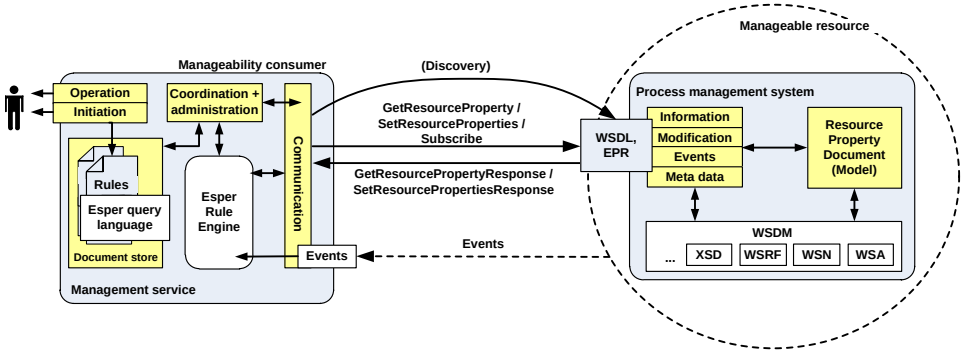


Abbildung 7.11: Prozess-Engine als verwaltbare Ressource mit WSDM und Ereignisverarbeitung mit Esper (ZBH+10)

Operation	Beschreibung
Init	Übergibt ein nach Abschnitt 6.3.5 spezifiziertes Management-Dokument an den Managementdienst und initiiert die im Management-Dokument angegebenen Management-Vorgänge. Als Rückgabe erfolgt im Erfolgsfall eine Referenz auf das interpretierte und gespeicherte Management-Dokument.
Update	Übergibt die nach der Init-Operation erhaltenen Referenz auf ein durch den Managementdienst verwaltetes Management-Dokument und die daran vorzunehmenden Änderungen (in Form eines gültigen Management-Dokuments).
Stop	Übergibt die nach der Init-Operation erhaltenen Referenz auf ein durch den Managementdienst verwaltetes Management-Dokument und beendet alle darin enthaltenen Management-Vorgänge.

Tabelle 7.12: Schnittstelle des Managementdienstes (tlw. nach (Str09))

nutzerdefinierten Rahmenbedingungen in Form der in Abschnitt 6.3.5 vorgeschlagenen *Managementdokumente* und die Kommunikation für die Nutzung der Managementfunktionalitäten der verwaltbaren Ressourcen durchführen (vgl. Abbildung 7.11). Dabei werden die Management-Dokumente an der öffentlichen Schnittstelle des Managementdienstes übergeben, interpretiert und analog zu den Prozessmodellen im Prozessmanagementsystem für die Zeitdauer ihrer Gültigkeit in einer Datenhaltungskomponente (*Document Store*) gespeichert. Ein Koordinations- und Verwaltungsmodul (*Coordination and administration module*) ist (falls notwendig) für die Identifikation des Management-Zugriffspunktes, für das Abonnement der relevanten Ereignisse und für die Berücksichtigung von etwaigen Zeitbeschränkungen verantwortlich. Ein Kommunikationsmodul (*Communication module*) dient zum einen für das Auffinden und den Aufruf von konkreten Diensten als Implementierung des Reaktionsteils der Managementregeln, zum anderen wird an dieser Stelle eine Ersetzung der lokalen und realen Instanzreferenzen in den ein- und ausgehenden Nachrichten vorgenommen [Str09, ZBH+10].

In Abschnitt 3.7.5 wurde die Ableitung von Anpassungsbedarf durch die Verarbeitung von prozessorientierten Ereignissen als wesentlicher Ansatz zur Überwachung und Steuerung von Prozessen in fachlicher Hinsicht untersucht. Zur Verarbeitung der (technischen) Managementregeln ist der hier vorgestellte Managementdienst in der Lage, die Funktionalitäten von bereits bestehen-

den Systemen zur Ereignisverarbeitung zu integrieren. Im Folgenden wird als Beispiel für die mögliche Integration einer Middleware zur Verarbeitung von Ereignisströmen und der Ableitung komplexer Ereignisse die Anbindung der Plattform *Esper* [Esp12] erläutert. Dazu wird zunächst die mögliche Implementierung des Regelteils innerhalb des Managementdokuments durch Ausdrücke der von Esper verwendeten Anfragesprache aufgezeigt (vgl. Abschnitt 7.4.1). Im Anschluss wird die Integration des ereignisverarbeitenden Systems und der Vorgang der Regelauswertung mit Esper zusammengefasst (vgl. Abschnitt 7.4.2).

7.4.1 EPL-basierte Management-Regeln

In Abschnitt 6.3.5 wurde eine abstrakte Beschreibung eines Managementdokuments vorgestellt, welches die durch den Auftraggeber einer entfernt ausgeführten Prozesspartition vordefinierten Rahmenbedingungen repräsentiert. Dabei wurde offen gelassen, welche Art regelbasiertes System die Auswertung der Managementregeln übernimmt und wie diese konkret beschrieben werden. Als Machbarkeitsnachweis wird daher in diesem Abschnitt die Verwendung von Ausdrücken der *Event Processing Language* (*EPL*) [Esp11] zur Beschreibung von Ereignismustern und deren Integration mit den durch WSDM definierten Managementfunktionalitäten aufgezeigt.

Die von Esper verwendete *Event Processing Language* (*EPL*) erlaubt Abfragen über Ereignisströme (*Event Stream Queries*) und über Muster zwischen Ereignissen (*Event Patterns*). Die Abfragen über Ereignisströme werden über eine SQL-basierte Syntax realisiert. Im Gegensatz zu SQL werden die Quelldaten hierbei jedoch nicht aus einer Datenbank bezogen, sondern aus Datenströmen, welche einzelne Ereignisse oder Ereignis-Tupel enthalten (vgl. Abschnitt 3.7.5). Anstelle von Tabellen werden zudem Sichten (*Views*) verwendet, um das Eintreten von Ereignissen in einem angegebenen Zeit- oder Mengenfenster zu filtern und/oder zu aggregieren. Listing 7.1 zeigt die allgemeine Syntax von EPL-Anfragen im Überblick. Als wichtigste Klausel spezifiziert die *SELECT*-Anweisung die für die Abfrage relevanten Ereignisse bzw. die relevanten Eigenschaften der Ereignisse. Die *FROM*-Klausel gibt die Ereignisströme bzw. deren Benennung an. Die *WHERE*-Klausel beschreibt die Bedingungen, nach denen ein Ereignis oder eine Kombination von Ereignissen gesucht werden soll. Dabei können wie in SQL Vergleichs- und Verknüpfungsoperatoren verwendet werden, um Ereignisströme miteinander in Beziehung zu setzen oder einzuschränken [Esp11].

In einer *FROM*-Klausel können *filterbasierte* oder *musterbasierte* Ereignisströme angegeben werden (vgl. Listing 7.2). Filterbasierte Ereignisströme können zum Beispiel durch die Angabe des gesuchten Ereignistyps definiert werden und können zur weiteren Einschränkung bestimmte Eigenschaften der empfangenen Ereignisse als Filterkriterien angeben. Durch die Definition von musterbasierten Ereignisströmen können in eine *FROM*-Klausel jedoch auch Ereignismuster (*Event Patterns*) eingebettet werden. Die Einbettung eines Er-

```
1  [annotations]
2  [expression_declarations]
3  [CONTEXT context_name]
4  [INSERT INTO insert_into_def]
5  SELECT select_list
6  FROM stream_def [AS name] [, stream_def [AS name]] [,...]
7  [WHERE search_conditions]
8  [GROUP BY grouping_expression_list]
9  [HAVING grouping_search_conditions]
10 [OUTPUT output_specification]
11 [ORDER BY order_by_expression_list]
12 [LIMIT num_rows]
```

Listing 7.1: Allgemeine Syntax von EPL-Abfragen (*Event Stream Queries*) (Esp11)

```
1  stream_def = event_stream_name [(filter_criteria)] |
2               PATTERN [pattern_expression]
```

Listing 7.2: Definition von filterbasierten oder musterbasierten Ereignisströmen innerhalb der FROM-Klausel (vgl. Zeile 6 in Listing 7.1) (Esp11)

eignismusters wird durch das Schlüsselwort *PATTERN* angegeben. Für jedes Ereignismuster (*Pattern Expression*) können sogenannte *Atome* (d. h. einzelne Ereignisfilter, zeitbasierte Ausdrücke oder benutzerdefinierte Ausdrücke) durch die Verwendung von *Operatoren* miteinander in Beziehung gesetzt werden. Es existieren fünf Arten von Operatoren [Esp11]:

- Operatoren, welche die Wiederholung von (Teil-)Ausdrücken eines Ereignismusters angeben: EVERY, EVERY-DISTINCT, [NUM], UNTIL.
- Logische Operatoren: AND, OR, NOT.
- Zeitliche Operatoren, welche eine Folge von Ereignissen ausdrücken: ->(„followed-by“).
- Bedingungen, welche die Lebensdauer von (Teil-)Ausdrücken eines Ereignismusters kontrollieren (*Guards*), u.a.: TIMER:WITHIN, TIMER:WITHINMAX, WHILE.
- Bedingungen, welche Zeitereignisse beobachten (*Observer*), u.a.: TIMER:INTERVAL, TIMER:AT.

Listing 7.3 zeigt die Einordnung der EPL-Ausdrücke in die Management-Regeln. Dabei wird der abstrakte *Rule-Pattern*-Teil der Management-Regeln durch eine gültigen EPL-Abfrage (*EPL-Query*) implementiert. Im Kontext des Managementdienstes werden für eine automatische Ableitung von Anpassungsbedarf die durch die Ereignisschnittstelle des Prozessmanagementsystems als verwaltbare Ressource spezifizierten Ereignistypen verwendet (vgl.

```
1  MANAGEMENT-RULES  : {MANAGEMENT-RULE}
2    MANAGEMENT-RULE
3      NAME          : <String>
4      [DESCRIPTION  : <String>]
5      RULE-PATTERN  : <EPL-Query>
6      RULE-ACTION   : <EPL-Query-based Service-Invocation>
7    END MANAGEMENT-RULE
8    ...
9  END MANAGEMENT-RULES
```

Listing 7.3: Einbettung von EPL-Ausdrücken in Management-Regeln (vgl. Listing 6.1 in Abschnitt 6.3.5)

```
1  RULE-ACTION
2    SERVICE
3      ENDPOINT-REFERENCE : <URI>
4      OPERATION          : <Name of Operation>
5      PARAMETERS         : {PARAMETER}
6        PARAMETER
7          TYPE           : <Name of Parameter>
8          VALUE          : <Value>  | %<EPL-Expression>
9        END PARAMETER
10       ...
11     END PARAMETERS
12   END SERVICE
13 END RULE ACTION
```

Listing 7.4: Syntax für die Beschreibung eines Dienstaufrufs als Reaktionsteil der Managementregel (vgl. Zeile 6 in Listing 7.3)

Abschnitt 7.3). Die Ereignistypen, ihre Bedeutung und ihre Eigenschaften sind im Rahmen von WSDM bzw. durch das Referenzmodell eindeutig definiert. Die Ereignisse können daher durch die Angabe ihrer URI abstrakt in der Regelbeschreibung integriert werden und analog zu bzw. in Kombination mit weiteren anwendungsspezifischen Ereignissen empfangen und verarbeitet werden.

Aufgrund der Tatsache, dass die Reaktion auf ein eingetretenes Ereignis jedoch unter Umständen konkrete Informationen dieses Ereignisses berücksichtigen muss (z. B. Korrelationsdaten zur Identifikation einer Prozessinstanz), ist es zudem notwendig, im *Rule-Action*-Teil der Managementregel ebenfalls auf die Ergebnisse der EPL-Abfrage zugreifen zu können. Daher muss auch zumindest der Teil zur Beschreibung der Aufrufparameter an die Syntax der verwendeten EPL angepasst werden und kann nicht vollständig unabhängig von der verwendeten Sprache zur Beschreibung der Ereignismuster sein. Listing 7.4 zeigt eine einfache Syntax zur Beschreibung des Reaktionsteils unter Berücksichtigung von Ergebnissen der EPL-Abfrage für die Zu-

```
<Rule>
  <Name>Expiration</Name>
  <Trigger>
     SELECT * FROM PATTERN
     [EVERY (e1=org.vsis.pmaas.ActivityStateChanged(modelId="1",activityId="4",
                state="executing")
             -> TIMER:INTERVAL(30 MINUTES)
             -> NOT e2=org.vsis.pmaas.ActivityStateChanged(modelId="1",activityId="4",
                       state="executed")
     ]
     WHERE e1.instanceId=e2.instanceId
  </Trigger>
  <Action>
     <Service epr="http://vsis.informatik.uni-hamburg.de/demac/pmaas.wsdl"
              operation="CancelProcess">
        <Param type="ProcessInstanceID"><Value>%{e1.instanceId}</Value></Param>
    </Service>
  </Action>
</Rule>
```

Listing 7.5: Konkretes Beispiel einer Management-Regel mit EPL-Ereignismuster und Dienstaufruf der Managementschnittstelle (XML-Repräsentation)

weisung von Werten zu Parametern und deren dynamischer Einbindung zur Laufzeit, d. h. zum Zeitpunkt des Eintretens des komplexen Ereignisses.

Listing 7.5 zeigt ein (vereinfachtes) konkretes Beispiel für eine Management-Regel in XML-Repräsentation, welche als Ereignismuster (*Trigger*) einen EPL-Ausdruck enthält. Dieser EPL-Ausdruck repräsentiert eine Regel, welche feuert, wenn bei einer beliebigen Prozessinstanz des Prozessmodells mit der ID=1 die Ausführung der Aktivität mit der ID=4 länger dauert als 30 Minuten. Die fettgedruckten Anweisungen und Operatoren innerhalb des *Trigger*-Elements sind Sprachkonstrukte der EPL. Die nicht fettgedruckten Objekte innerhalb des *Trigger*-Elements stellen die Ereignisse und deren Eigenschaften dar, welche durch das Referenzmodell des Prozessmanagementsystems als verwaltbare Ressource spezifiziert sind. Als Reaktion auf das Eintreten des so festgelegten komplexen Ereignisses wird in dem dargestellten Beispiel die Operation *CancelProcess* am vorgegebenen Dienstzugriffspunkt (*EPR*) aufgerufen. Dabei wird durch den Dienstaufruf als exemplarische Anpassungsmaßnahme die Prozessinstanz, für die das im Ereignismuster definierte Ereignis eingetreten ist, abgebrochen.

7.4.2 Ereignisverarbeitung mit Esper

Esper ist eine javabasierte Plattform für die Verarbeitung von Ereignisströmen (*Event Stream Processing*) und zur Korrelation von Ereignissen (*Complex Event Processing*) [Esp11, Esp12]. Insbesondere stellt Esper durch die Implementierung eines Zustandsautomaten (*State Machine*) die Funktionalität der Mustererkennung für die Ereignisverarbeitung zur Verfügung. Der hier vorgeschlagene Managementdienst nutzt diese Funktionalität, um die in den Managementregeln definierten EPL-Abfragen auf eingehende Sequenzen von (einfachen oder komplexen) Ereignissen anzuwenden. Die für die Mustererkennung

verantwortliche zentrale Verarbeitungseinheit von Esper wird im Folgenden als *Esper-Engine* bezeichnet.

Die aus der Integration von Esper resultierende Funktionsweise des Managementdienstes ist (vereinfacht) Abbildung 7.12 zu entnehmen. Wird durch den Auftraggeber einer auszuführenden Prozesspartition ein neues Management-Dokument (*MD*) mit EPL-basierten Management-Regeln an den Managementdienst übergeben (Vorgang ①), so werden die in den Management-Regeln gekapselten EPL-Abfragen als neues *Statement* bei der Esper-Engine registriert. Ein *Statement* stellt dabei eine an Esper übergebene Abfrage dar, welche die Benachrichtigung einer *Listener*-Komponente auslöst, sobald Ereignisse eintreten, welche die Ergebnismenge des Statements verändern. Alternativ können die Ergebnisse der Abfrage bei Bedarf über ein API nach dem Pull-Modell zugegriffen werden [Esp11]. Damit Ereignisse verarbeitet werden können, muss der Managementdienst zudem die im Management-Dokument spezifizierten Ereignistypen abonnieren und die eingehenden Ereignisse dem ereignisverarbeitenden System in geeigneter Form zur Verfügung stellen. Für Esper wird dabei eine Abbildung der Ereignisse auf Java-Objekte vorgenommen.

Durch den Aufruf der für das (komplexe) Ereignis spezifizierten *Listener*-Komponente wird automatisch die Bearbeitung des Aktionsteils der Managementregel ausgelöst. Dazu werden die im Management-Dokument enthaltenen Informationen zum Dienstaufruf und die Ergebnismenge der Abfrage zu einem konkreten Dienstaufruf verarbeitet (*Generic Action Executor*). Sofern keine (Zeit-)Beschränkungen für die Gültigkeit der EPL-Abfrage angegeben wurden, bleibt die Abfrage nach Auslösen der Regel weiter aktiv. Sie würde daher bei jedem weiteren Eintreten der spezifizierten Zusammenhänge erneut auslösen und es würde jedes Mal die entsprechend definierte Reaktion ausgeführt werden. Das Metamodell für die Spezifikation des Management-Dokuments stellt prinzipiell weitere Optionen bereit, um die Beendigung des Managements automatisch durchzuführen (vgl. Abschnitt 6.3.5). Alternativ können die Management-Regeln jedoch auch zur Laufzeit dynamisch angepasst oder entfernt werden. Dies ist für die hier angestrebte Flexibilisierung besonders wichtig, da nur durch die Möglichkeit zur *fortwährenden Anpassbarkeit* ein Kompromiss zwischen Offline-Management und Ad-hoc-Anpassungen erzielt werden kann.

Der Vorgang ② in Abbildung 7.12 zeigt die prinzipielle Vorgehensweise zur Änderung von Management-Regeln (d. h. zum Löschen, Hinzufügen und Anpassen von Ereignismustern und Aktionen) sowie zur vollständigen Beendigung des Managements. Für eine ausschließliche Änderung am Aktionsteil einer Management-Regel ist keine Interaktion mit Esper notwendig. Die geänderten Aktionen werden mit dem gesamten Management-Dokument im *Document Store* gespeichert, wo sie beim nächsten Auslösen einer Regel abgefragt und direkt ausgeführt werden können. Zur Anpassung von Ereignismustern wird die Registrierung von nicht länger benötigten EPL-Abfragen bei der Esper-Engine beendet. Zudem werden die Abonnements für die ent-

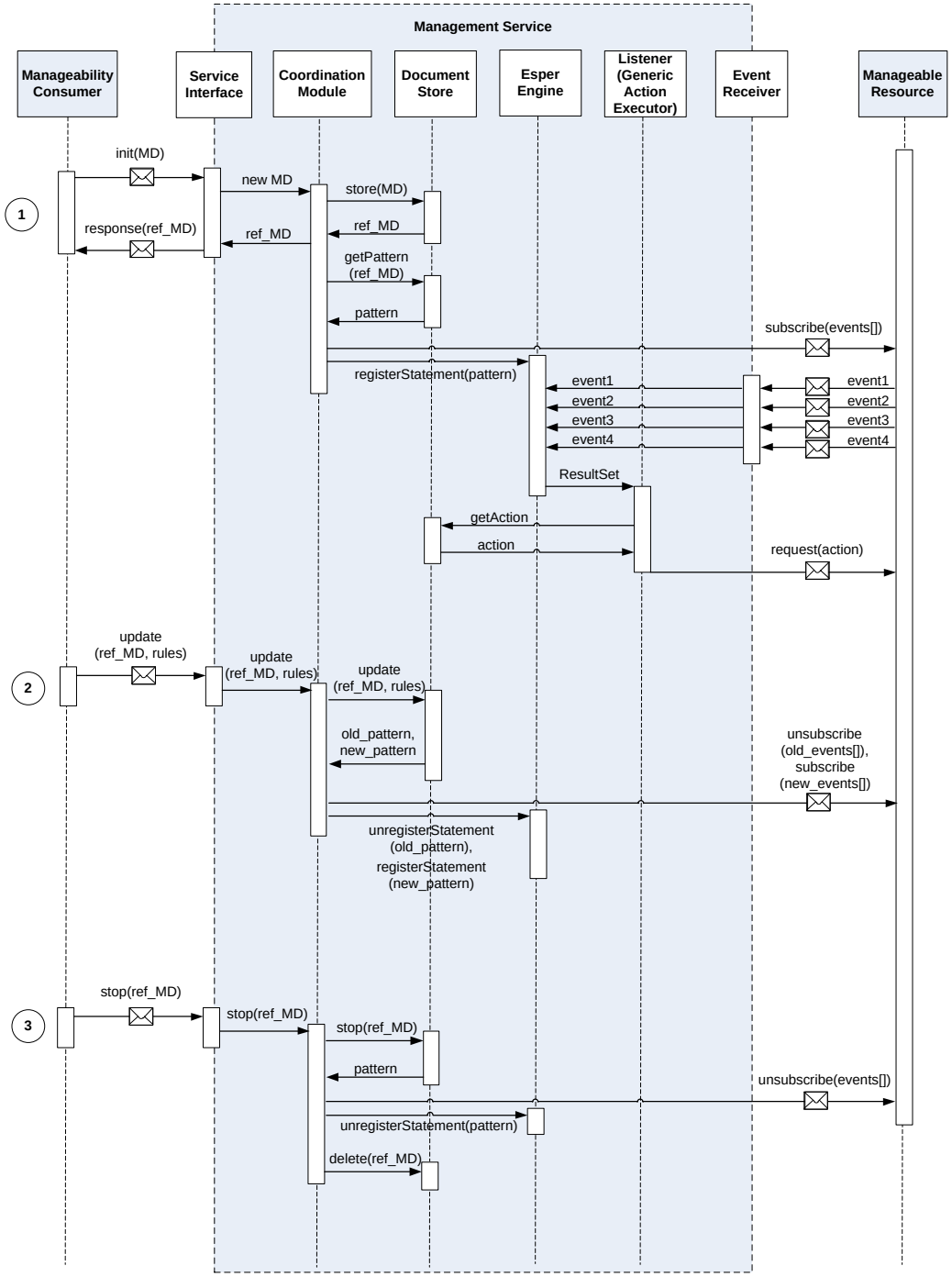


Abbildung 7.12: Implementierung und Interaktion des Managementdienstes mit Esper (tlw. nach (Str09))

sprechenden Ereignisse zu den gelöschten Abfragen beendet, sofern die Ereignisse nicht noch in anderen (aktiven) Ereignismustern spezifiziert sind. Analog dazu werden neue EPL-Abfragen bei der Esper-Engine registriert und die hierfür zusätzlich benötigten Ereignisse abonniert. Die Änderung einer einzelnen bestehenden EPL-Abfrage ist leider nicht möglich. Hier muss die alte EPL-Anfrage entfernt und durch die neue Abfrage ersetzt werden. Die bisher zu dieser Abfrage intern gesammelten (Teil-)Ergebnismengen gehen dabei verloren.

Der Vorgang ③ in Abbildung 7.12 zeigt schließlich die prinzipielle Vorgehensweise zur manuellen Beendigung des Managements. Hierbei wird das Abonnement aller im Management-Dokument spezifizierten Ereignisse abbestellt und die Registrierung aller EPL-Regeln in der Esper-Engine beendet. Schließlich wird das Management-Dokument aus dem *Document Store* entfernt.

Die Umsetzung des Managementdienstes mit Esper zeigt, dass eine flexible Anpassung automatisiert ablaufender Management-Regeln für die Überwachung und Steuerung entfernt ausgeführter Prozesspartitionen möglich ist. Eine Anpassung einzelner Prozessmodelle oder -instanzen ist hierfür nicht notwendig. Der Managementdienst kann dabei sowohl auf Seiten des Auftraggebers als auch auf Seiten der verwaltbaren Ressource implementiert werden, so dass die in Abbildung 7.12 dargestellte Übertragung von Nachrichten entweder nur zwischen Konsument und Managementdienst oder nur zwischen Managementdienst und verwaltbarer Ressource auftritt. Das Implementierungsbeispiel zeigt jedoch auch, dass bei einer Nutzung des Managementdienstes auf der Seite einer verwaltbaren Ressource auch die dort verwendete Sprache zur Spezifikation von Regelausdrücken (hier EPL) verwendet werden muss. Da es hierfür noch keinen im Rahmen des Prozessmanagements empfohlenen Standard gibt, ist die Flexibilität in einem Szenario mit vielen im Voraus unbekannten Teilnehmern der Prozessausführung aufgrund der mangelnden Interoperabilität an dieser Stelle eingeschränkt. Eine maximale Flexibilität kann daher nur bei Vorliegen einer gemeinsam verwendeten Prozessbeschreibungssprache und einer gemeinsam verwendeten Sprache zur Definition von Management-Regeln erreicht werden.

7.5 Migrationsdienst

In Abschnitt 6.2 wurde ein allgemeines Verfahren zur dynamischen Verteilung von prozessorientierten Anwendungen auf Basis einer logischen Partitionierung von Prozessen durch die Migration von laufenden Prozessinstanzen vorgestellt. Voraussetzung hierfür ist die öffentliche Bereitstellung der Funktionalität zur Prozessausführung für einzelne Prozessinstanzen unter Berücksichtigung ihres aktuellen Ausführungsfortschritts. Die für eine verteilte Ausführung zur Verfügung stehenden Prozessmanagementsysteme werden dazu durch einen Dienst gekapselt, welcher analog zum Konzept des *Process-Management-as-a-Service* (*PMaaS*) das Prozessmodell als

Ausführungsanweisung für die Prozessinstanz und zusätzlich die Instanz der dazugehörigen Migrationsdaten nach dem in Abschnitt 6.2.2 eingeführten Migrationsmodell als Aufrufparameter entgegennimmt. Um den verteilt auszuführenden Prozess auf der jeweils aktuellen Ausführungseinheit zu installieren, fortzuführen und nach der Ausführung der vorgegebenen Prozesspartition(en) ggf. wieder zu deinstallieren, kann ein solcher *Migrationsdienst* dazu als lokaler Konsument der unter Abschnitt 7.3 vorgestellten Überwachungs- und Steuerungsfunktionalität auftreten. In diesem Abschnitt wird auf dieser Basis eine Architektur für den Migrationsdienst zur dynamischen Verteilung von Prozessinstanzen vorgestellt. Alternativ kann der Migrationsdienst jedoch auch direkt auf dem jeweiligen Prozessmanagementsystem aufsetzen. Ein kurzes Beispiel hierfür wird in Abschnitt 8.2.3 aufgezeigt.

Abbildung 7.13 zeigt die vorgeschlagenen Komponenten des Migrationsdienstes im Überblick [ZHKK10]. Für den Transfer der Prozessbeschreibung und der Instanzdaten des verteilt auszuführenden Prozesses sowie für die Durchführung von Kommunikationsvorgängen zur Koordination im Fall von verteilt parallel ausgeführten Prozesspartitionen stellt der Migrationsdienst eine einheitliche Schnittstelle bereit (*PMaaS service interface*). Diese kann zum Beispiel durch einen Web Service oder durch eine andere an die (ggf. mobilen) Konsumenten angepasste Kombination von Beschreibungen und Protokollen repräsentiert werden (vgl. Abschnitt 7.2). Im einfachsten Fall existiert mindestens eine Operation, welche das Prozessmodell des (verteilt) auszuführenden Prozesses PM_m in direkt ausführbarer Repräsentation und die dazugehörigen Migrationsdaten $M_{m,i}$ mit den Instanzinformationen des Prozesses als Eingabeparameter entgegennimmt [ZHKK10]. Wurde der auszuführende Prozess mit Sicherheitsmechanismen versehen, welche von der betrachteten Ausführungsumgebung unterstützt werden, so müssen die über die Schnittstelle erhaltenen Eingabedaten zunächst entschlüsselt bzw. auf ihre Authentizität und/oder Autorisierung überprüft werden. Diese Aufgaben werden von einer optionalen, dem eigentlichen Migrationsdienst vorgeschalteten Sicherheitskomponente (*Privacy manager*) wahrgenommen. Das Ergebnis ist ein für die Ausführung autorisiertes entschlüsseltes Prozessmodell PM_m und ein entsprechendes Migrationsdatendokument $M_{m,i}$, welche zusammen analog zu ungesicherten Prozessen weiterverarbeitet werden können [ZHKK10].

Die für die Ausführung bzw. die Migration einer Prozessinstanz erforderliche Vor- bzw. Nachbereitung wird durch eine Komponente vorgenommen, welche einerseits das Prozessmodell auf der unterliegenden Prozess-Engine installiert und andererseits die Informationen aus den zugehörigen Migrationsdaten ausliest und verarbeitet (*Migration manager*). Hierbei werden zunächst die benutzerdefinierten Rahmenbedingungen für die verteilte Ausführung (vgl. Abschnitt 6.2.2) überprüft und festgestellt, ob diese von der lokalen Prozess-Engine erfüllt werden. Liegen alle Voraussetzungen für eine Weiterführung der Prozessinstanz vor, wird die Prozessinstanz $P_{m,i}$ in ihrem aktuellen Zustand wieder hergestellt, über die Managementschnittstelle (*Management interface*, vgl. Abschnitt 7.3) an die als verwaltbare Ressource gekap-

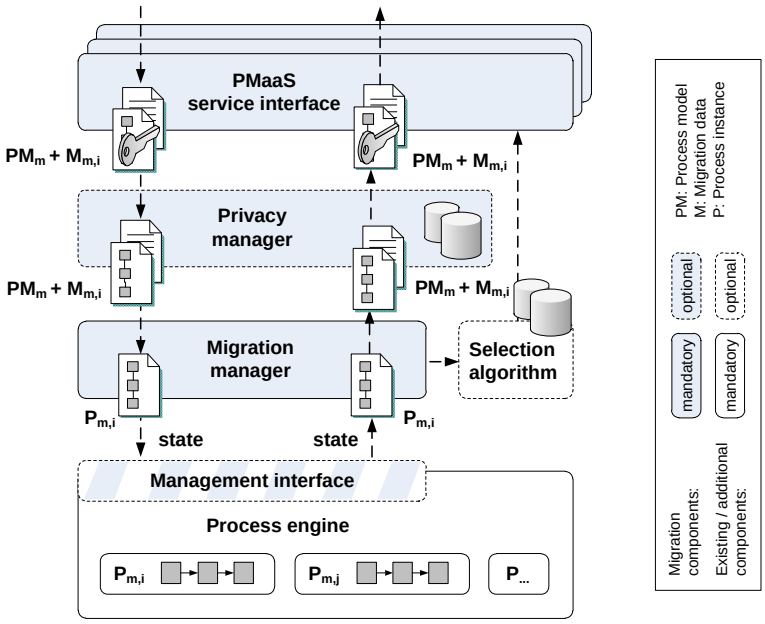


Abbildung 7.13: Komponenten des Migrationsdienstes zur dynamischen Verteilung von Prozessinstanzen (Übersicht) (ZHKK10))

selte Prozess-Engine übergeben und deren Ausführung wieder aufgenommen. Bei erfolgreicher Übernahme des Prozesses erhält der Aufrufer des Migrationsdienstes als Rückgabe einen eindeutigen Identifikator, über den der auszuführende Prozess auf der Zielausführungsumgebung wieder aufzufinden ist [ZHKK10].

Liegen benutzerdefinierte Rahmenbedingungen aus den Migrationsdaten oder aus lokalen Richtlinien zur Ausführung von (migrierten) Prozesse vor, so ist die Migrationskomponente zudem dafür verantwortlich, während der Ausführung des Prozesses über die Einhaltung dieser Richtlinien zu wachen. Die für die Ableitung von Anpassungsbedarf erforderliche Funktionalität kann einerseits über die lokale Management-Schnittstelle und andererseits über die Kommunikation mit anderen Ausführungseinheiten bezogen werden. Wird dabei festgestellt, dass die Ausführung des Prozesses durch eine andere Ausführungseinheit fortgesetzt werden soll, wird die lokale Ausführung des Prozesses beendet und die erforderlichen Migrationsdaten aus der Prozessinstanz $P_{m,i}$ generiert. Soll der auszuführende Prozess mit Sicherheitsmechanismen versehen werden, wird die Sicherheitskomponente verwendet, um eine entsprechend der definierten Rahmenbedingungen verschlüsselte Form des Prozessmodells und der Migrationsdaten zu erzeugen. Schließlich wird durch den Aufruf eines festgelegten oder ausgewählten Migrationsdienstes der Prozess zur Fortführung an eine andere Ausführungseinheit transferiert [ZHKK10].

Durch die Umsetzung der Schnittstelle zur Migration laufender Prozessinstanzen kann das Problem der Auswahl einer geeigneten Ausführungsumgebung zur Fortführung des Prozesses auf die Suche und

Auswahl von (funktionalen) Diensten im Allgemeinen zurückgeführt werden. Dementsprechend können hierfür bestehende Verfahren zur dynamischen Dienstsuche und -auswahl verwendet werden. Entsprechende Algorithmen (*Selection algorithms*) sind daher nicht Inhalt dieser Arbeit. Eine mögliche Umsetzung der Dienstauswahl für die Ausführung von Prozessen in dynamischen Umgebungen unter spezieller Berücksichtigung nicht-funktionaler Aspekte kann unter anderem einer weiterführenden Arbeit [HSZ11] entnommen werden.

Im Folgenden werden die einzelnen Komponenten des Migrationsdienstes im Detail betrachtet. Der Schwerpunkt liegt dabei auf der Realisierung der Migrationskomponente (*Migration Manager*), da dieser als obligatorischer Bestandteil die Hauptaufgabe für die Realisierung dynamisch verteilt ausgeführter Prozesse zukommt.

7.5.1 PMaaS-Schnittstelle

Der Migrationsdienst enthält eine öffentliche *Process-Management-as-a-Service*-Schnittstelle, über welche zum einen die Migration von (laufenden) Prozessinstanzen an andere (ausgewählte) Ausführungseinheiten erfolgen kann und zum anderen die Suche und Koordination verteilt parallel ausgeführter Prozesspfade unterstützt wird (vgl. Tabelle 7.13). Wesentlicher Bestandteil ist die Operation *Execute instance*, welche das Prozessmodell und die Instanzdaten nach dem spezifizierten Migrationsmodell entgegennimmt. In den folgenden beiden Unterabschnitten werden diese beiden wichtigsten Eingabeparameter des Migrationsdienstes näher erläutert. Die anderen Operationen der Schnittstelle werden bei der Beschreibung des Migrationsmanagers in Abschnitt 7.5.3 wieder aufgegriffen.

Eingabeparameter Prozessmodell

Das Prozessmodell liegt als ausführbare Prozessbeschreibung nach einem (im Idealfall standardisierten) Metamodell vor und entspricht dem im Rahmen des Deployments für eine nicht-verteilte Ausführung an die Prozess-Engine übergebenen Material. Dieses Material muss nicht zwingend aus einem einzelnen Dokument bestehen, sondern kann mehrere Bestandteile (wie zum Beispiel Skripte, Benutzeroberflächen oder Schnittstellenbeschreibungen) umfassen, deren Zusammenhänge in der Prozessbeschreibung selbst oder in einem zusätzlichen *Deployment-Descriptor* beschrieben sein können. Als Vereinfachung wird für die hier vorliegende Implementierung zunächst angenommen, dass ein etwaiger Deployment-Descriptor von allen teilnehmenden Ausführungseinheiten auf die gleiche Weise verarbeitet werden kann. Dies ist zum Beispiel der Fall, wenn alle Teilnehmer der verteilten Prozessausführung die gleiche Art von Prozess-Engine verwenden. Alternativ muss der Deployment-Descriptor in eine plattformunabhängige Repräsentation übersetzt werden. Da diese Repräsentation jedoch immer

Operation	Beschreibung
Execute instance	**Migration**: Übergibt ein Prozessmodell in Form einer ausführbaren Prozessbeschreibung und ein nach Abschnitt 6.2.2 definiertes Migrationsmodell, um die darin definierte Prozessinstanz auf der durch den Migrationsdienst gekapselten Ausführungseinheit (weiter) auszuführen. Die Eingabeparameter können im Klartext oder verschlüsselt übergeben werden. Als Rückgabe erfolgt im Erfolgsfall eine Referenz auf die installierte Prozessinstanz.
Migrate instance	**Migration**: Übergibt den Identifikator einer auf der lokalen Prozess-Engine zentral ausgeführten Prozessinstanz und optional ein nach Abschnitt 6.2.2 definiertes Migrationsmodell, um die angegebene Prozessinstanz nach den ggf. angegebenen benutzerdefinierten Rahmenbedingungen oder etwaigen lokalen Richtlinien verteilt auszuführen. Werden keine Migrationsdaten übergeben, so werden diese bei Aufruf der Operation generiert und dem Aufrufer als Rückgabewert zur weiteren Verfeinerung übergeben. Werden Migrationsdaten übergeben, so werden diese direkt angewendet.
Migrate all instances	**Migration**: Übergibt den Identifikator eines auf der lokalen Prozess-Engine installierten Prozessmodells und optional ein nach Abschnitt 6.2.2 definiertes Migrationsmodell, um alle nachfolgend von diesem Prozessmodell abgeleiteten Prozessinstanzen nach den ggf. angegebenen benutzerdefinierten Rahmenbedingungen oder etwaigen lokalen Richtlinien verteilt auszuführen. Werden keine Migrationsdaten übergeben, so werden diese bei Aufruf der Operation generiert und dem Aufrufer als Rückgabewert zur weiteren Verfeinerung übergeben. Werden Migrationsdaten übergeben, so werden diese direkt angewendet.
Find instance	**Suche**: Übergibt den eindeutigen Identifikator der migrierten Prozessinstanz und optional eine Liste von Identifikatoren von Aktivitäten dieser Prozessinstanz, um festzustellen, ob diese Aktivitäten auf der lokalen Prozess-Engine ausgeführt werden. Als Rückgabe erfolgt je nach Ergebnis der Anfrage eine Fehlanzeige, eine Erfolgsanzeige oder – im Fall einer bereits erfolgten Migration des angefragten Prozesses – der Identifikator der Ausführungseinheit, an welche der Prozess migriert wurde.
Synchronize	**Synchronisation**: Fordert zur Zusammenführung von Replikaten einer bestimmten Prozessinstanz auf der angegebenen Ausführungseinheit auf.
Receive event	**Ereignisverarbeitung**: Generische Schnittstelle zum Weiterleiten von Ereignissen.
Request conflict resolution	**Optimistische Konfliktauflösung**: Anfrage, ob die für eine bestimmte Prozessinstanz angegebene Konfliktauflösung (Präzedenz) mit den etwaigen lokal gewählten optimistischen Ausführungen anderer paralleler Ausführungspfade übereinstimmt (vgl. Abschnitt 6.2.5). Als Rückgabe erfolgt eine Bestätigung der eingegebenen Präzedenz oder ein Widerspruch unter Angabe der sich in Widerspruch befindlichen Präzedenzen.
Apply conflict resolution	**Optimistische Konfliktauflösung**: Fügt die für eine bestimmte Prozessinstanz angegebene Präzedenz für die optimistische Konfliktauflösung hinzu.
Cancel conflict resolution	**Optimistische Konfliktauflösung**: Bricht die optimistische Konfliktauflösung mit der für eine bestimmte Prozessinstanz angegebenen Präzedenz ab.

Tabelle 7.13: Schnittstelle des Migrationsdienstes

noch von der verwendeten Prozessbeschreibungssprache und etwaigen Erweiterungen abhängig sein kann, kann hierfür keine allgemeingültige Struktur vorgegeben werden. Die hier vorgeschlagene Umsetzung der Migrationsdaten enthält daher ein optionales, generisches Element *Deployment-Descriptor*, welches im Bedarfsfall entsprechende Informationen für einen plattformübergreifenden Austausch von Prozessmodellen aufnehmen kann. Ein Beispiel hierfür wird in Abschnitt 8.2.2 vorgestellt.

Eingabeparameter Migrationsdaten

Der Aufbau der Migrationsdaten als XML-Repräsentation des in Abschnitt 6.2.2 vorgeschlagenen (Meta-)Modells ist schematisch in Listing 7.6 dargestellt. Dabei wurde insbesondere berücksichtigt, dass unter Umständen nicht alle (potentiell) an einer verteilten Prozessausführung teilnehmenden Ausführungseinheiten über Möglichkeiten zur Auswertung des gesamten Mi-

grationsmodells verfügen müssen (z. B. leistungsbeschränkte mobile Endgeräte). Die Migrationsdaten enthalten hierzu zunächst einen Prolog (*Prologue*), welcher die grundlegenden obligatorischen Informationen beinhaltet, die von allen teilnehmenden Ausführungseinheiten verstanden und verarbeitet werden können müssen. Mit der (alleinigen) Fähigkeit zur Verarbeitung des Prologs kann damit bereits das einfache oder gezielte Weiterleiten von Prozessen unterstützt werden, ohne den Prozess auf der Prozess-Engine installieren und im Kontrollfluss voranschreiten zu müssen. Der Prolog gibt zunächst einen global eindeutigen Identifikator (*UUID*) zur eindeutigen Identifikation der migrierten Prozessinstanz an. Dieser wird zum einen bei der Kommunikation zur Übertragung des Prozesses verwendet, besitzt jedoch auch für die verteilte parallele Ausführung von Prozessen eine große Relevanz, um zu kennzeichnen, dass es sich bei mehreren verteilt ausgeführten Replikaten eines Prozesses um dieselbe Prozessinstanz handelt. Desweiteren wird im Prolog die verwendete Prozessbeschreibungssprache (*Process Description Language*, kurz *PDL*) des Prozessmodells aufgeführt. Die Angabe der Prozessbeschreibungssprache ist zur Auswahl einer geeigneten Prozess-Engine erforderlich, welche die Verarbeitung dieser Prozessbeschreibung unterstützt. Prozesse, deren Prozessbeschreibungssprache auf der ausgewählten Engine nicht unterstützt werden, können damit direkt zurückgewiesen werden, ohne den Versuch einer Installation vornehmen zu müssen. Zum anderen können auf diese Weise auch Prozess-Engines gekapselt werden, welche verschiedene Prozessbeschreibungssprachen unterstützen. Als optionales Element enthält der Prolog zudem die Angabe der als nächsten aufzurufenden Ausführungseinheit (*NextParticipant*). Ist das Element undefiniert oder stimmt beim Aufruf des Migrationsdienstes der Wert dieses Elements mit dem eigenen Identifikator überein (in der Implementierung als UUID umgesetzt), so wird der migrierte Prozess weiter verarbeitet und bei Vorliegen aller weiter spezifizierten Rahmenbedingungen auf dieser Prozess-Engine weiter ausgeführt. Stimmt der Identifikator nicht überein, wird der Prozess nach Möglichkeit an die angegebene Ausführungseinheit weitergeleitet. Diese Strategie des „Zwischenlagerns“ von Prozessen erlaubt zum Beispiel die Durchsetzung von Sicherheitseigenschaften, wenn der zuletzt ausführenden Organisation die nachfolgende Ausführungseinheit und ihre organisatorische Zuordnung nicht bekannt werden soll. In diesem Fall wird das initial verschlüsselte Element der nachfolgend aufzurufenden Ausführungseinheit als *NextParticipant* gesetzt und der Prozess an einen vertrauenswürdigen Dritten weitergeleitet, welcher das Element entschlüsseln und die gewünschte Zuordnung vornehmen kann. Schließlich erlaubt die Angabe der für die Fortführung des Prozesses angegebenen Ausführungseinheit das gezielte Weiterleiten von Prozessen in nicht dauerhaft zusammenhängenden, mobilen Netzen (z. B. in mobilen Ad-hoc-Netzen), ohne dass durch leistungsbeschränkte Teilnehmer eine Auswertung komplexer Regelwerke zur Identifikation nachfolgender Teilnehmer vorgenommen werden muss.

```
Migrationdata
  Prologue
    UUID
    PDL
    [NextParticipant]
  End Prologue

  [Deployment-Descriptor]

  State
    ProcessState
    Startactivities
      {Startactivity}
    End Startactivities
    ActivityStates
      {Element}
        ID
        ActivityState
      End Element
    End ActivityStates
    VariableValues
      {Variable}
        Name
        Value
      EndVariable
    End VariableValues
  End State

  [Dataclasses
    {Variable}
      Name
      Dataclass
  End Dataclasses]

  [Synchronisation
    {Precedence}
  End Synchronisation]

  [AssignmentStrategy
    [ProcesslevelAssignment
        Assignment
          StrategyID
          StrategyImpl
        End Assignment
    End ProcesslevelAssigment]
    [ElementlevelAssignment
      {Element}
        ID
        Assignment
          StrategyID
          StrategyImpl
        End Assignment
      End Element
    End ElementlevelAssignment]
  End AssignmentStrategy]

  [Privacy
    [Signature]
    [Encryption]
  End Privacy]

  [Log]
End Migrationdata
```

Listing 7.6: Aufbau der an der Schnittstelle übergebenen Migrationsdaten (vgl. Abbildung 6.6)

Als optionales Element der Migrationsdaten folgt als nächstes der *Deployment-Descriptor* zum Transport der oben erläuterten plattformübergreifenden Deployment-Informationen. Nach erfolgter Installation des Prozessmodells werden die Zustandsdaten ausgewertet (*State*). Hierbei werden nach dem in Abschnitt 6.2.2 angegebenen (Meta-)Modell der Zustand der Prozessinstanz (*ProcessState*, bei einer Migration immer im Zustand *transferring*), die Liste der Startaktivitäten (*Startactivities*), die Zustände der einzelnen Aktivitäten (*ActivityStates*) und die aktuellen Werte der Prozessvariablen (*VariableValues*) angegeben. Wie im Konzept für die dynamische Verteilung von Prozessinstanzen vorgeschlagen wurde (vgl. Abschnitte 6.2.4, 6.2.5 und 6.2.6) gelten hierbei auch alle durch die Prozess-Engine selbst ausgeführten bzw. ausgewerteten Kontrollflusskonstrukte als „Aktivitäten" im Sinne der Migration. Es werden daher unter *ActivityStates* alle Elemente (*Elements*) der Prozessbeschreibung aufgezählt, für die – in Abhängigkeit der Prozessbeschreibungssprache – prinzipiell die Charakterisierung durch eine Zustandsangabe und eine Zuweisung zu einer bestimmten Ausführungseinheit möglich ist.

Die optionale Angabe von Datenklassen zur Spezifikation des Synchronisationsverhaltens von Variablen ist von der Zustandsangabe getrennt aufgeführt (*Dataclasses*). Prozessvariablen, welche nicht in (potentielle) Abhängigkeitskonflikte involviert sind, müssen hier nicht angegeben werden (vgl. Abschnitt 6.2.5). Somit müssen im Element *Dataclasses* nur Variablen angegeben werden, für die ein spezielles Synchronisationsverhaltens vorgesehen ist. Die Datenklassen werden zudem erst bei einer Parallelausführung und dabei erst bei der Notwendigkeit zur Synchronisation benötigt. Bei einer sequentiellen Ausführung ist eine Verarbeitung des Elements *Dataclasses* somit gar nicht erforderlich. Das gleiche gilt für das Element *Synchronisation*. Es enthält die während der Parallelausführung zum Prozessmodell hinzugefügten Präzedenzen (*Precedences*), um die während der Ausführung festgelegte Reihenfolge der Variablenzugriffe festzuhalten.

Das optionale Element *AssignmentStrategy* bildet die Zuordnung von Elementen des Prozessmodells auf Ausführungseinheiten ab (vgl. Abschnitt 6.2.2). Es kann demnach eine globale Zuordnungsstrategie auf der Ebene der Prozessinstanz (*ProcessLevelAssignment*) und/oder Zuordnungsstrategien für einzelne Elemente des Prozessmodells besitzen, die auch unter *ActivityStates* aufgeführt sind und durch eine eindeutige *ID* gekennzeichnet werden können (*ActivityLevelAssignment*). Zuordnungsstrategien auf Aktivitätsebene überschreiben Zuordnungsstrategien auf Prozessebene. Jede Art der Zuordnung besitzt genau ein *Assignment*-Element, welches sich durch einen Identifikator (*StrategyID*) und ein generisches Konstrukt zur Implementierung der Strategie (*StrategyImpl*) auszeichnet. Der Identifikator der Zuordnungsstrategie kennzeichnet dabei das Softwaremodul des Migrationsdienstes, welches später für die Auswertung des Ausdrucks innerhalb von *StrategyImpl* verantwortlich ist. Damit ist die Art und Anzahl der Zuordnungsstrategien frei erweiterbar. Nach den Vorschlägen für die Strukturierung benutzerdefinierter

Vorgaben in Abschnitt 6.2.2 gilt für die prototypische Realisierung folgende Konvention:

- Besitzt das Element *StrategyID* den Wert „org.vsis.pmaas.FixedParticipant“, so enthält das Element *StrategyImpl* den eindeutigen Identifikator der zugeordneten Ausführungseinheit als UUID.
- Besitzt das Element *StrategyID* den Wert „org.vsis.pmaas.Role“, so enthält das Element *StrategyImpl* eine nach dem verwendeten Organisationsmodell gültigen Rollennamen.
- Besitzt das Element *StrategyID* den Wert „org.vsis.pmaas.Variable“, so enthält das Element *StrategyImpl* den Namen der Prozessvariable, aus welcher der eindeutige Identifikator der zuzuordnenden Ausführungseinheit als UUID ausgelesen werden kann.
- Besitzt das Element *StrategyID* den Wert „org.vsis.pmaas.PerformerOfActivity“, so enthält das Element *StrategyImpl* die ID einer Aktivität und das Protokoll (*Log*) besitzt zu dieser Aktivität einen Eintrag, dem der eindeutige Identifikator der zuzuordnenden Ausführungseinheit als UUID entnommen werden kann.
- Besitzt das Element *StrategyID* den Wert „org.vsis.pmaas.RessourceUnavailable“, so wird für den Fall, dass die für die Ausführung dieser Aktivität notwendige Ressource lokal nicht zur Verfügung steht, der Prozess an eine andere Ausführungseinheit migriert. Optional enthält das Element *StrategyImpl* dazu eine Liste mit eindeutigen Identifikatoren von alternativ zuzuordnenden Ausführungseinheiten als UUIDs.
- Besitzt das Element *StrategyID* den Wert „org.vsis.pmaas.ServiceSelection“, so enthält das Element *StrategyImpl* einen Ausdruck, welcher durch eine Komponente zur dynamischen Dienstauswahl verarbeitet werden kann. Die Spezifikation der dazugehörigen Beschreibungssprache und die Implementierung dieser Komponente kann einer weiterführenden Arbeit [HSZ11] entnommen werden.
- Besitzt das Element *StrategyID* einen anderen als den oben angegeben Wert, so wird geprüft, ob ein entsprechendes Softwaremodul für die individuelle Auswertung des in *StrategyImpl* angegebenen Ausdrucks existiert. Ist dies nicht der Fall, muss der Prozess an eine andere Ausführungseinheit transferiert werden, welche den Ausdruck auswerten kann.
- Existiert keine Zuordnungsstrategie auf der Ebene der Prozessinstanz (*ProcessLevelAssignment*) und taucht ein Element in der Liste der Zu-

ordnungsstrategien für einzelne Elemente des Prozessmodells (*Activity-LevelAssignment*) nicht auf, so kann das Element frei zugeordnet werden.

Das optionale Element *Privacy* enthält Inhalte, welche von der Komponente des *Privacy Managers* verarbeitet werden. Es werden dabei Inhalte zur Verschlüsselung des Prozesses zur Sicherstellung der Vertraulichkeit (*Encryption*) und Inhalte zur Gewährleistung der Integrität der Prozessbeschreibung bzw. der Migrationsdaten sowie zum Nachweis deren Ursprungs durch die Integration digitaler Signaturen unterschieden (*Signature*). Ein möglicher Aufbau und die Verarbeitung der sicherheitsbezogenen Metadaten werden in Abschnitt 7.5.2 erläutert.

Zum Sammeln von allgemeinen Informationen über die Prozessausführung, zur Nachvollziehbarkeit der vorgenommenen Anpassungen und zur Behandlung von speziellen Kontrollflusskonstrukten, welche nicht durch das allgemeine Migrationsmodell erfasst werden können, kann optional ein Protokoll geführt werden, welches an die Migrationsdaten angehängt werden kann (*Log*). Im Protokoll können dabei zu verschiedenen (tlw. anwendungsspezifischen) Kategorien Einträge hinterlassen und später ausgewertet werden. Es wird daher im Folgenden an den entsprechenden Stellen darauf Bezug genommen.

7.5.2 Umsetzung grundlegender Sicherheitsmechanismen

In Abschnitt 6.2.7 wurden verschiedene Sicherheitsmechanismen diskutiert, welche in Bezug auf die dynamische Verteilung eines Prozesses durch Migration zum Einsatz kommen können. Zum Schutz des Prozesses wurde für den hier beschriebenen Migrationsdienst exemplarisch das dort vorgeschlagene Sicherheitskonzept durch die (teilweise) Verschlüsselung des migrierten Prozesses mittels bestehender Verschlüsselungsmechanismen umgesetzt. Da die technische Repräsentation der Migrationsdaten sowie die im Kontext dieser Arbeit unterstützten Prozessbeschreibungssprachen auf XML basieren (vgl. Abschnitte 5.2.4 und 8.3) können die durch die W3C definierten Verfahren *XML Signature* [BBF+08] und *XML Encryption* [IDS+12] angewendet werden [ZHKK10]. Es wird im Folgenden das Vorliegen einer bestehenden *Public Key Infrastructure* (*PKI*) zwischen den beteiligten Ausführungseinheiten angenommen. Eine mögliche Implementierung zum Schutz der Ausführungseinheiten vor nicht autorisierten Prozessen kann der Arbeit von KOTTKE [Kot10] entnommen werden.

XML Signature bildet digitale Signaturen in einer XML-Repräsentation ab. Die signierten Daten können dabei in die Signatur eingebettet oder referenziert werden. Die Signaturen werden schließlich hierarchisch blockweise in die Migrationsdaten eingebunden (vgl. Listing 7.6). Dies erlaubt die (wahlweise) Verarbeitung einer signierten Prozessbeschreibung auch für Ausführungseinheiten, welche nicht über die optionale Komponente des *Privacy Managers* verfügen. Im Kontext dieser Arbeit wird folgende Vorgehensweise für das Signieren von migrierten Prozessen vorgeschlagen:

- Die Prozessbeschreibung (d. h. das Prozessmodell) wird im Laufe der verteilten Bearbeitung in der Regel nicht geändert. Es wird daher eine Prüfsumme über das XML-Dokument der Prozessbeschreibung gebildet und diese in die Signatur des Initiators der verteilten Prozessausführung eingebunden.

- Zustände und Variablenwerte werden vor einer Migration durch die ändernde Ausführungseinheit signiert. Eine Kopie der Änderungen wird mit einem Zeitstempel als *Append-only*-Log in einen speziellen Bereich des Protokolls (*Log*) eingefügt. Somit bleibt eine genaue Historie der Verantwortlichkeiten für die Prozessausführung zurück, welche von allen nachfolgenden Teilnehmern der verteilten Prozessausführung eingesehen und nachvollzogen werden kann.

- Benutzerdefinierte Vorgaben werden durch den jeweiligen Ersteller der Vorgabe signiert. Eine Kopie der Vorgabe wird mit einem Zeitstempel als *Append-only*-Log in einen speziellen Bereich des Protokolls (*Log*) eingefügt. Somit bleibt bei einer Anpassung eine genaue Historie der Verantwortlichkeiten für die Änderungen an den Vorgaben zurück, welche von allen nachfolgenden Teilnehmern der verteilten Prozessausführung eingesehen und nachvollzogen werden kann.

Die Verschlüsselung zur Gewährleistung der Vertraulichkeit von Prozessinformationen wird über *XML Encryption* realisiert. *XML Encryption* unterstützt sowohl die Verschlüsselung ganzer XML-Dokumente als auch die Verschlüsselung einzelner Elemente innerhalb eines XML-Dokuments [IDS+12] und ist deshalb gut für die Verschlüsselung von sicherheitsrelevanten Teilbereichen von Prozessbeschreibungen und von Migrationsdaten geeignet. Das Ergebnis des Verschlüsselungsprozesses wird wiederum im XML-Format repräsentiert. Zudem ist die Einbettung oder die Referenz verwendeter Verschlüsselungsalgorithmen und verschlüsselter Schlüssel für die spätere Entschlüsselung vorgesehen [IDS+12]. Eine Verarbeitung auf diese Weise verschlüsselter Prozesse kann durch eine entsprechende Erweiterung des XML-Parsers für die Interpretation der Prozessbeschreibung bzw. für die Migrationsdaten angestoßen werden.

Ein Problem bei der Verarbeitung teilweise verschlüsselter Prozessbeschreibungen tritt auf, wenn mehrere Elemente des XML-Dokuments individuell für unterschiedliche Empfänger verschlüsselt wurden. Als Konsequenz kann nicht die gesamte Prozessbeschreibung entschlüsselt werden, was zu einem unvollständigen Prozessmodell führt. Wird beim Deployment des Prozessmodells die gesamte XML-Prozessbeschreibung interpretiert (und z. B. in eine interne Objektrepräsentation übersetzt), so kann es dadurch zu Fehlern kommen, da das Ergebnis kein gültiges Prozessmodell darstellt. Es ist daher notwendig, die nach der Verarbeitung durch den *Privacy Manager* immer noch verschlüsselten (für die Ausführung der aktuell zugewiesenen Prozesspartitionen nicht benötigten) Elemente temporär durch Platzhalter-Elemente

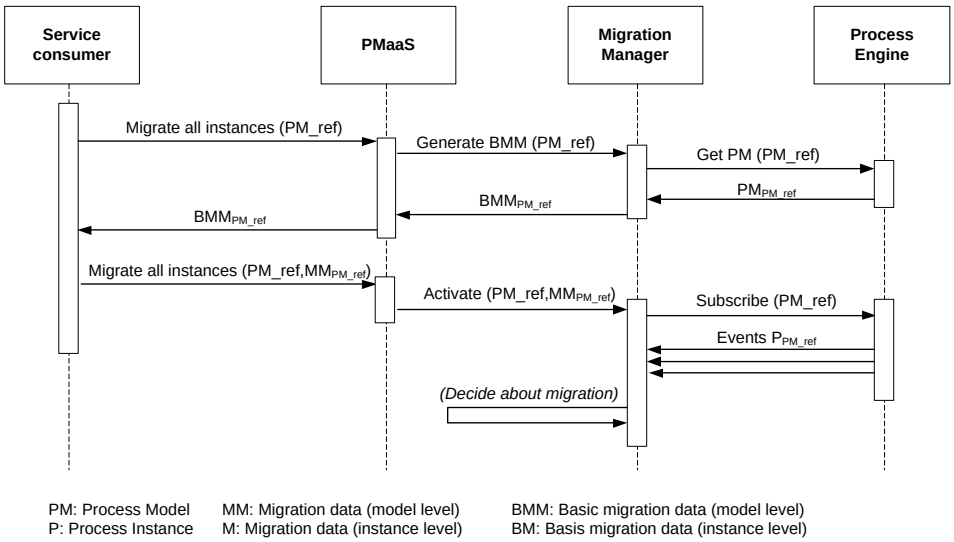
zu ersetzen [ZHKK10]. Da die Platzhalter-Elemente immer einer anderweitig zugewiesenen Prozesspartition entsprechen, kommen diese lokal nicht zur Ausführung. Das Ergebnis der Prozessausführung wird durch diese also nicht beeinträchtigt. Soll der lokal ausgeführte Prozess jedoch zu einem späteren Zeitpunkt an eine weitere Ausführungseinheit übertragen werden, so muss das Prozessmodell wieder durch die ursprüngliche Prozessbeschreibung ersetzt werden. Dazu ist es notwendig, dass der Migrationsmanager die (verschlüsselte) Prozessbeschreibung im Original speichert und bei einer Migration einer davon abgeleiteten Prozessinstanz das entsprechende Originaldokument versendet. Diese Aufgabe wird im Folgenden bei der genaueren Betrachtung des Migrationsmanagers wieder aufgegriffen.

7.5.3 Migrationsmanager

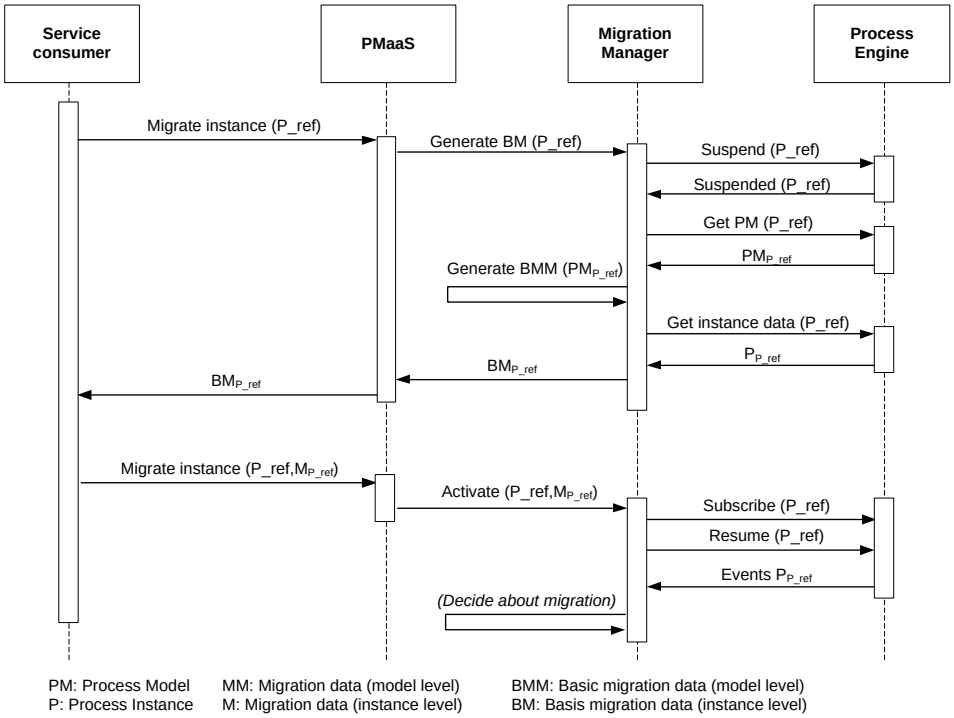
Der Migrationsmanager (*Migration Manager*, vgl. Abbildung 7.13) ist für die Interpretation und Verwaltung der Eingabedokumente des Migrationsdienstes und für die Koordination aller mit der Migration von Prozessinstanzen zusammenhängenden Vorgänge verantwortlich [ZHKK10]. Nach der in Abschnitt 6.2.1 vorgeschlagenen Methodik existieren insbesondere zwei zu unterstützende Teilszenarien: Erstens die spontane Migration eines ursprünglich zentral ausgeführten Prozesses und zweitens die Aus- bzw. Fortführung von Prozessen, welche bereits über Verteilungsinformationen in Form von Migrationsdaten verfügen.

Initiale Erzeugung von Migrationsdaten

Für das erste Teilszenario können Migrationsdaten für Prozessmodelle und/oder für (einzelne) zentral ausgeführte Prozessinstanzen abgeleitet werden. Der einfachste Fall liegt vor, wenn ein Prozessmodell bereits auf der lokal betrachteten Prozess-Engine installiert wurde und alle zukünftig davon abgeleiteten Prozessinstanzen denselben Rahmenbedingungen für eine potentielle Verteilung unterliegen sollen. In diesem Fall kann das Prozessmodell über den Migrationsdienst referenziert werden und bei Bedarf die Erzeugung eines grundlegenden Migrationsmodells (*Basismigrationsmodell*) angestossen werden, welches dem aufrufenden Benutzer zur (optionalen) weiteren Verfeinerung zurückgegeben wird (vgl. Operation *Migrate all instances* in Tabelle 7.13). Nach einer etwaigen (manuellen oder automatischen) Überarbeitung wird das Migrationsmodell (wieder) an den Migrationsdienst übergeben und es werden in Folge alle zukünftig von dem angegebenen Prozessmodell abgeleiteten Prozessinstanzen nach den darin spezifizierten Rahmenbedingungen ausgeführt. Dieselbe Vorgehensweise wird auch für eine (einzelne) zentral ausgeführte Prozessinstanz angewendet (vgl. Operation *Migrate instance* in Tabelle 7.13) – mit dem Unterschied, dass die betreffende Prozessinstanz erst einmal kontrolliert angehalten werden muss, bevor die Migrationsdaten initial angefertigt und ggf. durch benutzerdefinierte Rahmenbedingungen ergänzt werden



(a) Initiale Verteilung auf der Ebene aller von einem Prozessmodell zukünftig abgeleiteten Prozessinstanzen



(b) Initiale Verteilung auf der Ebene einer einzelnen Prozessinstanz

Abbildung 7.14: Migration ursprünglich zentral ausgeführter Prozesse (vereinfacht)

können. Die beiden Vorgehensweisen zur Erzeugung der Migrationsdaten sind zum Vergleich in Abbildung 7.14 schematisch dargestellt.

Der Dienstkonsument (*Service Consumer*) des PMaaS-Dienstes kann dabei entweder ein Prozessmodellierer, ein Prozessinitiator oder ein ausführender Teilnehmer des Prozesses sein, wobei die Interaktion mit dem Prozessmanagementsystem natürlich auch durch eine Softwareanwendung vorgenommen werden kann, welche den Benutzer durch eine angemessene (graphische) Benutzungsschnittstelle unterstützt und dann (stellvertretend) die angebotene PMaaS-Schnittstelle mit den Eingaben des Benutzers aufruft. Wie Abbildung 7.14 weiter zu entnehmen ist, wird auf Basis des Prozessmodells in beiden Fällen zunächst ein Basismigrationsmodell (BMM) erzeugt. Die Erzeugung des Basismigrationsmodells auf Prozessmodellebene erfolgt in Abhängigkeit der konkret verwendeten Prozessbeschreibungssprache und liefert ein Migrationsmodell zurück, welches prinzipiell alle für die Migration notwendigen Daten ohne die konkreten Instanzdaten und ohne die individuellen Rahmenbedingungen des Benutzers beinhaltet. Es stellt somit eine Art „Minimalvorschlag“ für die Verteilung der von dem betreffenden Prozessmodell abgeleiteten Prozessinstanzen dar. Listing 7.7 zeigt einen abstrakten Algorithmus für die Erzeugung eines Basismigrationsmodells für das Prozessmodell PM_m.

Bei der Erzeugung des Basismigrationsmodells werden zunächst alle relevanten Elemente der in Listing 7.6 dargestellten Struktur generiert und bei Vorliegen der entsprechenden Informationen mit Daten gefüllt. Insbesondere werden die einer Aktivität im Migrationsmetamodell entsprechenden Elemente der Prozessbeschreibung unter den Aktivitäten aufgeführt, damit diese später vom Benutzer mit einer Zuordnungstrategie verbunden werden können. Zudem wird der initiale Zustand einer von diesem Prozessmodell abgeleiteten Prozessinstanz erfasst, z. B. in Form der Initialwerte der Prozessvariablen. Den bei einer verteilt parallelen Ausführung in einen potentiellen Abhängigkeitskonflikt involvierten Variablen wird zudem die Datenklasse *Synchronized* als Kennzeichnung des restriktivsten Korrektheitskriteriums für eine spätere Synchronisierung zugeordnet. Durch die Auswahl dieser Variablen kann ein Benutzer (insbesondere der Prozessmodellierer) später relativ leicht auswählen, ob das hierdurch vorgegebene Synchronisationsverhalten im Anwendungskontext angemessen ist. Schließlich werden auf Basis vorgegebener Einschränkungskriterien bereits Vorschläge für konkrete Zuordnungsstrategien gemacht. In der prototypischen Realisierung werden dabei (exemplarisch) Aktivitäten herausgesucht, welche eine asynchrone Kommunikation auf Anwendungsebene ausdrücken und welche eine Kompensation einer vorhergehend ausgeführten Aktivität darstellen (vgl. Abschnitt 6.2.4). Als Vorschlag wird dabei als Zuordnungsstrategien auf Aktivitätsebene ausgedrückt, dass die als nachfolgend identifizierte Aktivität A_n jeweils von der Ausführungseinheit der zuvor ausgeführten Aktivität A_p verwaltet werden muss. Zur Laufzeit des Prozesses kann auf diese Weise automatisch eine Migration herbeigeführt werden, um eine Einschränkung der Verteilung zu er-

```
1  create Prologue:
2    UUID ← undefined
3    PDL ← process description language of PM_m
4    if deploment descriptor required
5      Deployment-Descriptor ← platform-independent representation of deployment
           descriptor
6    end if
7
8  create State:
9    ProcessState ← undefined
10   create Startactivities:
11     for each activity-like element A_n of PM_m without incoming transitions
12       create Startactivity:
13         Startactivity ← A_n
14     end for each
15   create ActivityStates:
16     for each activity-like element A_n of PM_m
17       create Element:
18         ID ← ID of A_n
19         ActivityState ← undefined
20     end for each
21   create VariableValues
22       for each process variable V_j of PM_m
23         create Variable:
24           Name ← Name of V_j
25           Value ← Initial value of V_j
26       end for each
27
28 create Dataclasses:
29   detect dependency conflicts (design time)
30   for each process variable V_j of PM_m involved in a dependency conflict
31       create Variable:
32           Name ← Name of V_j
33           Dataclass ← synchronized
34   end for each
35
36 create AssignmentStrategy:
37   create ElementLevelAssignment:
38     for each activity-like element A_n of PM_m
39         for each activity-like element A_p of PM_m previous to A_n
40           if A_n is asynchronous response of A_p
41             create Element:
42               create Assignment:
43                 StrategyID ← org.vsis.pmaas.PerformerOfActivity
44                 StrategyImpl ← A_p
45           end if
46           if A_n is compensation activity of A_p
47             create Element:
48               create Assignment:
49                 StrategyID ← org.vsis.pmaas.PerformerOfActivity
50                 StrategyImpl ← A_p
51           end if
52       end for each
53     end for each
54
55 create Privacy:
56   create Signature
57     Processmodel ← signed checksum of PM_m
```

Listing 7.7: Abstrakter Algorithmus zur Erzeugung eines Basismigrationsmodells BMM_m auf Grundlage des Prozessmodells PM_m (Beispiel-Konfiguration)

zwingen. Als Beispiel für die Umsetzung von Sicherheitsmechanismen wird das vorliegende Prozessmodell digital signiert.

Listing 7.8 zeigt den (abstrakten) Algorithmus für die Erzeugung eines Basismigrationsmodells auf Prozessinstanzebene, also für den Fall, dass eine bestimmte Prozessinstanz $P_{m,i}$ verteilt ausgeführt werden soll. Das vorliegende Basismigrationsmodell auf Prozessmodellebene wird dazu „instantiiert", d. h. auf die ausgewählte Prozessinstanz und ihren aktuellen Zustand angepasst. Zunächst wird eine UUID erzeugt, um die zu migrierende Prozessinstanz auch im Kontext weiterer Prozess-Engines eindeutig zu kennzeichnen. Zudem wird eine Momentaufnahme (*Snapshot*) der Prozessausführung erzeugt und der aktuelle Zustand der bereits auf Prozessmodellebene identifizierten Elemente und Variablen gesetzt. Wie oben erläutert wurde, muss die Prozessinstanz zur erstmaligen Erzeugung von Migrationsdaten temporär angehalten werden. Sie befindet sich daher zum Zeitpunkt der Migrationsdatenerstellung immer im Zustand *suspended*. Die Aktivitätszustände und Variablenwerte können dabei durch die Angabe ihrer Identifikatoren abgefragt werden. Durch die geleistete (einmalige) Vorarbeit auf Prozessmodellebene kann damit die Zeit zur Erzeugung von Migrationsdaten zur Laufzeit des Prozesses verkürzt werden.

Soll eine bereits laufende Prozessinstanz für die Migration vorbereitet werden, so ist in der Regel bereits ein Teil des Prozesses ausgeführt. Die Anzahl der in einen potentiellen Abhängigkeitskonflikt involvierten Variablen kann sich damit bereits reduziert haben. Entsprechend werden die nicht mehr zu berücksichtigenden Variablen aus der Definition der Datenklassen entfernt. Zudem werden die bereits im Rahmen einer etwaigen Parallelausführung hinzugefügten Präzedenzen aufgeführt (*Synchronisation*).

Der Benutzer kann auch in diesem Fall die vom Migrationsdienst vorgeschlagenen Migrationsdaten einsehen und bei Bedarf erweitern oder modifizieren. Nach der Übergabe des als tatsächlich anzuwendend deklarierten Migrationsmodells (welches auch aus dem „Minimalvorschlag" des Migrationsdienstes bestehen kann) wird dieses durch den Migrationsmanager gespeichert und auf die entsprechende(n) Prozessinstanz(en) angewendet. Es liegen nun alle Informationen vor, welche auch über die Operation *Execute instance* von anderen Ausführungseinheiten zur Nutzung der PMaaS-Funktionalität übergeben werden würden. Die weitergehende Vorgehensweise wird dazu integriert im nächsten Unterabschnitt erläutert.

Durchführung von Migrationen

Im zweiten Teilszenario und damit beim Hauptanwendungsfall der dynamisch verteilten Prozessausführung sind bereits Migrationsdaten vorhanden und werden gemeinsam mit dem Prozessmodell als Ausführungsanweisung für die Prozessinstanz an den Migrationsdienst übergeben (vgl. Operation *Execute instance* in Tabelle 7.13). Im Fall einer sequentiellen Ausführung ist der Aufwand für die Verteilung des Prozesses auf die Installation bzw. Deinstallation des Prozesses und ein relativ einfaches Protokoll zur Übergabe der Verantwortlich-

```
1  init Prologue:
2    UUID ← create UUID
3
4  init State:
5    ProcessState ← suspended
6    remove Startactivities
7    create Startactivities:
8      for each active pointer to activity-like element A_n in P_{m,i}
9        create Startactivity:
10         Startactivity ← A_n
11     end for each
12   init ActivityStates:
13     for each activity-like element A_n of PM_m
14       init Element:
15         ActivityState ← current state of A_i in P_{m,i}
16     end for each
17   init VariableValues
18       for each process variable V_j of PM_m
19         init Variable:
20           Value ← Current value of V_j in P_{m,i}
21       end for each
22
23 update Dataclasses:
24   detect dependency conflicts (runtime)
25   for each process variable V_j of Dataclasses in BMM_m
26     if not V_j of P_{m,i} actually involved in a dependency conflict
27       remove Dataclass entry of V_j
28     end if
29   end for each
30
31 create Synchronisation:
32   for each precedence S_j added to solve a dependency conflict
33     create Precedence:
34       Precedence ← S_j
35   end for each
```

Listing 7.8: Abstrakter Algorithmus zur Instantiierung eines Basismigrationsmodells $BM_{m,i}$ auf Grundlage der (laufenden) Prozessinstanz $P_{m,i}$ und des Migrationsmodells BMM_m

keit für dessen Ausführung beschränkt. Im Allgemeinen kann die Vorgehensweise der Migration einer laufenden Prozessinstanz (ungeachtet möglicher Fehlerfälle) in vier unterschiedliche Phasen eingeteilt werden [BDH+12]:

1. Auf dem Quellausführungssystem wird die Ausführung des zu migrierenden Prozesses in einem nach dem Zustandsmodell migrierbarer Prozesse definierten konsistenten Zustand angehalten. Dazu werden keine neuen Aktivitäten mehr gestartet und es wird gewartet, bis die zur Zeit der Migrationsentscheidung auf der lokalen Prozess-Engine ausgeführten atomaren Aktivitäten beendet sind.

2. Die nach dem Migrationsmodell relevanten Instanzdaten des zu migrierenden Prozesses werden aus der Quellausführungsumgebung extrahiert und in das Migrationsdatendokument übernommen.

3. Das Quellausführungssystem übergibt das Prozessmodell und das Migrationsdatendokument dem Migrationsdienst der Zielausführungsumgebung als Eingabeparameter (Operation *Execute instance*). Nach einer Prüfung des Prozessmodells auf seine prinzipielle

Ausführbarkeit auf der lokalen Prozess-Engine und der syntaktischen Korrektheit der Migrationsdaten installiert das Zielausführungssystem das Prozessmodell des migrierten Prozesses, stellt den durch die Migrationsdaten definierten Ausführungszustand wieder her und abonniert die für die Prozessausführung definierten fachlichen Ereignisse.

4. Nach Erhalt einer Antwortnachricht über die erfolgreiche Wiederaufnahme der Ausführung wird das Prozessmodell auf dem Quellausführungssystem deinstalliert, die Registrierung für fachliche Ereignisse dieses Prozesses aufgehoben und die angehaltene Prozessinstanz gelöscht. Bei Bedarf können die Instanzdaten der migrierten Prozessinstanz als Prozesshistorie auf dem Quellausführungssystem erhalten bleiben. Das Zielausführungssystem nimmt die Ausführung der Prozessinstanz an der durch die Migrationsdaten definierten Position im Kontrollfluss wieder auf.

Für die Umsetzung dieser Funktionalität verwaltet der Migrationsmanager die Migrationsmodelle, welche die auf der lokalen Prozess-Engine ablaufenden Prozesse betreffen. Abbildung 7.15 zeigt einen Überblick über die vorgeschlagene Architektur des Migrationsmanagers. Nach Aufruf der Operation *Execute instance* werden das Prozessmodell und die Migrationsdaten zunächst zwischengespeichert (*PMaaS Documents*). Da es den Migrationsvorgang vereinfacht und da für die Teilverschlüsselung von Prozessen das zu installierende Prozessmodell für die Zeitdauer der lokalen Ausführung verändert werden muss (vgl. Abschnitt 7.5.2), erhält der Migrationsmanager das Prozessmodell jeweils im Original, d. h. exakt so wie es an der PMaaS-Schnittstelle übergeben wurde. Soll später eine Migration der dazugehörigen Prozessinstanz veranlasst werden, so kann daher das ursprünglich empfangene Prozessmodell weitergeleitet werden.

Die in den Migrationsdaten enthaltenen Rahmenbedingungen (*Assignment Strategies*) werden an die entsprechend mit den angegebenen Strategien verknüpfte Softwaremodule weitergeleitet (*Migration Decision Modules*). Diese abonnieren die für die Durchführung der Migrationsentscheidung relevanten Ereignisse der lokalen Prozess-Engine (vgl. Abschnitt 7.5.4) bzw. beziehen die hierfür benötigten Informationen aus anderen externen Quellen. Dasselbe gilt für die Softwaremodule zur Überwachung etwaiger lokaler Richtlinien der Prozess-Engine (*Local Policies*). Die prototypische Realisierung setzt exemplarisch eine einfache lokale Richtlinie um, welche den Prozess migriert, sobald eine im Prozess definierte Ressource für die Ausführung einer Aktivität nicht zugegriffen werden kann (*Resource Unavailable Impl.*). Die Möglichkeit zum Rebinding von Ressourcen ist von der Prozessbeschreibungssprache, der verwendeten Prozess-Engine und dem Vorliegen eines Organisationsmodells für Ressourcen (z. B. Rollenmodell) abhängig. Dieser Aspekt wird daher in Abschnitt 8.2 bei der Betrachtung konkreter Prozessmanagementsysteme wieder aufgegriffen.

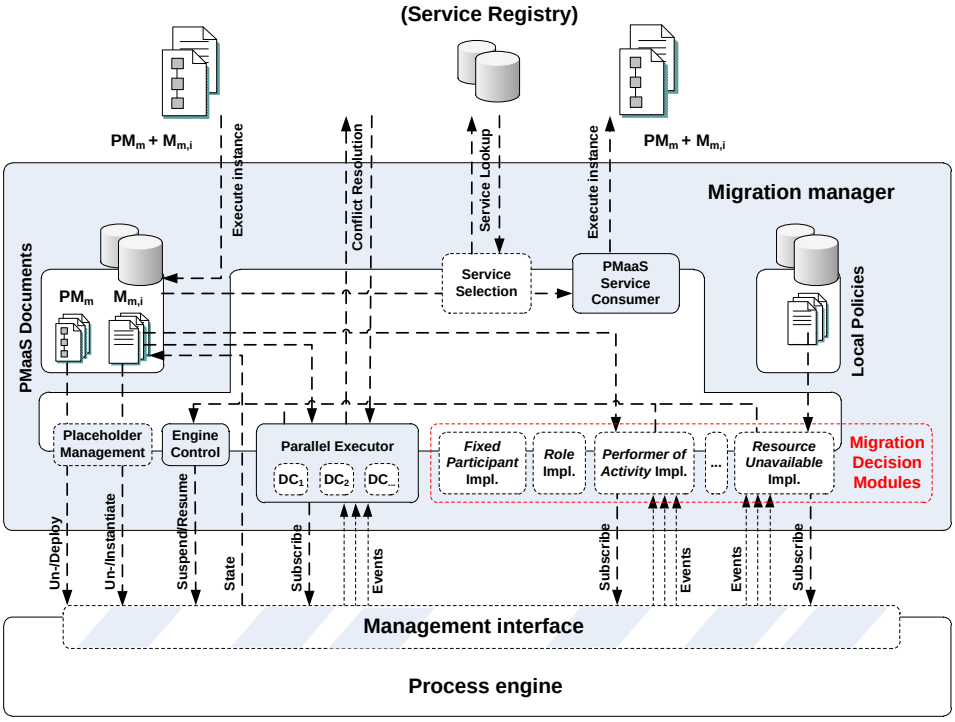


Abbildung 7.15: Architektur des Migrationsmanagers (Überblick)

Werden nach dem Deployment, der Wiederherstellung des Ausführungszustandes aus den Migrationsdaten und der Wiederaufnahme der Prozessausführung Ereignisse empfangen, welche bei einem der überwachenden Softwaremodule zu einer Migrationsentscheidung führen, so kann die Prozessausführung kontrolliert angehalten (*Engine Control*), der aktuelle Ausführungszustand extrahiert und die Migrationsdaten aufbereitet werden. Konnte durch das über die Migration entscheidende Softwaremodul die nachfolgende Ausführungseinheit identifiziert werden, so wird diese im Prolog der Migrationsdaten vermerkt (vgl. Abschnitt 7.5.1). Ist die nachfolgende Ausführungseinheit undefiniert oder muss diese anhand einer Reihe von spezifizierten Eigenschaften ausgewählt werden, können Maßnahmen zur Dienstauswahl (*Service Selection*) integriert werden, um eine konkrete Ausführungseinheit aus einem (beliebigen) Dienstverzeichnis (*Service Registry*) auszuwählen. Der Abschluss des Vorgangs wird durch den Aufruf der Operation *Execute instance* bei der zur Fortführung des Prozesses ausgewählten Ausführungseinheit eingeleitet. Nach Empfang der entsprechenden Antwortnachricht über die erfolgreiche Übernahme des Prozesses kann dieser auf der lokalen Prozess-Engine deinstalliert und das Dokumenten-Paar aus Prozessmodell und Migrationsdaten aus dem Speicher des Migrationsmanagers entfernt werden.

Verteilte Ausführung paralleler Prozesspartitionen

Soll ein Prozess für eine verteilte Ausführung einzelner paralleler Prozesspfade repliziert werden, so wird die Operation *Execute instance* bei den für die Ausführung ausgewählten Ausführungseinheiten (mehrfach) mit demselben Prozessmodell und den individualisierten Migrationsdaten mit unterschiedlichen Startaktivitäten aufgerufen. Dabei wird als Identifikator im Prolog der Migrationsdaten jeweils die UUID des Ursprungsprozesses verwendet. Auf diese Weise ist später zur Synchronisation eine eindeutige Zuordnung der parallel ausgeführten Prozesspartitionen und deren Zusammenführung auf eine vereinigte Prozessinstanz möglich. Für die Vereinigung wird wiederum die Operation *Execute instance* bei der für die Zusammenführung ausgewählten Ausführungseinheit aufgerufen. Die Vereinigung wird ausgelöst, wenn bei einem Migrationsmanager mindestens zwei migrierte Prozessinstanzen mit derselben UUID vorliegen. Die Synchronisation erfolgt dabei wie in Abschnitt 6.2.5 beschrieben, wobei für eine verteilte Koordination die Auswahl einer Ausführungseinheit bzw. ein Wartezustand zur Zusammenführung der Replikate durch den Aufruf der *Synchronize*-Operation angezeigt werden kann (vgl. Tabelle 7.13).

Während der parallelen Ausführung kann es in Abhängigkeit der im Migrationsmodell definierten Datenklassen zu der Notwendigkeit einer Abstimmung zwischen den Ausführungseinheiten paralleler Prozesspartitionen kommen. Dazu müssen die im Migrationsmodell spezifizierten Datenklassen ausgewertet werden und die Einhaltung des dort geforderten Synchronisationsverhaltens während der Ausführung überwacht werden. Wie Abbildung 7.15 zu entnehmen ist, existiert hierzu das Modul *Parallel Executor*, welches eine ähnliche Funktion besitzt wie die Module zur Durchführung von Migrationsentscheidungen (*Migration Decision Modules*). Es bekommt jedoch keine benutzerdefinierten Zuordnungsstrategien aus den Migrationsdaten geliefert, sondern nimmt die Identifikatoren der Prozessvariablen entgegen, für die in den Migrationsdaten Datenklassen spezifiziert wurden. Der Überwachungsmechanismus wird aktiviert, sobald festgestellt wird, dass parallele Pfade verteilt ausgeführt werden. Dieser Umstand ist aus dem Prozessmodell und aus den der lokalen Prozess-Engine durch die Definition der Startaktivitäten zugewiesenen Pfaden zu erkennen. In Abhängigkeit der Datenklasse werden in Folge die zur Überwachung erforderlichen Ereignisse bei der Prozess-Engine abonniert und bei Eintritt eines relevanten Ereignisses das im Rahmen der Datenklasse zur Synchronisation spezifizierte Verhalten ausgelöst.

In der prototypischen Realisierung wurde dabei nur die Implementierung der Datenklasse *Synchronized* umgesetzt. Für die Behandlung von Abhängigkeitskonflikten von auf verschiedenen parallel auszuführenden Prozesspartitionen verwendeten Variablen wird dabei das in Abschnitt 6.2.5 vorgeschlagene Verfahren zur optimistischen Konfliktauflösung umgesetzt. Hierfür ist es erforderlich, dass Koordinationsnachrichten mit den

Ausführungseinheiten anderer parallel auszuführenden Prozesspartitionen ausgetauscht werden können (vgl. auch [Ham09]). Die Operation *Find instance* (vgl. Tabelle 7.13) dient dazu dem Auffinden einer (migrierten) Prozessinstanz unter Angabe der Aktivitäten, deren Ausführung für die Koordination relevant ist. Die Operation kann dabei entweder über einen Broadcast bei allen potentiell beteiligten Ausführungseinheiten aufgerufen werden oder die Anfrage wird gezielt an Ausführungseinheiten weitergeleitet, welche den Prozess bearbeitet und/oder weitergeleitet haben. Das von HAMANN [Ham09] vorgeschlagene Protokoll zur optimistischen Konfliktauflösung kann dabei über die Verwendung der drei Operationen *Request conflict resolution*, *Apply conflict resolution* und *Cancel conflict resolution* umgesetzt werden, deren Eingabe- und Rückgabewerte vom Modul *Parallel Executor* verarbeitet werden. Nähere Informationen zur Umsetzung des *Parallel Executors* können der Arbeit von [Ham09] entnommen werden.

7.5.4 Anbindung an das Prozessmanagementsystem

Während der Ausführung der Prozessinstanzen muss der Migrationsmanager die potentiell zu migrierenden Prozesse überwachen und steuern. Der Migrationsmanager bedient sich dazu den unter Abschnitt 7.3.2 erläuterten grundlegenden Managementfunktionalitäten des Prozessmanagementsystems als verwaltbare Ressource. Die wesentlichen Interaktionen zwischen Migrationsmanager und Prozess-Engine sind bereits in Abbildung 7.15 dargestellt. In diesem Abschnitt wird eine mögliche Abbildung der für die Prozessmigration benötigten Managementfunktionalitäten auf die durch die verwaltbare Ressource bereitgestellten Operationen vorgestellt. Dabei wird zwischen Funktionalitäten zur Umsetzung der *Migration* und Funktionalitäten zur Umsetzung von *Migrationsentscheidungen* unterschieden.

Funktionalitäten zur Umsetzung der Migration umfassen die Installation des Prozessmodells, die Bildung einer neuen Prozessinstanz, das Aktualisierung des Zustandes, die Wiederaufnahme der Ausführung sowie die entsprechenden Maßnahmen zum Anhalten der Ausführung, zur Extraktion des Zustandes, zur Entfernung der Prozessinstanz und zur Deinstallation des Prozessmodells:

- Die Installation bzw. Deinstallation von Prozessmodellen gehört zur Standardfunktionalität von Prozessmanagementsystemen und kann damit direkt durch Aufruf der entsprechenden Operationen *Deploy process* und *Undeploy process* umgesetzt werden (vgl. Tabelle 7.10).

- Die Bildung einer neuen Prozessinstanz birgt hingegen größere Herausforderungen. Da die ursprünglichen Aufrufparameter des Prozesses nach einer Migration nicht bekannt sind, muss quasi eine „leere“ Prozessinstanz erzeugt werden, welche später um die Zustände der Aktivitäten und Variablen erweitert werden kann. Insbesondere darf die neu erzeugte Prozessinstanz nicht direkt nach ihrer Erzeugung ausgeführt wer-

den. Die Operation *Instantiate suspended process* erlaubt dazu, eine Prozessinstanz im Zustand *suspended* zu erzeugen (vgl. Tabelle 7.10).

- Mit Hilfe der Operation *SetResourceProperties* von *WS-Resource-Properties* (vgl. Tabelle 7.2) können die Eigenschaften der Subressource *Prozessinstanz* (vgl. Tabelle 7.5) und ihrer entsprechenden Subressourcen auf den in den Migrationsdaten angegebenen Stand gebracht werden. Insbesondere werden die Werte der Variablen (Eigenschaft *Value*, vgl. Tabelle 7.8) und der Aktivitäten (Eigenschaft *State*, vgl. Tabelle 7.7) gesetzt.

- Die fertige Prozessinstanz kann über die Operationen *Resume* und *Suspend* gestartet bzw. angehalten werden (vgl. Tabelle 7.10).

- Mit Hilfe der Operation *GetResourceProperty* von *WS-Resource-Properties* (vgl. Tabelle 7.2) können die Eigenschaften der Subressource *Prozessinstanz* (vgl. Tabelle 7.5) und ihrer entsprechenden Subressourcen ausgelesen und in die Migrationsdaten eingetragen werden. Insbesondere werden die Werte der Variablen (Eigenschaft *Value*, vgl. Tabelle 7.8) und der Aktivitäten (Eigenschaft *State*, vgl. Tabelle 7.7) erfasst.

- Sofern die Prozessinstanz nicht mehr benötigt wird, wird sie mittels der Operation *Cancel* abgebrochen bzw. gelöscht (vgl. Tabelle 7.10).

Für die Durchführung von Migrationsentscheidungen müssen in Abhängigkeit der implementierten Zuordnungsstrategien bzw. lokalen Richtlinien unterschiedliche Funktionalitäten unterstützt werden:

- Für die Anwendung der exemplarisch implementierten Zuordnungsstrategien (*FixedParticipant*, *Role*, *Variable* und *PerformerOfActivity*) auf Aktivitätsebene ist es ausreichend, für die angegebene Aktivität das Ereignis *ActivityStateChanged* mit blockierender Wirkung für das Eintreten des Zustands *ready* zu abonnieren (vgl. Abschnitt 6.2.2 und Tabelle 7.11). In diesem Fall wird die Ausführung durch das blockierende Ereignis angehalten und der Prozess kann wie oben beschrieben migriert werden. Ein Abonnement weiterer (blockierender) Ereignisse ist nicht notwendig. Die Prozessausführung wird hierbei also außerhalb der gezielten Migrationsentscheidung nicht beeinträchtigt.

- Für die Anwendung der exemplarisch implementierten Zuordnungsstrategien (*FixedParticipant*, *Role*, *Variable* und *PerformerOfActivity*) auf Prozessebene kann bereits vor dem Beginn der Ausführung geprüft werden, ob die Bedingungen für die lokale Ausführung erfüllt sind. In den oben genannten Fällen ist dafür ein Abgleich der geforderten Eigenschaften mit der lokalen Identität nötig. Ein einmaliger Abruf der Identitätsinformationen der Prozess-Engine ist hierfür ausreichend. Die Prozessausführung wird hierbei also nicht beeinträchtigt.

- Für die Anwendung der exemplarisch implementierten lokalen Richtlinie zur Überwachung nicht verfügbarer Ressourcen (*Resource Unavailable*) kann für alle Aktivitäten das Ereignis *On Error* mit blockierender Wirkung für das Auftreten der Fehlermeldung *Resource unavailable* abonniert werden (vgl. Tabelle 7.11). In diesem Fall wird die Ausführung der Prozessinstanz durch das blockierende Ereignis angehalten und der Prozess kann wie oben beschrieben migriert werden. Ein Abonnement weiterer (blockierender) Ereignisse ist hierfür nicht notwendig. Die Prozessausführung wird also außerhalb der geplanten Migration nicht beeinträchtigt.
- Für die Anwendung einer möglichen lokalen Richtlinie zur Migration des Prozesses für den Fall, dass die Auslastung der Prozess-Engine zu hoch wird, kann die Änderung der Eigenschaft der Prozess-Engine *Workload* über einen bestimmten Schwellwert abonniert werden (vgl. Tabelle 7.3). Bei Eintreten des Ereignisses wird die Operation *Suspend* aufgerufen und die Prozessinstanz kontrolliert angehalten, so dass diese migriert werden kann (vgl. Tabelle 7.10). Die Prozessausführung wird hierbei außerhalb der geplanten Migration nicht beeinträchtigt.
- Es ist prinzipiell möglich, weitere benutzerdefinierte Zuordnungsstrategien zu formulieren, welche es erfordern, dass für alle Aktivitäten des Prozesses das Ereignis *ActivityStateChanged* mit blockierender Wirkung für das Eintreten eines bestimmten Zustands abonniert werden muss. Dies ist der Fall, wenn für jede lokal aktivierte Aktivität im Einzelnen überprüft werden muss, ob die Grundlage zur weitergehenden Ausführung bzw. zur Migration gegeben ist. Eine derartige Umsetzung mit Hilfe der Managementschnittstelle der verwaltbaren Ressource ist grundsätzlich möglich. Durch die Blockierung jeder Aktivität zu deren individueller Überprüfung kann es jedoch insgesamt zu einer Verzögerung der Prozessausführung kommen. Es ist daher anzustreben, benutzerdefinierte Zuordnungsstrategien in Hinblick auf die Verzögerung des Prozesses möglichst effizient umzusetzen und im Einzelfall zu prüfen, ob eine Blockierung der Prozessausführung für den Zeitraum der Auswertung einer Migrationsentscheidung notwendig ist.

Die dargestellte Vorgehensweise zeigt nur den prinzipiellen Einsatz der durch die verwaltbaren Ressource bereitgestellten Managementfunktionalitäten. In Abhängigkeit der verwendeten Prozessbeschreibungssprache und den individuellen Eigenschaften der Prozess-Engine muss diese entsprechend angepasst oder erweitert werden. Eine konkrete Anbindung an verschiedene Prozessmanagementsysteme wird in Abschnitt 8.2 diskutiert.

7.6 Interaktionsdienst

Werden Prozesse verteilt ausgeführt, so ist unter Umständen zur Entwicklungszeit nicht bekannt, auf welchem Endgerät der Benutzer eine Interak-

Operation	Beschreibung
Perform interaction	Übergibt eine nach Abschnitt 6.4.2 definierte abstrakte Interaktionsbeschreibung (AID) sowie die Werte der in der AID referenzierten Prozessvariablen, um eine kontextabhängige Benutzungsschnittstelle (CUI) zu erzeugen und dem Benutzer zur Anzeige zu bringen. Nach Abschluss der Interaktion werden die als Ausgabeparameter definierten Prozessvariablen zurückgegeben.

Tabelle 7.14: Schnittstelle des Interaktionsdienstes

tion mit dem Prozess ausführen wird (vgl. Abschnitt 6.4). Die Realisierung des *Interaktionsdienstes* übernimmt die plattform- und kontextspezifische Darstellung von Aktivitäten eines (verteilt) ausgeführten Prozesses, welche eine durch die Aktivitätsbeschreibung festgelegte Interaktion mit einem menschlichen Benutzer erfordern und nicht bereits durch den Aufruf einer auf die fachliche Aufgabe spezialisierte Anwendung abgedeckt werden. Im Gegensatz zu den bisher vorgestellten Middleware-Komponenten baut der Interaktionsdienst nicht auf der bereitgestellten Managementfunktionalität auf, sondern erweitert die Menge der (lokal verfügbaren) funktionalen Dienste um eine generische Schnittstelle zu den menschlichen Prozessteilnehmern der jeweiligen Ausführungsumgebung (vgl. Abschnitt 7.1). Dazu erhält der Interaktionsdienst als Eingabeparameter die abstrakte (d. h. plattform- und kontextunabhängige) Beschreibung der Benutzungsschnittstelle, welche einem zur Laufzeit ausgewählten menschlichen Prozessteilnehmer zur Erfüllung einer interaktiv oder manuell auszuführenden Aktivität des (verteilten) Prozesses in geeigneter Weise angezeigt werden soll. Abbildung 7.16 zeigt den schematischen Aufbau eines Interaktionsdienstes, welcher prinzipiell verschiedene Modalitäten für die Interaktion mit dem Benutzer besitzen kann.

Tabelle 7.14 zeigt die einfache Schnittstelle des Interaktionsdienstes mit nur einer Operationen zur Erzeugung der Benutzungsschnittstelle. Konform zu einem dem in Abschnitt 6.4.2 vorgestelltem Metamodell entsprechenden XML-Schema wird die Beschreibung abstrakter Benutzungsschnittstellen in XML-Repräsentation ausgedrückt und anstelle der plattformabhängigen Interaktionsbeschreibung mit der Prozessbeschreibung an die Ausführungsumgebung(en) übertragen (vgl. [Vil08] zu Details). Soll dem Benutzer während der Ausführung eine Aufgabe angezeigt werden, so wird die Interaktionsbeschreibung an den (lokalen) Interaktionsdienst übergeben (vgl. Abschnitt 6.4.4). Die XML-Beschreibung der Interaktionsaktivität wird zunächst durch einen XML-Parser in eine abstrakte Objektrepräsentation umgewandelt, welche über das Entwurfsmuster der *Fabrikmethode* [GHJJ10] auf Basis der zur Laufzeit anzuwendenden Modalität in eine konkrete Benutzungsschnittstelle umgewandelt wird (*Interaction Factory*). Dabei können die einzelnen Modalitäten jeweils durch die lokale Ausführungsumgebung konkretisiert oder durch einen Kontextdienst dynamisch verwaltet werden [ZBV09]. In der prototypischen Implementierung wurde die visuelle Interaktion mit dem Benutzer über eine Bildschirmausgabe in Verbindung mit Tastatureingaben bzw. Eingaben über einen Touch-Screen jeweils für einen Arbeitsplatzrechner

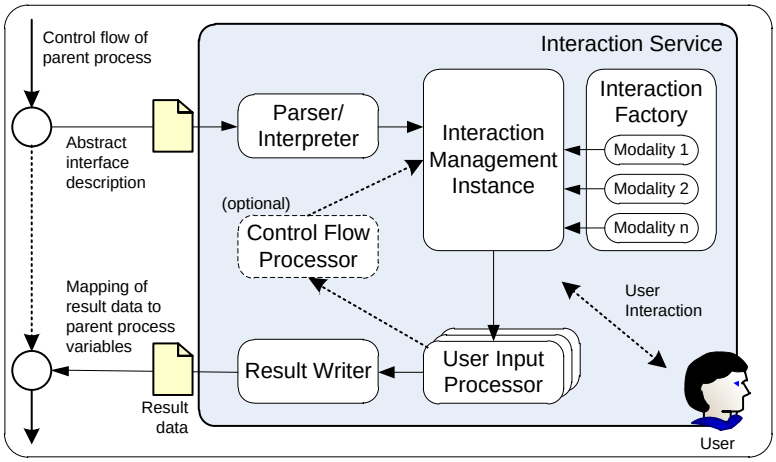


Abbildung 7.16: Komponenten des Interaktionsdienstes zur Repräsentation kontextbasierter Benutzungsschnittstellen (ZBV09))

und ein Mobiltelefon realisiert (vgl. [Vil08, ZBV09, ZVBK09]. Die konkreten Fabrikmethoden führen hierbei eine Zuweisung der in Abschnitt 6.4.2 definierten abstrakten Interaktionselemente auf die (auf dem zur Interaktion genutzten Endgerät mit der dort verfügbaren Software) in der konkreten Situation tatsächlich zur Verfügung stehenden Interaktionskomponenten durch. Dabei werden auch die Werte der aus der übergeordneten Prozessbeschreibung übernommenen Variablen sowie die Datentypen der zu erwartenden Eingaben des Benutzers eingearbeitet, so dass als Ergebnis dieses Arbeitsschritts eine vollständige und konkrete Benutzeroberfläche erzeugt wird.

Das Interaktionsmanagement (*Interaction Management Instance*) ist für die Ausgabe der auf diese Weise erzeugten Benutzungsschnittstelle und für die Entgegennahme etwaiger Benutzereingaben verantwortlich. Sollen diese Eingaben für direkt nachfolgende Dialoge mit dem Benutzer zur Verfügung stehen, so können diese (optional) durch eine Komponente zur Verwaltung des Kontrollflusses zwischen mehreren zusammenhängenden Benutzungsschnittstellen entgegengenommen werden (*Control Flow Processor*). Die Eingaben des Benutzers stellen zudem das Ergebnis der Interaktion und somit nach Abschluss der Interaktion die Rückgabe des Interaktionsdienstes dar. Ist die Interaktion abgeschlossen, werden die Eingaben des Benutzers daher wie in der Interaktionsbeschreibung vorgesehen auf die entsprechenden Variablen des Prozesses abgebildet. Für die Aufbereitung und Kapselung der Eingaben in einer Antwortnachricht ist ein XML-Writer (*Result Writer*) verantwortlich [ZBV09].

Eine individuelle Ausgestaltung des Interaktionsdienstes für einzelne Endgeräte ist möglich, jedoch nicht notwendig. So kann im Bereich der mobilen Endgeräte bereits durch die Spezialisierung des Interaktionsdienstes auf Geräteklassen wie zum Beispiel der Java-, iOS- oder Android-basierten Mobiltelefone ein Großteil aktuell verfügbarer mobiler Endgeräte mit ihren individuellen Ein- und Ausgabeschnittstellen und Interaktionselementen abgedeckt

werden. Im Bereich der Desktop- oder Notebook-Arbeitsplätze kann der Interaktionsdienst als eigenständige Anwendung mit graphischer Benutzeroberfläche oder durch eine HTML-basierte Browseranwendung abgebildet werden. Ein Beispiel für einen Interaktionsdienst zur Abbildung der abstrakten Interaktionselemente auf konkrete Interaktionselemente der *Java Micro Edition* (*Java ME*) für die Nutzung Java-basierter Mobiltelefone zur Interaktion mit verteilt ausgeführten Prozessen kann der Arbeit von VILENICA [Vil08] entnommen werden.

Durch die Möglichkeit zur individuellen Ausgestaltung der Fabrikmethoden und durch die Übergabe der abstrakten Beschreibung an den Interaktionsdienst wird eine späte Generierung von Benutzungsschnittstellen unter Berücksichtigung individueller Rahmenbedingungen ermöglicht. Die Komplexität der Darstellung kann damit sowohl individuell an die Leistungsfähigkeit der Ein- und Ausgabemedien als auch an das typische Erscheinungsbild der lokalen Ausführungsumgebung angepasst werden. Somit kann letztendlich auch die Ausführung verteilt ausgeführter Prozesse für die lokalen Prozessteilnehmer weitestgehend transparent gehalten werden und es besteht nicht die Notwendigkeit, von der gewohnten Art und Weise der Aufgabenbearbeitung abweichen zu müssen.

7.7 Integration von Kontextdaten und Kontextdatenprognosen

Wie in den Abschnitten 3.5 und 4.2 erläutert wurde, stellt die Integration von Umgebungsinformationen zur Ableitung von Anpassungsbedarf insbesondere für die Prozessausführung auf mobilen Ausführungseinheiten einen wichtigen Aspekt dar. In diesem Abschnitt wird exemplarisch die Anbindung eines Kontextmanagementsystems erläutert, um die dynamische Verteilung der Prozessausführung auch auf Basis aktueller Umgebungsbedingungen vornehmen zu können und insbesondere die Entscheidung über die Verteilung durch die in Abschnitt 6.5 vorgestellte Kontextdatenprognose zu unterstützen. In Abschnitt 7.7.1 wird dazu zunächst in Kürze die Integration von Kontextdaten am Beispiel eines bestehenden Kontextmanagementsystems diskutiert. Darauf aufbauend wird in Abschnitt 7.7.2 die Implementierung des Rahmenwerks zur Kontextdatenprognose vorgestellt und deren Einsatz in der Middleware für dynamisch verteilt ausgeführte Prozesse erläutert.

7.7.1 Anbindung von Kontextmanagementsystemen

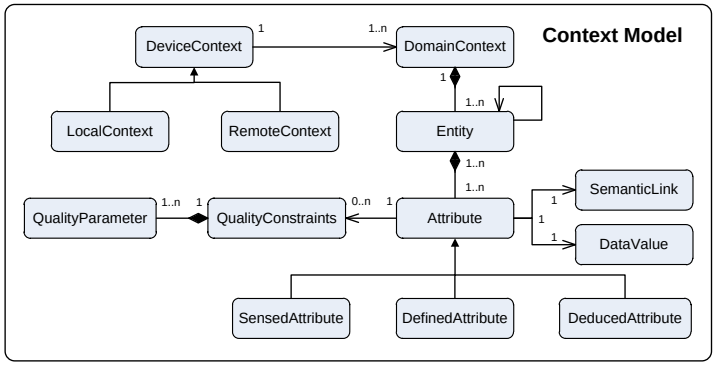
Ein Kontextmanagementsystem dient im Allgemeinen der Abstraktion von konkreten Sensoren zur Erhebung von Umgebungsinformationen und zur Aufbereitung von plattformspezifischen Repräsentationen der durch die Sensoren gewonnenen Kontext(roh)daten (vgl. Abschnitt 3.5.1). Im Rahmen dieser Arbeit erfolgt die Nutzung eines Kontextmanagementsystems einerseits als transparenter Bestandteil der verwaltbaren Ressource (vgl. Referenzmodell in Abbildung 6.29). Es stellt dabei eine wichtige Basis für kon-

textbasierte Migrationsentscheidungen dar und und dient im Speziellen als Datenquelle für die Vorhersage von Dienstverfügbarkeiten in bestimmten Ausführungsumgebungen (vgl. Abschnitt 7.7.2). Andererseits werden Kontextinformationen für die Anpassung von Benutzungsschnittstellen an den situativen Kontext des Benutzers als Prozessteilnehmer verwendet (vgl. Abschnitte 6.4 und 7.6).

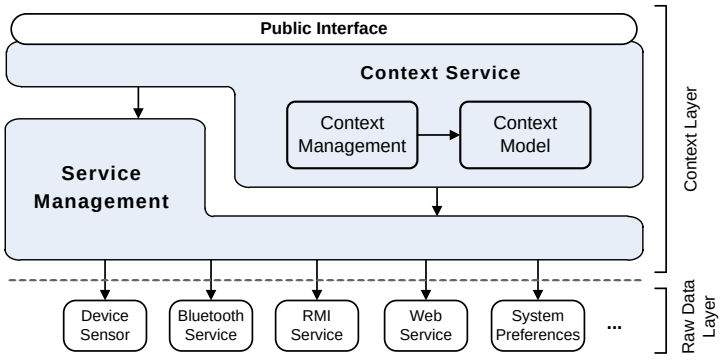
Um die kontextbasierte Ausführung von verschiedenartigen (dynamisch verteilten) prozessorientierten Anwendungen in unterschiedlichen Anforderungen zu unterstützen, ist ein möglichst generisch anwendbares Kontextdatenmetamodell und ein entsprechend anwendungsunabhängiges Kontextmanagementsystem von Vorteil [KZTL08, Kun08]. Abbildung 7.17 zeigt einen Überblick über ein solches Kontextdatenmetamodell und dessen Verarbeitung durch eine unterstützende Middleware-Komponente. Das Kontextdatenmetamodell (vgl. Abbildung 7.17a) erlaubt den Aufbau einer Struktur, welche den Kontext der lokalen Ausführungseinheit beschreibt (*lokal relevanter Kontext*). Eine Entität *Entität* (*Entity*) stellt hierbei ein Objekt mit seinen jeweiligen *Attributen* (*Attributes*) dar und dient der Repräsentation komplexer Kontextdaten [KZTL08, ZKL09b]. Für die Unterstützung des in Abschnitt 7.6 vorgestellten Interaktionsdienst stellen z. B. der Benutzer oder das von ihm verwendete Endgerät solche Entitäten dar. Die einzelne Entität „Endgerät“ besitzt dabei Attribute, welche die Erzeugung einer für den Benutzer geeigneten Benutzungsschnittstelle unterstützen, z. B. das Attribut „Displaygröße“. Die Entität „Benutzer“ definiert Attribute, welche seine momentane Situation angeben, z. B. das Attribut „Tätigkeit“. Das Element *Domäne* (*DomainContext*) fasst eine Menge von Entitäten für einen bestimmten Anwendungsbereich zusammen [KZTL08, ZKL09b]. Über diese Art von Filter erhält z. B. der Interaktionsdienst nur die für ihn relevanten Kontextinformationen. Alle für eine Ausführungseinheit relevante Domänen bilden schließlich den *Gerätekontext* (*Device Context*). Dieser setzt sich aus den verfügbaren Kontextinformationen des eigenen lokalen Kontextes (*Local Context*) sowie aus dem relevanten Kontext anderer (erreichbarer) Ausführungseinheiten (*Remote Context*) zusammen [KZTL08, ZKL09b]. Das für die dynamisch verteilte Prozessausführung zugrunde liegende Kontextmodell kann dabei für alle Ausführungseinheiten identisch sein oder nach dem Prinzip der *Kontextföderation* (vgl. [Kun08]) über verschiedene Ausführungseinheiten integriert werden.

Abbildung 7.17b zeigt die Einbettung des Kontextmodells in eine Softwarekomponente, welche für die Erhebung der relevanten Kontextdaten sowie für deren Aufbereitung und Bereitstellung für (beliebige) konsumierende Anwendungen verantwortlich ist (*Context Service*). Dabei wird für jedes der im Kontextmodell modellierten Attribute ein interner Dienst implementiert (*Service Management*), über welchen die aktuellen Kontext(roh)daten von externen Schnittstellen bzw. von konkreten Sensoren bezogen werden (*Raw Data Layer*).

Neben den bereits genannten Entitäten für den Kontext des Benutzers wurde als weitere exemplarische Kontextinformation die für den Einsatzbereich der Prozessausführung auf mobilen Ausführungseinheiten relevante Entität



(a) Kontextdatenmetamodell



(b) Verarbeitungskomponente

Abbildung 7.17: Anwendungsunabhängig einsetzbares Kontextmanagementsystem (KZTL08, ZKL09b)

Position modelliert. Über eine Simulation eines GPS-Empfängers werden hierzu Positionsangaben ermittelt und durch das Kontextmanagementsystem als Datensatz aus X- und Y-Koordinate abgelegt. Für die Prognose von Dienstverfügbarkeiten nach dem in Abschnitt 6.5.5 vorgestellten Prognosenetz ist als weiteres Beispiel die *Netzgröße* (*NetSize*) umgesetzt worden. Diese umfasst die Anzahl an über das Netzwerk zu erreichenden Ausführungseinheiten (d. h. PMaaS-Anbieter), welche bei Nutzung der unter Abschnitt 7.2 vorgestellten Dienstinfrastruktur direkt über die lokale Verwaltung des Overlay-Netzwerks abgefragt werden kann.

Konsumenten des Kontextmanagementsystems (d. h. hier der Interaktionsdienst der lokalen Ausführungsumgebung, die öffentliche Schnittstelle der verwaltbaren Ressource und die (erweiterbaren) Module zur Durchführung von Migrationsentscheidungen) verwenden eine einheitliche Schnittstelle (*Public interface*), um die vom Kontextmanagementsystem bereitgestellten Daten abzurufen. Durch diese Vorgehensweise wird der Zugriff auf die zu nutzenden Kontextdaten und dadurch die Entwicklung der oben genannten Middleware-Komponenten stark vereinfacht, da diese durch die vorgenommene Abstraktion für verschiedene Plattformen eingesetzt werden können. Im Folgenden wird die Nutzung von Kontextdaten am Beispiel der Prognose von Dienstverfügbarkeiten aufgezeigt.

7.7.2 Nutzung des Rahmenwerks zur Kontextdatenprognose

Das Verfahren zur strukturierten Kontextdatenprognose (vgl. Abschnitt 6.5) wird in der vorliegenden Middleware zur Unterstützung dynamisch verteilt ausgeführter Prozesse verwendet, um die Verfügbarkeit von bestimmten Diensten vorherzusagen. Dazu wird das in Abschnitt 6.5.5 definierte Prognosemodell verwendet. Die Prognosen werden von der prototypischen Implementierung des Migrationsdienstes verwendet, welcher ein Modul zur Implementierung der lokalen Richtlinie besitzt, dass Prozessinstanzen migriert werden sollen, sobald die für die anstehenden Aktivität aufzurufende Ressource in der Ausführungsumgebung der lokalen Prozess-Engine nicht verfügbar ist (vgl. Abschnitt 7.5.3). Die Vorhersage von Dienstverfügbarkeiten wird dabei verwendet, um die Migrationsentscheidung dieses Moduls zu verbessern. In Folge wird ein Prozess nur noch dann migriert, wenn die benötigte Ressource mit einer angegebenen Wahrscheinlichkeit in einem angegebenen Zeitraum voraussichtlich nicht zur Verfügung stehen wird.

Überblick über die Architektur des Prognosesystems

Abbildung 7.18 zeigt die grundlegende Architektur des Prognosesystems als Teil des Rahmenwerks zur strukturierten Kontextdatenprognose. Das Erkennen von Zusammenhängen (*Learning*) und die Vorhersage (*Prediction*) von Kontexten sind als nebenläufige Vorgänge auf zwei unterschiedliche Komponenten abgebildet. Beide Komponenten werden durch eine Wissenskomponente (*Knowledge*) miteinander verbunden. Durch das Erkennen von Zusammenhängen wird dabei Wissen erzeugt bzw. aktualisiert und steht der Vorhersage-Komponente zur Verfügung, um zur Laufzeit Prognosen für eine diese Funktionalität konsumierende Anwendung (*Application*) zu erstellen. Die drei genannten Komponenten haben jeweils Zugriff auf die implementierten Prognosemethoden (*Prediction methods*). Jede Methode besitzt ihr eigenes Wissen (z. B. Häufigkeiten von Variablenwerten in einer Wahrscheinlichkeitstabelle) und bietet ein Verfahren zur Aktualisierung des Wissens und zur Vorhersage der Variablenwerten von den mit der Prognosemethode verknüpften Variablen an [Mei09, MZL10, ZML11].

Im Detail verwaltet die Wissenskomponente die Eigenschaften und Zusammenhänge der beobachteten Kontextvariablen. Der erste Teil dieses Wissens kann dem Prognosenetz entnommen werden, welches aus dem gegebenen Prognosemodell (*Prediction model*) extrahiert wird. Der zweite Teil des Wissens liegt in Form von Instanzdaten (*Instance data*) vor, welche zur Laufzeit durch den Einsatz von adaptivem Online-Learning unter Verwendung der im Prognosemodell angegebenen Methoden erzeugt und aktualisiert werden (vgl. Meiners [Mei09]). Die Vorhersage-Komponente führt den in Abschnitt 6.5.3 vorgestellten Algorithmus aus, um die im Prognosemodell angegebenen Prognosemethoden zu integrieren. Um für eine aussagekräftige Prognose genügend historische Daten zu sammeln, bezieht die Komponente zum Erkennen von Zusammenhängen (*Learning*) in regelmäßigen (durch das Prognose-

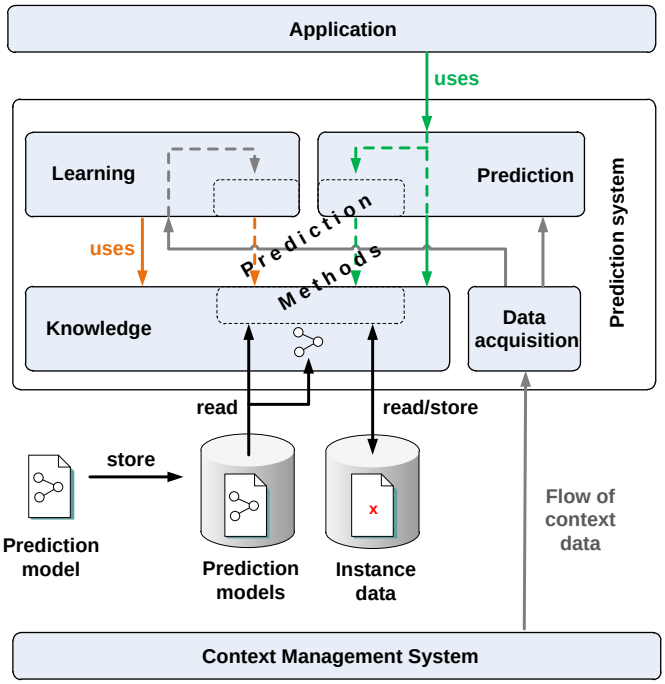


Abbildung 7.18: Architektur des Rahmenwerks zur strukturierten Kontextdatenprognose (Mei09, MZL10, ZML11))

modell vorgegebenen) Abständen Kontextdaten und gibt diese als Trainingsdaten in die angegebenen Prognosemethoden ein [Mei09, MZL10, ZML11]. Die hierfür erforderlichen Kontextdaten können dabei prinzipiell direkt aus Sensoren, aus einem Kontextmanagementsystem oder durch Kommunikation mit (anderen) Anwendungen oder Prognosesystemen gewonnen werden [Mei09, MZL10, ZML11]. In der exemplarischen Umsetzung zu dieser Arbeit werden die benötigten Daten über das in Abschnitt 7.7.1 genannte Kontextmanagementsystem gewonnen. Zusätzliches Anwendungswissen wird mit den Migrationsdaten der verteilt ausgeführten Prozesse transportiert und direkt als Instanzdaten abgelegt. Es wird dabei also über die konsumierende Anwendung selbst integriert.

Konfiguration für die Vorhersage von Dienstverfügbarkeiten

Für die Vorhersage von Dienstverfügbarkeiten wird das in Abschnitt 6.5.5 vorgestellte Prognosemodell in XML-Repräsentation ausgedrückt und an das oben dargestellte Prognosesystem übergeben. Als zentrale Prognosemethode werden einfache Wahrscheinlichkeitstabellen verwendet, welche die Häufigkeit des Auftretens der im Prognosemodell spezifizierten (diskreten) Variablenwerte über einen bestimmten Zeitraum aufzeichnen und diese entsprechend für Zukunftsprognosen über das Auftreten dieser Variablenwerte zur Verfügung stellen. Dabei kann über die relative Häufigkeit des Auftretens eines Variablenwertes in einem bestimmten Zusammenhang direkt eine Wahrscheinlichkeit abgeleitet werden, so dass neben dem Prognoseergebnis („Dienst x

```
Migrationdata
...
  AssignmentStrategy
    ElementlevelAssignment
      Element
        ID = Activity3
        Assignment
          StrategyID = org.vsis.pmaas.SCP-AwareRessourceUnavailable
          StrategyImpl
            ListOfAlternativeExecutionUnits = [...]
            SCP-Data
              PredictionModelID = ServiceAvailability
              Expression = V15:"Office":p=1
            End SCP-Data
          End StrategyImpl
        End Assignment
      End Element
    End ElementlevelAssignment
  End AssignmentStrategy
...
End Migrationdata
```

Listing 7.9: Integration von Instanzdaten für die strukturierte Kontextdatenprognose (vgl. Listing 7.6)

zum angefragten Zeitpunkt *verfügbar*“ oder „*nicht verfügbar*“) auch eine Qualitätsangabe für das Prognoseergebnis geliefert werden kann (z. B. „Dienst x mit einer Wahrscheinlichkeit von 98% verfügbar“).

Für die Integration von anwendungsspezifischen Instanzdaten in das Migrationsmodell muss die Struktur der Migrationsdaten nicht erweitert werden. Stattdessen werden die Daten in die Zuordnungsstrategie für die betreffende Aktivität integriert, zu deren Ausführung der gesuchte Dienst aufgerufen werden soll. Listing 7.9 zeigt exemplarisch die Eingabedaten für das Modul, welches die (neue) Zuordnungsstrategie *SCP-AwareRessourceUnavailable* implementiert. Hierbei enthält das Element *SCP-Data* die Referenz auf das Prognosemodell, für welches die Instanzdaten geliefert werden, sowie einen Ausdruck (*Expression*), welcher sich auf genau dieses Prognosemodell bezieht. In dem gezeigten Beispiel wird angegeben, dass für die Aktivität namens „Activity3“ gelten soll, dass die in der Prozessbeschreibung angegebene Ressource an einem diskreten Ort (Variable V_{15} des Prognosenetzes) namens „Office“ mit einer Wahrscheinlichkeit von $p = 1$ zu finden ist. In Folge kann auch ohne konkrete Erfahrungswerte zu der gesuchten Ressource nach (mobilen) Ausführungseinheiten gesucht werden, welche voraussichtlich in Kürze den angegebenen Ort erreichen werden.

Bei einer (verteilten) Ausführung der Prozessinstanz wird nach erfolgter Installation des Prozesses auf der lokalen Prozess-Engine durch den Migrationsmanager das angegebene Modul *SCP-AwareRessourceUnavailable* aktiviert, um die Migrationsentscheidung für die angegebene Aktivität zu steuern. Es wird dabei im Folgenden angenommen, dass alle potentiell an der verteilten Prozessausführung teilnehmenden Ausführungseinheiten das Prognosesystem, das Prognosemodell zur Vorhersage von Dienstverfügbarkeiten

und das Modul zur Entscheidung des Migrationsverhaltens auf Basis der Verfügbarkeitsprognose besitzen und für das Prognosemodell mit der Erhebung von Trainingsdaten beginnen. Da die Anzahl temporär verfügbarer Dienste bei Zugriff auf das Internet sehr groß werden kann, werden keine periodischen Abfragen beliebiger Dienstverfügbarkeiten zum Sammeln von Trainingsdaten implementiert. Stattdessen wird bei jeder Anfrage der Prozess-Engine nach einem bestimmten Dienst abgespeichert, ob die Anfrage erfolgreich verlaufen ist oder nicht. Zusätzlich werden bei jeder Dienstanfrage entsprechend des Prognosenetzes die Variablenwerte zu den Attributen *Position* und *NetSize* beim Kontextmanagementdienst abgerufen (vgl. Abschnitt 7.7.1). Durch diese Strategie werden zwar weniger Daten erhoben als prinzipiell möglich wäre, jedoch werden die Anfragen auf die für die jeweilige Ausführungseinheit *relevanten* Daten begrenzt. Somit werden bei jeder Prozess-Engine jeweils Erfahrungswerte über die Verfügbarkeit der individuell benötigten Dienste aufgebaut. Der für die Datenhaltung benötigte Speicherplatz kann hierdurch begrenzt werden (vgl. Abschnitt 8.4.1).

Das für die Durchführung von Migrationsentscheidungen verantwortliche Modul des lokalen Migrationsdienstes abonniert zur Umsetzung der genannten Datenerhebung für alle lokal ausgeführten Aktivitäten das Ereignis *ActivityStateChanged* (ohne blockierende Wirkung) für den Zustand *executed* und lässt sich für jede Aktivität den ausführenden Teilnehmer (*Performer*) angeben (vgl. Tabellen 7.7 und 7.11). Handelt es sich hierbei um einen Dienst, werden die Daten dem Prognosesystem übergeben. Desweiteren wird für alle Aktivitäten das Ereignis *On Error* mit blockierender Wirkung für das Auftreten der Fehlermeldung *Resource unavailable* abonniert (vgl. Abschnitt 7.5.4). Kann zur Ausführung einer Aktivität temporär kein geeigneter Dienst gefunden werden, so wird für den angegebenen Dienst eine Prognose aufgerufen. Dazu wird folgende Vorgehensweise durchgeführt:

1. Liegen in den Migrationsdaten des Prozesses Informationen zur Dienstverfügbarkeit vor, welche mit den bereits gesammelten Erfahrungswerten integriert werden können, so werden die Angaben aus den Migrationsdaten zunächst mit den lokalen Instanzdaten integriert.

2. Da eine Migration des Prozesses mit einem gewissen Aufwand verbunden ist, wird zunächst durch eine Prognose geprüft, ob der Dienst in absehbarer Zeit bei der lokalen Ausführungseinheit verfügbar sein wird. Fällt das Prognoseergebnis mit einer annehmbaren Wahrscheinlichkeit positiv aus, so wird die Prozessinstanz zunächst angehalten (Operation *Suspend*) und nach einer Wartezeit erneut geprüft, ob der Dienst verfügbar ist (Operation *Restart Activity*).

3. Fällt das Prognoseergebnis für die lokale Dienstverfügbarkeit negativ aus oder ist die Wahrscheinlichkeit einer positiven Dienstverfügbarkeit zu gering, so werden die Prognosesysteme anderer (potentieller) Ausführungseinheiten mit einer Prognose über die Verfügbarkeit

des betreffenden Dienstes beauftragt. Auf Basis der eingehenden Ergebnisse migriert der aktuell verantwortliche Migrationsmanager dann den Prozess zu der Ausführungseinheit, deren Prognosedienst die höchste Wahrscheinlichkeit für die Verfügbarkeit des Dienstes in dem angefragten Zeitraum angibt (vgl. [Mei09]).

Können Ausführungseinheiten nicht an der (verteilten) Kontextdatenprognose teilnehmen, so wird die Prozessausführung hierdurch nicht beeinträchtigt. Die Prozessinstanz kann bei Vorliegen aller relevanten Rahmenbedingungen auch ohne Prognose von Dienstverfügbarkeiten an andere Ausführungseinheiten weitergereicht werden. Die Integration von Kontextdatenprognosen stellt daher eine *optionale* Möglichkeit zur Verbesserung der Migrationsentscheidung dar, welche sich flexibel in die hier vorgestellte Middleware einbetten lässt.

7.8 Zusammenfassung

In diesem Kapitel wurde ein Vorschlag für eine technische Infrastruktur aus dienstbasierten Middleware-Komponenten zur Umsetzung der in Kapitel 6 erarbeiteten Flexibilisierungsstrategien für verteilt ausgeführte Prozesse vorgestellt. Auf Basis einer grundlegenden Web-Service-Architektur für das dynamische Anbieten und Nutzen von Diensten in heterogenen Systemumgebungen wurden dazu zunächst die wesentlichen Eigenschaften und Operationen von *Prozessmanagementsystemen als verwaltbare Ressourcen* mit dem Standard *Web Service Distributed Management* (*WSDM*) abgebildet und über eine einheitliche Schnittstelle bereitgestellt. Die hierdurch entstehenden Möglichkeiten zur Überwachung und Steuerung der auf dem gekapselten Prozessmanagementsystem (entfernt) ausgeführten Prozesse bzw. Prozesspartitionen stehen dadurch zur Verfügung, um Daten für eine etwaige Anpassung der Prozessausführung zu erheben, ohne den fachlichen Prozess dabei zu verändern.

Darauf aufbauend wurden mit dem *Managementdienst* und dem *Migrationsdienst* zwei weitere Middleware-Komponenten vorgestellt, welche die grundlegende Funktionalität dieser Schnittstelle konsumieren und auf dieser Basis höherwertige Funktionalitäten zur dynamischen Anpassung der (verteilten) Prozessausführung zur Verfügung stellen. Der Managementdienst erlaubt exemplarisch den Aufruf beliebiger Dienste als Reaktion auf die Erkennung von Ereignismustern, welche durch Ausdrücke der *Event Processing Language* (*EPL*) spezifiziert und mittels der *Esper*-Plattform zur Verarbeitung komplexer Ereignisse auf eingehende Ereignisdatenströme angewendet werden können. Der Migrationsdienst setzt das Paradigma des *Process-Management-as-a-Service* (*PMaaS*) für verteilt ausgeführte Prozesse um. Für die fortwährende Anpassbarkeit der Verteilung wurde dazu die Migration laufender Prozessinstanzen unter Beibehaltung des unveränderten Prozessmodells als Ausführungsanweisung für beteiligte Ausführungseinheiten umge-

setzt. Zur Erkennung von Anpassungsbedarf wurde zudem eine Architektur zur Umsetzung benutzerdefinierter Rahmenbedingungen für die dynamische Verteilung von Prozessinstanzen durch eine erweiterbare Menge von Softwaremodulen zur Durchführung von Migrationsentscheidungen vorgestellt.

Für die Darstellung von kontextbasierten Benutzungsschnittstellen für dynamisch verteilt ausgeführte Prozesse wurde ein *Interaktionsdienst* vorgeschlagen, welcher auf Basis einer abstrakten Beschreibung der vorzunehmenden Interaktion und des aktuellen Kontextes der Ausführungsumgebung eine lokal angepasste Benutzungsschnittstelle erzeugen kann. Als Beispiel für eine Anpassung der verteilten Prozessausführung unter Berücksichtigung der Annahmen über zukünftige Entwicklungen wurde schließlich die Vorhersagen von Dienstverfügbarkeiten durch *strukturierte Kontextdatenprognose* umgesetzt und in den Migrationsdienst integriert.

Der dargestellte Vorschlag für die prototypische Realisierung einer Middleware zur Unterstützung dynamisch verteilt ausgeführter Prozesse zeigt, dass die in Kapitel 6 erarbeiteten Konzepte sowohl individuell als auch aufeinander aufbauend umsetzbar sind. Insbesondere kann festgestellt werden, dass die Funktionalitäten des als verwaltbare Ressource bereitgestellten Prozessmanagementsystems nicht nur für die Auftraggeber einer (entfernt) ausgeführten Prozesspartition, sondern auch für Konsumenten in Form von lokalen Anwendungen zur Bereitstellung höherwertiger Dienste eine wertvolle Basis darstellen. Es konnte zudem gezeigt werden, dass die Umsetzung der vorgeschlagenen Konzepte zum Großteil unter Verwendung bestehender Technologien vorgenommen werden kann und dass eine Integration in bestehende Infrastrukturen (insbesondere in dienstorientierte Architekturen) möglich ist.

Für die Umsetzung der Middleware-Komponenten und deren Interaktion untereinander sind dabei vielfach Standards einsetzbar, welche die Interoperabilität zwischen verschiedenen Ausführungseinheiten einer verteilten Prozessausführung unterstützen können. Dies ist insbesondere für die Abbildung der verwaltbaren Schnittstelle mittels WSDM und den Austausch von Prozessmodellen, welche in standardisierten Prozessbeschreibungssprachen verfasst werden, der Fall. Auch für die Spezifikation von Management-Regeln können wie im Beispiel von EPL-basierten Regeln etablierte Technologien eingesetzt werden. Würden alle hier vorgestellten Middleware-Komponenten miteinander zum Einsatz kommen, so besteht jedoch die Notwendigkeit, dass dabei von allen (potentiellen) Teilnehmern die Summe der genutzten Standards bzw. Technologien unterstützt werden muss. Dies kann unter Umständen zu einer Art „Markteintrittsbarriere“ für Ausführungseinheiten führen, welche nur eine Teilmenge der benötigten Middleware-Komponenten unterstützen, und somit durch die Einschränkung der möglichen Verteilungsziele wiederum die Flexibilität beeinträchtigen. Viele der vorgestellten Komponenten sind daher neben ihrer Rolle im Verbund der hier vorgeschlagenen Middleware auch individuell einsetzbar und in ihrer Anwendbarkeit nicht nur auf das spezielle Problem der Flexibilisierung (dynamisch) verteilt ausgeführter Prozesse beschränkt. Insgesamt sind daher im Rahmen dieser Arbeit aus

den Überlegungen zur Realisierung der Flexibilisierungsstrategien unter den in Abschnitt 4.4 genannten Anforderungen die folgenden voneinander unabhängigen Beiträge entstanden:

- eine Softwarearchitektur für das dynamische Anbieten und Nutzen von Diensten in heterogenen Systemumgebungen, insbesondere unter Berücksichtigung dynamischer lokaler Netze und mobiler Dienstnutzer und -konsumenten [ZDL09a, ZDL09b],
- die Abbildung eines Prozessmanagementsystems als verwaltbare Ressource mit WSDM zur Bereitstellung von Eigenschaften und Operationen zur dienst- und ereignisbasierten Überwachung und Steuerung der auf diesem Prozessmanagementsystem ausgeführten Prozesse, unter besonderer Berücksichtigung einer verteilten Ausführung der Prozesse, jedoch ohne Beschränkung auf ein bestimmtes Verteilungsmodell [ZBH+10],
- die exemplarische Umsetzung eines *Managementdienstes* zur automatischen Überwachung der Prozessausführung auf Basis von EPL-Ereignismustern und zur Durchführung von Reaktionen im Form von beliebigen Dienstaufrufen [ZBH+10],
- die Architektur eines Migrationsdienstes als Umsetzung des Paradigmas des *Process-Management-as-a-Service* für verteilt ausgeführte Prozesse und für die dynamische Verteilung bislang zentral ausgeführter Prozessinstanzen mit Möglichkeiten zur benutzerdefinierten Einschränkung der Verteilung [ZKML10],
- die Architektur eines Interaktionsdienstes zur kontextbasierten Darstellung von Benutzungsschnittstellen auf Basis der Eingabe einer abstrakten Beschreibung der vorzunehmenden Interaktion für dynamisch verteilt ausgeführte Prozesse oder für Prozesse, deren Instanzen auf Endgeräten mit unterschiedlichen Benutzungsschnittstellen ausgeführt werden müssen [ZKML10, ZL10].
- die Anwendung der strukturierten Kontextdatenprognose zur Vorhersage von Dienstverfügbarkeiten für die Optimierung der Prozessausführung auf Ausführungseinheiten in dynamischen lokalen Netzen [ZML11]

Die prototypisch realisierten Komponenten der hier vorgestellten Middleware werden im Folgenden auf konkrete Prozessmanagementsysteme und Prozessbeschreibungssprachen angewendet und im Rahmen der in Abschnitt 4.2 erläuterten Fallbeispiele evaluiert.

8 Anwendung und Bewertung

Die Betrachtung von Flexibilisierungsstrategien und die prototypische Umsetzung von Middleware-Komponenten zur Unterstützung einer dynamisch verteilten Prozessausführung erfolgten bislang auf Basis der in Kapitel 2 erarbeiteten abstrakten Eigenschaften und Gemeinsamkeiten prozessorientierter Anwendungen und ihrer computergestützten Ausführung, der in Kapitel 3 vorgenommenen Untersuchung der Flexibilität prozessorientierter Anwendungen und der in Kapitel 4 durchgeführten Anforderungsanalyse der verteilten Ausführung auf der Basis von klassischen Verteilungsmodellen und aktuellen Fallbeispielen. Um die praktische Einsetzbarkeit der für die Flexibilisierung verteilter Prozessausführung entwickelten Lösungsansätze zu überprüfen, erfolgt in diesem Kapitel zunächst eine exemplarische Anwendung der vorgeschlagenen Konzepte auf konkrete Ausführungssysteme, Prozessbeschreibungssprachen und Problemstellungen. Im Anschluss wird darauf aufbauend eine Bewertung hinsichtlich der durch die Umsetzung erreichbaren Flexibilität und des resultierenden Anpassungsaufwands vorgenommen.

Da Flexibilität im Allgemeinen nicht quantitativ messbar ist, wird in Abschnitt 8.1 zunächst auf das angewendete Vorgehen für die Bewertung der Flexibilität eingegangen und dabei die Ableitung geeigneter Bewertungskriterien vorgenommen. Für die Analyse der allgemeinen Anwendbarkeit wird darauf folgend die prototypische Integration der vorgestellten Middleware-Komponenten auf drei unterschiedliche, bereits bestehende Prozessmanagementsysteme für mobile und stationäre Infrastrukturen erläutert (vgl. Abschnitt 8.2). Darauf aufbauend wird die dynamische Verteilung von prozessorientierten Anwendungen der auf diesen Systemen unterstützten standardisierten Prozessbeschreibungssprachen XPDL, WS-BPEL und BPMN untersucht (vgl. Abschnitt 8.3). Die Analyse der praktischen Anwendbarkeit schließt mit der Anwendung der Lösungsstrategien auf die eingangs vorgestellten Fallbeispiele (vgl. Abschnitt 4.2) und den hierbei erzielten Ergebnissen (vgl. Abschnitt 8.4).

Die aus dem praktischen Teil der Anwendung gewonnenen Erkenntnisse werden im Anschluss in einer integrierten Flexibilitätsbetrachtung zusammengefasst (vgl. Abschnitt 8.5). Hierbei wird eine Bewertung des erzielten Flexibilitätsgewinns und des hierfür erforderlichen Aufwands im Vergleich mit dem allgemeinen Ansatz der physischen Partitionierung von Prozessen vorgenommen. Als Verallgemeinerung des zuvor exemplarisch dargestellten Nutzenpotentials erfolgt eine Überprüfung der bei der initialen Anforderungsdefinition (vgl. Abschnitt 4.4) erarbeiteten Rahmenbedingungen. Das Kapitel schließt mit einer Zusammenfassung der Bewertung und deren Implikation für diese Arbeit (vgl. Abschnitt 8.7).

8.1 Übersicht und Vorgehensweise

Wie in Abschnitt 3.1 diskutiert wurde, liegt der Bewertung von Flexibilität in soziotechnischen Systemen oft eine anwendungsabhängige Einschätzung des für die Anpassung eines Objektes notwendigen Aufwands und ein nach bestimmten Kriterien zu definierender Flexibilitätsgewinn zugrunde. Im Kontext dynamisch verteilt ausgeführter Prozesse ist die *Anpassung* dabei primär durch den Vorgang der individuellen Verteilung einer einzelnen Prozessinstanz determiniert. Da Prozesse bzw. Prozesspartitionen im Zuge der Verteilung entfernt ausgeführt werden können, sind des weiteren sekundär auch die Maßnahmen zur Erhebung von Informationen zur Ableitung von Anpassungsbedarf für die Einschätzung des Aufwands einzubeziehen. Als Bezugsgröße wird hierbei die zentrale (d. h. nicht-verteilte) Ausführung einer Prozessinstanz gegenübergestellt. Damit kann der für die Anpassung notwendige *Aufwand* als alle Maßnahmen angesehen werden, welche ergänzend zu einer zentralen Ausführung des Prozesses vorgenommen werden müssen, um eine nach bestimmten (benutzerdefinierten) Kriterien verteilte Ausführung eines Prozesses zu erreichen. Der durch die vorgestellten Flexibilisierungsstrategien entstehende Aufwand lässt sich damit in drei unterschiedlich zu berücksichtigende Einheiten unterteilen:

1. Aufwand für die einmalig notwendige Anpassung zur Einrichtung von (ergänzenden) Middleware-Komponenten, wobei der Aufwand zur Entwicklungszeit jeder potentiell teilnehmenden Prozessausführungseinheit auftritt,

2. Aufwand für eine etwaige (einmalig oder mehrfach) notwendige Anpassung der Prozessmodelle für die Verteilung hiervon zukünftig abgeleiteter Prozessinstanzen, wobei der Aufwand zur Entwicklungszeit jedes potentiell zu verteilenden Prozessmodells auftritt, und

3. Aufwand für die bei der Erhebung von Daten oder der Verteilung einer Prozessinstanz durchzuführende Anpassung, wobei der Aufwand zur Laufzeit jeder potentiell zu verteilenden oder bei jeder tatsächlich zu verteilenden Prozessinstanz bei jeder Erhebung von Daten bzw. bei jeder Verteilung oder auch grundsätzlich auftreten kann.

Für die Bewertung der Flexibilität ist nun jeweils die Verhältnismäßigkeit zwischen Aufwand und den durch die Maßnahmen der Flexibilisierung entstehenden positiven Effekte zu analysieren. Dem Aufwand für die einmalig notwendige Anpassung zur Einrichtung der Middleware-Komponenten (vgl. o.g. Aufzählungspunkt 1) steht zunächst kein direkter Flexibilitätsgewinn gegenüber. Als Teil einer Durchführbarkeitsanalyse und um eine Einschätzung des Aufwands für die einmalig notwendigen Anpassungen zu gewinnen, wird im folgenden Abschnitt die exemplarische Anwendung der vorgeschlagenen Middleware-Komponenten auf drei bereits bestehende – teilweise für mobile

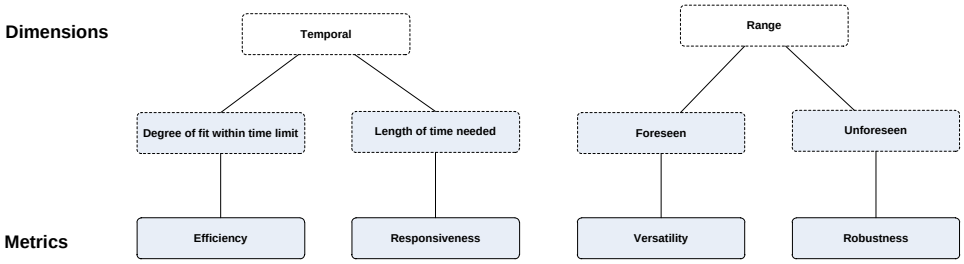


Abbildung 8.1: Beziehungen von Dimensionen und Messgrößen von Flexibilität nach GOLDEN und POWELL (GP00))

Einsatzszenarien geeignete – Prozessmanagementsysteme vorgenommen (vgl. Abschnitt 8.2).

Da für die in dieser Arbeit vorgeschlagene verteilte Ausführung prozessorientierter Anwendungen nicht die Anwendungen selbst anzupassen sind, wird im Gegenzug der Aufwand für die unter dem oben genannten Aufzählungspunkt 2 genannte Anpassung von Prozessmodellen weitestgehend minimiert. Es ist daher zu prüfen, ob die Maßnahmen zur Flexibilisierung zu einem angemessenen Anpassungsspielraum führen. Um dies exemplarisch nachzuweisen, wird in Abschnitt 8.3 die dynamische Verteilung von Prozessen durch die Migration von Prozessinstanzen auf die grundlegenden Kontrollflusskonstrukte dreier Standard-Prozessbeschreibungssprachen angewendet und die resultierenden *Verteilungsspektren* diskutiert.

Um den Aufwand und den Flexibilitätsgewinn für die bei jeder Erhebung von Daten oder der Verteilung eines Prozesses zur Laufzeit durchzuführende Anpassung zu konkretisieren (vgl. o. g. Aufzählungspunkt 3), werden im Folgenden aus den Erkenntnissen der in den Abschnitten 3.1 und 3.2 durchgeführten Flexibilitätsbetrachtung anwendungsspezifische Kriterien für die Bewertung gebildet. Als Grundlage dient das multidimensionale Modell von GOLDEN und POWELL [GP00] zur Definition und Bewertung von Flexibilität über *Zeit*, *Anpassungsspielraum*, *Verhalten* und *Wirkung*, wobei nur die Dimensionen *Zeit* und *Handlungsspielraum* als nicht-binäre Größen eine metrische Betrachtung rechtfertigen (vgl. Abschnitt 3.2). Abbildung 8.1 zeigt die von GOLDEN und POWELL vorgeschlagenen Beziehungen zwischen diesen beiden Dimensionen der Flexibilität und den hierfür vorgeschlagenen Bewertungskriterien.

Bei der dargestellten *Effizienz der Anpassung* (*Efficiency*) steht im Vordergrund, dass Anpassungen vorgenommen werden können, ohne dass die Leistung des Systems während der Anpassung wesentlich beeinträchtigt wird [GP00]. In Bezug auf die verteilte Ausführung von Prozessen muss daher hierfür untersucht werden, ob und wie es durch die alleinige Bereitstellung der Möglichkeit zur Anpassung (d.h. ohne Durchführung einer Anpassung) bzw. durch die Anpassung selbst (d.h. durch die Durchführung der Anpassung) hinsichtlich der Ausführungszeit zu einer negativen Beeinträchtigung der Prozessausführung kommt bzw. kommen kann. Es werden daher zu diesem Aspekt die folgenden Bewertungskriterien abgeleitet:

- Relative Veränderung der Ausführungszeit nicht-verteilt ausgeführter Prozessinstanzen eines Prozessmodells PM_1 durch die Möglichkeit zur Erhebung von Daten oder zur Verteilung einer Prozessinstanz eines anderen Prozessmodells PM_2.
- Relative Veränderung der Ausführungszeit nicht-verteilt ausgeführter Prozessinstanzen eines Prozessmodells PM_1 durch die Möglichkeit zur Erhebung von Daten oder zur Verteilung einer Prozessinstanz desselben Prozessmodells PM_1.
- Relative Veränderung der Ausführungszeit nicht-verteilt ausgeführter Prozessinstanzen eines Prozessmodells PM_1 durch die tatsächliche Erhebung von Daten oder der tatsächlichen Verteilung einer Prozessinstanz eines anderen Prozessmodells PM_2.
- Relative Veränderung der Ausführungszeit nicht-verteilt ausgeführter Prozessinstanzen eines Prozessmodells PM_1 durch die tatsächliche Erhebung von Daten oder der tatsächlichen Verteilung einer Prozessinstanz desselben Prozessmodells PM_1.

Die in Abbildung 8.1 dargestellte *Reaktionszeit* (*Responsiveness*) bezeichnet den Zeitbedarf, welcher für die Anpassung selbst notwendig ist [GP00]. In Bezug auf die verteilte Ausführung von Prozessen muss hierfür untersucht werden, wie viel Zeit vom Zeitpunkt der Erhebung von Daten zur Feststellung des Anpassungsbedarfs, über die Berechnung der durchzuführenden Anpassungsmaßnahmen bis hin zur Fertigstellung der Verteilung, d. h. bis zur Wiederaufnahme der Prozessausführung benötigt wird. Zur Bewertung des Zeitaufwands wird ferner das Kriterium der *fortwährenden Anpassbarkeit* herangezogen, welches die Fähigkeit zur Wiederholung von Anpassungsmaßnahmen über die gesamte Laufzeit einer Prozessinstanz bezeichnet. Es werden daher die folgenden Bewertungskriterien abgeleitet:

- Differenz der gesamten Ausführungszeit einer einmalig dynamisch verteilt ausgeführten Prozessinstanz $P_{m,v}$ zu der gesamten Ausführungszeit einer zentral (d. h. nicht-verteilt) ausgeführten Prozessinstanz $P_{m,z}$ desselben Prozessmodells unter denselben Eingabeparametern und Ausführungsbedingungen.
- Zeitbedarf für die einzelnen Anpassungsschritte einer mehrfach dynamisch verteilt ausgeführten Prozessinstanz zur Bewertung, ob der Zeitbedarf für die Anpassung bei jeder Anpassung ähnlich (gering) ist oder mit zunehmender Anzahl der Anpassungen ansteigt oder abnimmt.

In Hinsicht auf die Dimension des Anpassungsspielraum wird nach GOLDEN und POWELL [GP00] die *Vielfältigkeit an Möglichkeiten* (*Versatility*) bewertet, die einem System zur Verfügung stehen, um auf erwartete Situationen reagieren zu können (vgl. Abbildung 8.1). Analog dazu dient das Kriterium der *Robustheit* (*Robustness*) als Grad der Anpassungsfähigkeit an uner-

wartete Situationen [GP00]. In Bezug auf die verteilte Ausführung von Prozessen muss hierfür untersucht werden, inwieweit eine Erhebung von Daten über die Prozessausführung sowie eine Verteilung des Prozesses möglich ist, falls der Bedarf hierfür bereits vor der Instantiierung eines Prozesses bekannt ist, und inwieweit eine Erhebung von Daten über die Prozessausführung sowie eine Verteilung des Prozesses möglich ist, falls der Bedarf hierfür erst zur Laufzeit der Prozessinstanz festgestellt wird. Dabei werden insbesondere die Auswahl der zu erhebenden Daten, die Anpassung der Art der Datenerhebung, die Granularität der Verteilung und die Auswahl der Zielorte der resultierenden Prozesspartitionen betrachtet. Zur Bewertung der Anpassungsfähigkeit an unerwartete Situationen wird ferner ebenfalls das Kriterium der *fortwährenden Anpassbarkeit* herangezogen, welches die Fähigkeit zur Wiederholung der oben genannten Anpassungsmaßnahmen über die gesamte Laufzeit einer Prozessinstanz bezeichnet. Es werden zusammenfassend die folgenden Bewertungskriterien hierfür abgeleitet:

- Umfang der Möglichkeiten zur Anpassung der Datenerhebung, falls die zu erhebenden *Datenobjekte* und die *Art der Datenerhebung* vor Beginn der Ausführung der von der Datenerhebung betroffenen Prozessinstanz bekannt sind.
- Umfang der Möglichkeiten zur Anpassung der Datenerhebung, falls die zu erhebenden *Datenobjekte* und die *Art der Datenerhebung* erst während der Ausführung der von der Datenerhebung betroffenen Prozessinstanz bekannt werden.
- Umfang der Möglichkeiten zur Anpassung der Datenerhebung, falls die *Notwendigkeit* zur Datenerhebung erst während der Ausführung der von der Datenerhebung betroffenen Prozessinstanz bekannt wird.
- Umfang der Möglichkeiten zur Anpassung der Datenerhebung, falls eine *wiederholte Anpassung* der Datenerhebung notwendig wird.
- Umfang der Möglichkeiten zur Anpassung der Verteilung, falls die *Granularität* der Verteilung und die Auswahl der *Zielorte* der Verteilung vor Beginn der Ausführung der von der Verteilung betroffenen Prozessinstanz bekannt sind.
- Umfang der Möglichkeiten zur Anpassung der Verteilung, falls die *Granularität* der Verteilung und die Auswahl der *Zielorte* der Verteilung erst während der Ausführung der von der Verteilung betroffenen Prozessinstanz bekannt werden.
- Umfang der Möglichkeiten zur Anpassung der Verteilung, falls die *Notwendigkeit* zur Verteilung erst während der Ausführung der von der Verteilung betroffenen Prozessinstanz bekannt wird.
- Umfang der Möglichkeiten zur Anpassung der Verteilung, falls eine *wiederholte Anpassung* der Verteilung notwendig wird.

Ob Aufwand und Nutzen zueinander in einem angemessenen Verhältnis stehen, ist letztendlich eine Frage des jeweiligen Anwendungsfalls. Die Bewertung eines Flexibilitätsgewinns unterliegt daher neben einer solchen grundlegenden theoretischen Betrachtung insbesondere anwendungsspezifischen Kriterien, welche durch den konkreten Einsatz des hier vorgeschlagenen Konzepts im Rahmen der in Abschnitt 4.2 vorgestellten Fallstudien überprüft werden (vgl. Abschnitt 8.4).

Der Begriff der *Flexibilisierung* besagt zudem, dass es gegenüber einem Ist-Zustand durch die durchgeführten Maßnahmen zu einer Verbesserung der Flexibilität kommt. Anwendungsunabhängig wird daher angenommen, dass Aufwand und Nutzen in jedem Fall zueinander in einem angemessenen Verhältnis stehen, wenn der Aufwand nicht höher ist als bei anderen Verfahren zur verteilten Ausführung, Überwachung und Steuerung von Prozessen und gleichzeitig der erzielbare Nutzen gleich oder höher ist. Der Aufwand für die bei jeder Erhebung von Daten oder der Verteilung eines Prozesses durchzuführende Anpassung wird daher im weiteren Verlauf dieses Kapitels im Vergleich mit anderen Ansätzen hinsichtlich der identifizierten Flexibilitätskriterien bewertet. Exemplarisch wird hierfür das allgemeine Modell der *horizontalen Partitionierung von Prozessen* herangezogen. Hierunter sei im Folgenden die physische Zerteilung eines Prozessmodells in mehrere Prozesspartitionen, die Anreicherung der Prozesspartitionen mit zusätzlichen Anweisungen zur Übergabe des Kontrollflusses zwischen den Prozesspartitionen und die Verteilung der Prozesspartitionen auf Ausführungseinheiten verstanden (vgl. Abschnitt 5.4). Dabei sei es auch möglich, Prozesspartitionen zur Entwicklungszeit zu duplizieren und auf verschiedene Ausführungseinheiten zu verteilen, um zur Laufzeit eine Wahlmöglichkeit zwischen den Ausführungseinheiten zu erreichen. Für die Überwachung und Steuerung der verteilten Prozessausführung wird zum Vergleich der Ansatz der *Integration von Management-Aktivitäten* in die fachliche Prozessbeschreibung herangezogen, welche während der Ausführung des Prozesses einen zentralen Dienst zur Übergabe von Informationen über den Zustand der Prozessausführung bzw. zur Abfrage von Steuerungsanweisungen zugreifen (vgl. Abschnitt 5.7.3). Eine ergänzende Betrachtung anderer Ansätze erfolgt im Einzelfall.

8.2 Anwendung auf bestehende Prozessmanagementsysteme

Um den Anpassungsaufwand an bestehenden Systemen zur Ausführung von prozessorientierten Anwendungen zu untersuchen und die allgemeine Anwendbarkeit der vorgeschlagenen Flexibilisierungsstrategien und deren technische Umsetzung zu überprüfen, wurden drei Prozessmanagementsysteme mit unterschiedlichen Eigenschaften zugrunde gelegt und jeweils um ausgewählte Komponenten der Middleware für dynamisch ausgeführte Prozesse erweitert. Dabei wurden aufgrund deren Relevanz für verteilt ausgeführte Prozesse zwei Prozessmanagementsysteme ausgewählt, welche für einen Einsatz auf mobilen Endgeräten geeignet sind. Die Prozess-Engine der kontext-

sensitiven Middleware *DEMAC* stellt dabei einen Vertreter für ein Prozessmanagementsystem dar, welches bereits auf Verteilung ausgelegt ist und durch die Möglichkeit zu einem dynamischen (Re-)Binding von elektronischen Diensten ein großes Nutzenpotential für die Migration laufender Prozessinstanzen eröffnet (vgl. Abschnitt 8.2.1). Die Prozess-Engine *Sliver* ist mit der Ausführung von WS-BPEL-Prozessen auf mobilen Endgeräten ein Vertreter für eine mobile, aber dennoch bisher zentrale (d. h. nicht verteilte) Prozessausführung (vgl. Abschnitt 8.2.2). Als Vertreter für ein typisches serverbasiertes Prozessmanagementsystem ohne native Unterstützung für mobil oder verteilt ausgeführte Prozesse wurde die Prozess-Engine von *Activiti* um die Möglichkeit zur dynamisch verteilten Prozessausführung erweitert (vgl. Abschnitt 8.2.3). Da die hier erweiterten Prozessmanagementsysteme gleichzeitig für die Durchführung von Testfällen zur Bewertung der Migrierbarkeit von Prozessen verwendet werden (vgl. Abschnitt 8.3), wurde bei der Auswahl darauf geachtet, dass mit *XPDL*, *WS-BPEL* und *BPMN* jeweils unterschiedliche Prozessbeschreibungssprachen unterstützt werden. Zudem stellen die erweiterten Prozessmanagementsysteme die Grundlage für die exemplarische Umsetzung der in Abschnitt 4.2 vorgestellten Fallbeispiele dar.

Von den genannten Prozessmanagementsystemen lag jeweils eine ausführbare Version und der veränderbare Quellcode für die Anbindung der in Kapitel 7 vorgeschlagenen Middleware-Komponenten vor. Da die im Rahmen der Bewertung des Anpassungsaufwands zu analysierenden Änderungen zum großen Teil aus manuellen Handlungen (d. h. insbesondere Entwicklungsarbeit und Konfigurationen) bestehen, ist die Messung des Zeitbedarfs hierfür durch eine hohe Subjektivität geprägt. Es kann daher nur auf Basis einer einfachen Aufwandsschätzung in Form von *Personenstunden* eine vergleichbare Bewertung abgeleitet werden. Dabei wird im Folgenden die Personenstunde als Zeiteinheit angesehen, in welcher die Arbeitsleistung einer Person mehr oder weniger exklusiv für das Bearbeiten einer Aufgabe zur Verfügung steht.

8.2.1 DEMAC

Die am Fachbereich Informatik der Universität Hamburg entwickelte Middleware *DEMAC* (*Distributed Environment for Mobility-Aware Computing*) [Kun05, Kun08] stellt eine prototypische Implementierung für die technische Umsetzung des Konzepts der *kontextbasierten Kooperation* dar (vgl. Abschnitt 4.2.4). Die dort integrierte Prozess-Engine führt Prozesse der proprietären Prozessbeschreibungssprache *DPDL* (*DEMAC Process Description Language*) aus, welche auf einer Teilmenge von XPDL 1.0 [NM02] basiert und zusätzlich Elemente zur Beschreibung von Verteilungsanweisungen und zur Beschreibung des aktuellen Prozesszustandes enthält [KZL06, ZK07, Kun08]. Ein angebundenes Kontextmanagementsystem liefert der Prozess-Engine auf Anfrage Informationen über die in der aktuellen Ausführungsumgebung aufgefundenen Dienste, welche für die Prozessausführung aufgerufen werden

können. Stehen die benötigten Dienste nicht unter den in der Prozessbeschreibung angegebenen Rahmenbedingungen zur Verfügung, initiiert die Prozess-Engine über ein Overlay-Netzwerk den Transfer des Prozesses an ein anderes (mobiles) Gerät, welches ebenfalls über die relevanten *DEMAC*-Middleware-Komponenten verfügt. Da in den DPDL-Prozessmodellen über eine Umsetzung des Konzepts der *abstrakten Dienstklassen* (vgl. Abschnitt 6.2.3) nur eine abstrakte Beschreibung der benötigten Ressourcen als UUID erfolgt, ist ein dynamisches Binden von verschiedenen Diensten durch unterschiedliche Prozess-Engines der *DEMAC*-Middleware möglich [KZL07a, Kun08].

Gegenüber der bestehenden Version der *DEMAC*-Middleware (vgl. [Kun08]) wurden im Rahmen dieser Arbeit die folgenden zusätzlichen Middleware-Komponenten für dynamisch verteilt ausgeführte Prozesse integriert:

- **Migrationsdienst:** Als zentrale Erweiterung wurde der (weitestgehend) generische Migrationsdienst (vgl. Abschnitt 7.5) mit einer *Process-Management-as-a-Service*-Schnittstelle und einer automatischen Erzeugung von Basismigrationsmodellen für Standard-XPDL-Prozesse angebunden. Der Migrationsdienst wurde äquivalent zu den funktionalen Diensten als lokaler Dienst in die Ausführungsumgebung integriert, seine Funktionalität durch eine UUID gekennzeichnet und beim lokalen Verzeichnisdienst veröffentlicht. Die Migration erfolgt dementsprechend durch ein Auffinden aller temporär erreichbaren Migrationsdienste über das Dienstverzeichnis, der Auswahl einer nach den benutzerdefinierten Vorgaben geeigneten Zielausführungseinheit und deren Aufruf über die Dienstschnittstelle.

- **Komponente zur verteilten parallelen Ausführung:** Für DPDL-Prozesse wurde eine Erweiterung der bestehenden Ausführungs- und Migrationsmöglichkeiten um eine parallel verteilte Ausführung von Prozessen mit Kennzeichnung von Datenklassen und entsprechender Synchronisation der Daten paralleler Prozesspfade vorgenommen (vgl. Abschnitt 6.2.5 und [Ham09]).

- **Management-Schnittstelle:** Für die Umsetzung sowohl auf mobilen als auf stationären Endgeräten wurde eine vereinfachte WSDM-Schnittstelle (vgl. Abschnitt 7.3) zur Durchführung von *GetRessourceProperty*- und *UpdateRessourceProperty*-Operationen, den erweiterten Operationen zum Prozessmanagement und zum Abonnement von grundlegenden Ereignissen umgesetzt. Die resultierenden Dienste wurden äquivalent zu den funktionalen Diensten als lokale Dienste in die Ausführungsumgebung integriert, ihre Funktionalität jeweils durch eine UUID gekennzeichnet und beim lokalen Verzeichnisdienst veröffentlicht (vgl. [Wun09]). Die internen Schnittstellen werden darüber hinaus durch den oben genannten Migrationsdienst verwendet.

- **Managementdienst zum Offline-Management:** Aufbauend auf den öffentlich angebotenen Funktionalitäten der verwaltbaren Ressource

wurde das regel- und ereignisbasierte Management mit *Esper* (vgl. Abschnitt 7.4) als Komponenten der lokalen Middleware von DEMAC umgesetzt (vgl. [Str09]). Die EPL-basierten Management-Regeln werden zur Überwachung der migrierten Prozesse in die Migrationsdaten integriert. Bei einer Migration des Prozesses werden die Management-Regeln somit an die nächste Ausführungseinheit migriert und durch den jeweils lokalen Managementdienst für die Dauer der lokalen Ausführung überwacht bzw. ausgeführt. Die Implementierung der Prozess-Engine ist hiervon unberührt. Etwaige Teilereignisse der komplexen Ereignisverarbeitung werden bei eine Migration nicht berücksichtigt.

- **Interaktionsdienst:** Der Interaktionsdienst (vgl. Abschnitt 7.6) wurde als funktionaler lokaler Dienst in die Ausführungsumgebung integriert, seine Funktionalität durch eine UUID gekennzeichnet und beim lokalen Verzeichnisdienst veröffentlicht. Die abstrakte Beschreibung der Benutzungsschnittstelle kann dabei entweder als Referenz in Form einer URL (für XPDL-Prozesse) oder direkt als konstanter Wert einer Prozessvariablen (für DPDL-Prozesse) aus der Prozessbeschreibung referenziert werden [Vil08].

- **Prognosesystem:** Basierend auf dem bestehenden Kontextmanagementsystem und den Informationen des lokalen Verzeichnisdienstes werden durch *strukturierte Kontextdatenprognose* Daten für die Vorhersage von Dienstverfügbarkeiten erhoben, welche für eine Unterstützung der Migrationsentscheidungen genutzt werden (vgl. Abschnitt 7.7.2 und [Mei09]). Die Implementierung der Prozess-Engine ist hiervon unberührt.

Um der bisher auf DPDL-Prozesse beschränkten *DEMAC*-Middleware die (verteilte) Ausführung von Standard-XPDL-Prozessen zu ermöglichen, war im Rahmen der prototypischen Umsetzung eine fast vollständige Reimplementierung der Prozess-Engine notwendig. Da die zu verarbeitenden Standard-Prozessbeschreibungen keine der zuvor obligatorischen Elemente zur Spezifikation von Verteilungsinformationen aufweisen, mussten die Komponenten zur Interpretation der Prozessbeschreibung (*Parser*), das interne Objektmodell und die Komponente zur Steuerung des Kontrollflusses stark modifiziert werden. Da die *DEMAC*-Middleware vornehmlich für den Einsatz auf leistungsbeschränkten mobilen Endgeräten konzipiert wurde, verfügt die Prozess-Engine zudem nicht über eine Datenbank zur Verwaltung von Prozessinformationen. Es musste daher zur Kapselung der Prozess-Engine eine Sensor-/Effektor-Schnittstelle integriert werden, welche die notwendigen Informationen und Steuerungsfunktionalitäten für die Repräsentation des Prozessmanagementsystems als verwaltbare Ressource bereitstellt. Die Einbindung der Dienste in die bestehende dienstorientierte Architektur sowie deren Nutzung durch die höherwertigen Komponenten stellte über die Registrierung der Dienste hinaus keinen besonderen Aufwand dar.

Einschätzung des Aufwands für einmalige Anpassungen

Mit Ausnahme des Migrationsdienstes wurde jede der oben genannten zusätzlichen Middleware-Komponenten im Rahmen der praktischen Implementierung von jeweils insgesamt auf sechs Monate ausgelegten Diplomarbeiten realisiert (vgl. [Vil08, Ham09, Mei09, Wun09, Str09]). Dabei entfiel jeweils nur ein geringer Anteil auf die Integration der jeweils entwickelten prototypischen Komponenten (maximal 1 Personenmonat, d. h. 160 Personenstunden). Für die Integration musste in allen Fällen nur die bestehende Prozess-Engine angepasst werden. Eine Anpassung der bisher verwendeten Modellierungswerkzeuge, der funktionalen Dienste, des Kontextmanagementsystems oder des Verzeichnisdienstes war nicht erforderlich. Der Aufwand für die Reimplementierung der Prozess-Engine zur Realisierung der Prozessmigration ist als am höchsten einzuschätzen, wobei dieser in diesem Fall zum Großteil der Ersetzung der Prozessbeschreibungssprache DPDL durch XPDL und nicht der Integration der hier genannten Middleware-Komponenten zugerechnet werden muss.

8.2.2 Sliver

Die an der Washington University in St.Louis entwickelte Middleware *Sliver* integriert eine Prozess-Engine zur Ausführung von WS-BPEL-Prozessen sowie ein Kommunikationssystem zum Aufrufen und Anbieten von Diensten über SOAP [HHGR06, HGR07]. Die Architektur der Middleware ist dabei speziell auf die Anforderungen mobiler Endgeräte ausgelegt, wobei jeweils ein einzelnes (mobiles) Endgerät als zentrale Ausführungseinheit für WS-BPEL-Prozesse dient. Die blockstrukturierten Sprachelemente von WS-BPEL werden dabei interpretiert, hierarchisch geschachtelt aufgerufen und nach Prüfung der für das jeweilige Element definierten Vorbedingungen ausgeführt. Für die Ausführung von atomaren Aktivitäten zur Interaktion mit Diensten (*invoke*, *receive* und *reply*) kann über die Nutzung einer Erweiterung des *PartnerLink*-Elements von WS-BPEL eine dynamische Bindung von Prozessteilnehmern erreicht werden (vgl. [HGR07] zu Details). Ein dynamisches Auffinden von Diensten ist über die Nutzung von Bluetooth durch die *Bluetooth-Service-Discovery*-Funktionalität realisiert [HGR07]. Eine Verteilung des Prozesses im Sinne einer Partitionierung über mehrere Prozess-Engines ist durch die *Sliver*-Middleware jedoch bisher nicht vorgesehen.

Gegenüber der bestehenden Version der *Sliver*-Middleware (vgl. [HHGR06, HGR07]) wurden im Rahmen dieser Arbeit die folgenden zusätzlichen Middleware-Komponenten für dynamisch verteilt ausgeführte Prozesse integriert:

- **Migrationsdienst:** Für die dynamische Verteilung von laufenden WS-BPEL-Prozessinstanzen wurde ein Migrationsdienst implementiert, welcher zum einen eine spontane Migration der bisher auf der *Sliver*-Prozess-Engine zentral ausgeführten Prozesse anhand von benutzer-

definierten oder lokalen Richtlinien ermöglicht und zum anderen laufende Prozessinstanzen entgegennehmen und weiterführen kann. Es wird zudem die Erzeugung von Basismigrationsmodellen als Ausgangspunkt für die Modellierung benutzerdefinierter Rahmenbedingungen von WS-BPEL-Prozessen unterstützt. Die Funktionalität des *Process-Management-as-a-Service* wird analog zu den durch die WS-BPEL-Prozesse repräsentierten Funktionalitäten als Web Service angeboten (vgl. [ZKML10]).

- **Management-Schnittstelle:** Für die Beobachtung der Prozessausführung durch den Migrationsdienst wurde die Prozess-Engine so erweitert, dass bei jeder Ausführung eines WS-BPEL-Sprachelements über die dabei ausgelösten Zustandsveränderungen Ereignisse ausgelöst werden, welche vom Migrationsdienst verarbeitet werden. Zudem wurden vereinfachte Operationen zum Setzen der Zustände von Aktivitäten und zum Setzen von Variablenwerten integriert. Die Ereignisse und Operationen werden hierbei durch den oben genannten Migrationsdienst im Rahmen einer internen Kommunikation verwendet.

Für die Installation von WS-BPEL-Prozessen wird bei der Verwendung der *Sliver*-Prozess-Engine zunächst zusätzlich zum WS-BPEL-Prozessmodell ein Deployment-Deskriptor benötigt, welcher die WSDL-Beschreibungen der Dienste und die spezifizierten *Partner Link Types* enthält. Um den normalen Ablauf des Deployments von Prozessen auf der Prozess-Engine nicht verändern zu müssen, werden diese Daten für die Migration laufender Prozessinstanzen in die Migrationsdaten integriert. Da WSDL-Beschreibungen und *Partner Link Types* ohnehin schon in einer einheitlichen XML-Repräsentation vorliegen, kann der Anteil der plattformspezifischen Beschreibungen in den Migrationsdaten in diesem Fall weitestgehend begrenzt werden. Als Konsequenz kann die bereits vorhandene Operation zum Deployment von WS-BPEL-Prozessen auch für die Migration von laufenden Prozessinstanzen verwendet werden. Für die initiale Erzeugung von Migrationsdaten werden bereits beim Einlesen einer Prozessbeschreibung die Namen aller Aktivitäten und Variablen an den Migrationsdienst übergeben, so dass hieraus automatisch eine Vorlage für die Definition benutzerdefinierter Rahmenbedingungen abgeleitet werden kann (*Basismigrationsmodell*). Für die über den Migrationsdienst eingebrachten Prozesse wird nach dem Deployment ergänzend verhindert, dass durch eingehende Nachrichten eine neue Instanz des Prozesses erzeugt werden kann, da in diesem Fall nur eine bereits bestehende Instanz weitergeführt werden soll.

Da die *Sliver*-Prozess-Engine ebenso wie die *DEMAC*-Middleware nicht über eine zentrale Datenbank verfügt, in der die Zustände aller Aktivitäten und Prozessvariablen bei Bedarf abzurufen sind, werden alle auf der lokalen Prozess-Engine ausgeführten Prozessinstanzen über ein integriertes Ereignissystem durch den Migrationsdienst überwacht und deren aktueller Zustand jeweils in den Migrationsdaten widergespiegelt. Hierfür wurden zusätzliche Sensoren und Effektoren direkt in die hierarchische Abbildung der WS-BPEL-

Sprachelemente eingefügt, welche jeweils über eine eigene Operation zu deren individueller Ausführung verfügen. Da diese lediglich bei der Feststellung von Migrationsbedarf eine Reaktion verursachen, wird die Ausführung zentral ausgeführter Prozesse hierbei nicht wesentlich beeinflusst (vgl. Abschnitt 8.5.3).

Einschätzung des Aufwands für einmalige Anpassungen

Die Anpassung der Prozess-Engine *Sliver* erfolgte im Rahmen eines Beitrags für ein Teilarbeitspaket des europäischen Exzellenznetzwerks S-Cube [1] durch eine als studentische Hilfskraft beschäftigte Person. Die insgesamt in Rechnung gestellte Zeit für die prototypische Anpassung der Prozess-Engine betrug 150 Personenstunden, wobei jedoch bereits die Bereitstellung von Testfällen für die Auswertung zur Migrierbarkeit der Sprachkonstrukte von WS-BPEL eingerechnet ist (vgl. Abschnitt 8.3.2).

8.2.3 Activiti

Die Prozess-Engine von *Activiti* ist Teil der quelloffenen *Activiti-BPM-Plattform* zur graphischen Modellierung, technischen Umsetzung und computergestützten Ausführung von BPMN-Prozessen der Version 2.0 [B+11]. Das Ausführungssystem repräsentiert ein typisches Workflow-Managementsystem zur prozessgesteuerten Integration von Anwendern und automatisiert ausführbaren Aktionen, welche unter anderem durch Web Services, Java-Anwendungen oder (Java-)Skripte implementiert werden können. Die Prozessteilnehmer werden in einem zentralen Organisationsmodell verwaltet und können bei vorliegender Berechtigung die manuell bzw. interaktiv auszuführenden Aktivitäten aus einer ebenfalls zentralen Arbeitsliste (*Worklist*) zur Bearbeitung auswählen. Die von *Activiti* verarbeiteten Prozesse bestehen dementsprechend aus der XML-Repräsentation der verwendeten Standard-BPMN-Sprachkonstrukte und einer Menge von plattformabhängigen Erweiterungen zur Referenzierung von Entitäten des Organisationsmodells, zum Verweis auf Benutzungsschnittstellen und zum Aufruf von Methoden in Java-Klassen, welche mit dem Prozess bereitgestellt werden oder bereits in der Ausführungsumgebung vorhanden sind. Zusammengehörige Elemente werden zum Deployment in einem Archiv an die Prozess-Engine übergeben und werden anschließend in einer Datenbank gespeichert.

Die *Activiti-BPM-Plattform* ist auf einen stationären Betrieb ausgelegt, wobei die Prozesse von einer Serverkomponente verwaltet und ausgeführt werden und die Interaktion mit dem Benutzer über eine lokale Client-Anwendung oder eine Web-Anwendung stattfinden kann. Zur Integration mobiler Endgeräte existiert zudem eine Client-Komponente für Android- und iOS-Betriebssysteme, welche jedoch nur den Zugriff auf die zentrale Serverkomponente unter Darstellung einer für das Mobilgerät angemessenen Benutzero-

[1] The Software Services and Systems Network (S-Cube): European Community's Seventh Framework Programme FP7/2007-2013, http://www.s-cube-network.eu

berfläche anpasst. Eine Verteilung von Prozessen im Sinn einer Partitionierung über mehrere Prozess-Engines ist durch *Activiti* bisher nicht vorgesehen. Gegenüber der bestehenden Version der Prozess-Engine von *Activiti* (vgl. [B+11]) wurde im Rahmen dieser Arbeit die folgende zusätzliche Middleware-Komponente für dynamisch verteilt ausgeführte Prozesse integriert:

- **Migrationsdienst:** Für die Migration von Prozessinstanzen wurde ein Migrationsmanager implementiert, welcher die Migration eines aktuell laufenden BPMN-Prozesses auf der Basis des Wertes einer Prozessvariablen vornimmt. Laufende Prozessinstanzen aus BPMN-Prozessmodellen und Migrationsdaten können von anderen Activiti-Ausführungseinheiten entgegengenommen und fortgeführt werden.

Da die Prozess-Engine von *Activiti* sowohl die Prozessmodelle als auch die Zustände von Aktivitäten und Prozessvariablen in einer Datenbank speichert, kann eine Erhebung des aktuellen Zustands sowie dessen Veränderung durch von der Prozess-Engine weitestgehend entkoppelte Datenbankoperationen durchgeführt werden. Soll eine Anpassung erfolgen, muss zuvor die Ausführung der betreffenden Prozessinstanz kontrolliert angehalten werden und nach Abschluss der Anpassungsmaßnahme die Ausführung fortgesetzt werden, wobei in beiden Fällen sicherzustellen ist, dass sowohl der Zustand der Datenbank als auch der Zustand der von der Prozess-Engine aus der Datenbank geladenen Daten aktuell ist. Im einfachsten Fall wird vor Beginn jeder (atomaren) Aktivität durch die Auswertung einer Variablen geprüft, ob eine Migration stattfinden soll. Falls dies nicht der Fall ist, kann die lokale Prozessausführung ohne Verzögerung fortgesetzt werden. Der Anpassungsaufwand für diese Vorgehensweise ist dementsprechend auf das Prüfen, das kontrollierte Anhalten und die kontrollierte Wiederaufnahme der Ausführung einzelner Prozessinstanzen begrenzt. Die bestehende Implementierung der *Activiti*-Prozess-Engine musste hierfür daher nicht wesentlich verändert werden. Da bei einer Migration einer Prozessinstanz auch alle (plattformabhängigen) Elemente des Archivs weitergegeben werden müssen, ist zur Zeit jedoch nur eine Migration von Prozessen zwischen zwei gleichartigen *Activiti*-Prozess-Engines möglich.

Einschätzung des Aufwands für einmalige Anpassungen

Die Anpassung der Prozess-Engine erfolgte im Rahmen der Lehrveranstaltung *Business Process Management* für die Master-Studiengänge Informatik und Wirtschaftsinformatik am Fachbereich Informatik der Universität Hamburg. Für die Konzeption der Anpassung sowie deren Durchführung stand hierbei über einen Zeitraum von 14 Wochen ein Zeitrahmen von insgesamt vier Semesterwochenstunden (d. h. drei Zeitstunden pro Woche) für ein Bearbeitungsteam aus vier Personen zur Verfügung. Dies entspricht – unter Vernachlässigung der Zeit für Vor- und Nachbereitung der Veranstaltung – einer Zeitdauer von 168 Personenstunden, welche exklusiv für die Anpassung

genutzt werden konnte. Als Teilergebnis der Projektarbeit war eine Migration von sequentiellen BPMN-Prozessen über zwei gleichartige *Activiti-BPM-Plattformen* möglich.

8.2.4 Zusammenfassung der Ergebnisse

Die in dieser Arbeit vorgestellten Konzepte zur Flexibilisierung verteilter Prozessausführung beruhen auf dem Gedanken, nicht die verteilten Anwendungen in Form der Prozesse anzupassen, sondern die Middleware (d. h. das Prozessmanagementsystem) geeignet zu erweitern, um die für eine dynamische Verteilung von Prozessen benötigten Funktionalitäten angemessen bereitzustellen. Die Untersuchung von drei unterschiedlichen Prozessmanagementsystemen hat gezeigt, dass – unter der Voraussetzung der Verfügbarkeit des Quellcodes – die Möglichkeit zu einer solchen Erweiterung grundsätzlich besteht. Die Komplexität der Anpassung und die hierfür notwendige Vorgehensweisen sind dabei implementierungsabhängig. Mit der Implementierung von unmittelbaren Sensoren und Effektoren, einer ereignisbasierten Schnittstelle und einer datenbankbasierten Integration wurden (z. T. auch in Kombination) exemplarisch verschiedene Verfahren für die Erweiterung der Prozessmanagementsysteme eingesetzt. Für eine jeweils prototypische Anpassung der Prozess-Engines sind dabei Entwicklungsaufwände mit einer oberen Grenze von 168 Personenstunden angefallen, wobei die entwickelnden Personen keine speziellen Vorkenntnisse im Bereich der Implementierung von Prozessmanagementsystemen aufwiesen.

Die neue Möglichkeit zur dynamisch verteilten Ausführung von Prozessen legt jeweils die bereits bestehende Funktionalität der Prozess-Engines zur zentralen (nicht verteilten) Ausführung zugrunde, verändert diese jedoch nicht nachhaltig, so dass beide Varianten der Ausführung nebeneinander und weitestgehend voneinander unabhängig angeboten werden können. Die zentrale Ausführung von Prozessen wird durch die Möglichkeit zur Verteilung nicht messbar beeinträchtigt (vgl. auch Abschnitt 8.5). Als Einschränkung der hier untersuchten Prozessmanagementsysteme kann jedoch festgehalten werden, dass aufgrund von plattformspezifischen Besonderheiten eine dynamische Verteilung von Prozessen bislang nur zwischen gleichartigen Prozessmanagementsystemen möglich ist. Eine Untersuchung der Migrierbarkeit von Prozessinstanzen zwischen Prozessmanagementsystemen unterschiedlicher Hersteller (mit Unterstützung einer gemeinsamen (standardisierten) Prozessbeschreibungssprache) steht noch aus.

8.3 Untersuchung der Migrierbarkeit von Prozessinstanzen

Da für die dynamische Verteilung der Prozessausführung und die korrekte Ausführung des Prozesses bei allen teilnehmenden Ausführungseinheiten zwingend eine semantisch äquivalente Interpretation und Verarbeitung der Ausführungsanweisungen in Form der Prozessmodelle erforderlich ist, ist für

die dynamische Verteilung von Prozessen insbesondere die Nutzung von standardisierten Prozessbeschreibungssprachen vorteilhaft (vgl. Abschnitt 5.2.4). Das in Abschnitt 6.2 vorgestellte Konzept zur Migration laufender Prozessinstanzen verwendet daher das Prinzip der Abstraktion, um die Migration möglichst aller Prozessinstanzen zu unterstützen, deren Prozessmodell sich auf ein minimales gemeinsames Prozessmetamodell abbilden lässt. Dabei wird angenommen, dass die Prozesse derartig abbildbarer Prozessbeschreibungssprachen durch nicht-invasive Migrationsdaten erweitert werden können, so dass in Folge das Prozessmodell für eine individuelle dynamische Verteilung seiner Prozessinstanzen nicht verändert werden muss. Eine dynamische Verteilung dieser Prozesse kann daher erfolgen, ohne den Standard der verwendeten Prozessbeschreibungssprache oder die Semantik des durch den Prozess definierten Anwendungsfalls zu verletzen bzw. zu verändern. Ein weiterer Vorteil dieser zusätzlichen Abstraktionsebene liegt in einer möglichst langfristigen Unterstützung der verteilten Prozessausführung über die Entwicklung bzw. Weiterentwicklung einzelner aktueller Prozessbeschreibungssprachen hinaus.

In diesem Abschnitt werden mit XPDL (vgl. Abschnitt 8.3.1), WS-BPEL (vgl. Abschnitt 8.3.2) und BPMN (vgl. Abschnitt 8.3.3) exemplarisch drei aktuell praktisch relevante Prozessbeschreibungssprachen in Hinblick auf die Migrierbarkeit der durch die Kombination der durch sie definierten Kontrollflusskonstrukte modellierten Prozesse untersucht. Dabei steht insbesondere im Vordergrund, welcher *Anpassungsspielraum* bei der Ausführung von Prozessinstanzen dieser Prozessbeschreibungssprachen erreicht werden kann und inwieweit eine Einschränkung der Migration und somit des Anpassungsspielraums in Bezug auf die Verteilungskonfiguration notwendig werden kann. Für eine Bewertung der Flexibilisierung durch die vorgeschlagenen Maßnahmen erfolgt zudem jeweils eine vergleichende Betrachtung mit einer Verteilung durch physische Partitionierung der Prozesse.

Für die Untersuchung der allgemeinen Migrierbarkeit von Prozessen der genannten drei Prozessbeschreibungssprachen wurde als Testumgebung die ergänzende Implementierung der in Abschnitt 8.2 beschriebenen Ausführungseinheiten zugrunde gelegt, wobei die Ergebnisse der folgenden Untersuchung auf der Grundlage von Quell- und Zielausführungssystemen jeweils gleicher Art entstanden sind. Dabei wurden als Testfälle Prozesse modelliert, welche jeweils eine Auswahl der in der Prozessbeschreibungssprache verfügbaren und von der Ausführungseinheit unterstützten Kontrollflusskonstrukte nacheinander sequentiell und teilweise auch parallel verkettet. Als aufzurufende Ressourcen des Prozesses wurden Dienste zur Durchführung einfacher Rechenoperationen, zur Ausgabe von Werten und zur Benutzerinteraktion verwendet. Der oben genannte Vergleich mit der physischen Partitionierung erfolgt auf Basis einer theoretischen Betrachtung.

8.3.1 XPDL-Prozesse

Die XML-basierte Prozessbeschreibungssprache *XPDL* (*XML Process Definition Language*) wurde ursprünglich durch die *Workflow Management Coalition* (*WfMC*) als abstraktes Metaformat entwickelt, um den Austausch von Prozessmodellen zwischen verschiedenen Prozessmanagementsystemen zu unterstützen (vgl. auch Abschnitt 5.2.4). Da die in Abschnitt 8.2.1 vorgestellte Prozess-Engine der *DEMAC*-Middleware ausführbare XPDL-Prozesse der Version 1.0 [NM02] verarbeitet, wird im Folgenden insbesondere auf die Migrierbarkeit der durch diese Version modellierten Prozesse eingegangen. Die später entwickelten Versionen XPDL 2.0 [NMS05] und XPDL 2.1 [WfM08] enthalten im Wesentlichen Erweiterungen, um die bis zu diesem Zeitpunkt nur graphisch vorliegenden Prozesse der *BPMN*-Spezifikation im XML-Format abzubilden. Da BPMN seit der Version 2.0 über eine eigene XML-Repräsentation verfügt, wird diese anstelle der neueren Versionen von XPDL in Abschnitt 8.3.3 genauer betrachtet und an dieser Stelle die relativ einfach gehaltenen Version 1.0 untersucht.

Im Allgemeinen besitzt die Prozessbeschreibungssprache XPDL eine Graphstruktur und erlaubt den Verweis auf beliebige Softwareanwendungen, Maschinen und menschliche Prozessteilnehmer in Form von atomaren Aktivitäten (*Activities*). Für den Aufbau komplexerer Kontrollflussstrukturen können diese durch Transitionen (*Transitions*) miteinander verbunden und zu wiederverwendbaren Blöcken (*Activity Sets*) und schließlich zu Prozessen (*Workflow Processes*) komponiert werden. Alternative und parallele Pfade werden durch spezielle Kombinationen von Aktivitäten mit Transitionen und Transitionsbedingungen in Form von *OR*-, *XOR*- oder *AND*-Splits zur Verzweigung sowie entsprechende Join-Konstrukte zur Zusammenführung bzw. Synchronisation des Kontrollflusses ausgedrückt. Variablen (*Data Fields*), Datentypen (*Data Types*) und Personen bzw. Rollen (*Participants*) werden als aus der Sicht des Prozesses global verfügbare Elemente in einem Paket (*Package*) deklariert, welches verschiedene Prozesse vom Typ *Workflow Process* beinhalten kann. Dabei können die Prozesse in einem *Package* innerhalb ihres Kontrollflusses miteinander in einer aufrufenden Verbindung stehen (z. B. in einer Prozess-Subprozess-Beziehung) oder voneinander unabhängig sein.

Um eine Abbildung auf das minimale gemeinsame Prozessmetamodell des Migrationsmodells vornehmen (vgl. Abschnitt 6.2.2) und somit eine individuelle XPDL-Prozessinstanz migrieren zu können, ist es vorteilhaft, wenn ein *Package* jeweils nur einen einzelnen unabhängigen Prozess (ggf. mit weiteren abhängigen Subprozessen) beinhaltet. Ansonsten würden die Prozessmodelle der anderen im *Package* definierten Prozesse später bei jeder Migration einer Prozessinstanz einer dieser Prozesse unnötigerweise mit auf die ausgewählten Zielausführungseinheiten transferiert werden. Optional kann daher bei Vorliegen einer solchen Situation der (unabhängige) zu migrierende Prozess aus dem Container ausgeschnitten und die von ihm genutzten globalen Elemente repliziert werden [ZK07, ZHKK10]. Die Zusammenfassung von Prozessen, welcher

miteinander in aufrufender Verbindung stehen, ist hingegen für die dynamische Verteilung von Prozessinstanzen vorteilhaft, weil durch die Migration des gesamten *Packages* automatisch auch die Modellinformationen für die Subprozesse bei jeder Ausführungseinheit vorliegen. Ein Subprozess kann somit – bei Vorliegen aller sonstigen Rahmenbedingungen – auch direkt bei der jeweils aktuellen Ausführungseinheit instantiiert und ausgeführt werden kann.

Die Abbildung der kontrollflussbezogenen Sprachelemente innerhalb des *Workflow-Process*-Elements sowie das Ergebnis der Untersuchung der Migrierbarkeit von XPDL-Prozessinstanzen im Vergleich mit der physischen Partitionierung von Prozessen ist Tabelle 8.1 zu entnehmen und wird im Folgenden erläutert.

Wichtiger Betrachtungsgegenstand sind die atomaren Aktivitäten der XPDL-Version 1.0, welche entweder zu einem einseitigen (*One-Way*) oder einem synchronen (*Request-Response*) Aufruf einer Ressource von Seiten des Prozessausführungssystems führen können (vgl. Abschnitt 2.7.3). Die atomaren Aktivitäten sind daher in Hinblick auf die Kommunikation jeweils voneinander unabhängig, so dass eine Migration vor oder nach dem Aufruf einer Ressource vorgenommen werden kann und nicht aufgrund von asynchronen Antwortnachrichten an nachfolgende Aktivitäten eingeschränkt werden muss. Eine Migration während der Ausführung atomarer Aktivitäten ist nach dem hier vorgestellten Modell nicht möglich, da das Prozessausführungssystem in der Regel keine direkte Kontrolle über die Ressource und die Bearbeitung der mit der Aktivität verbundenen fachlichen Aufgabe besitzt. Dies gilt nicht für den Fall der *Block Activity*, da diese einen *Activity Set* referenziert und daher durch den hierdurch definierten Kontrollfluss ersetzt werden kann. Die *Block Activity* wird daher wie ein Behälter betrachtet, welcher in den Zustand *executing* versetzt wird, während die (atomaren oder strukturierten) Aktivitäten des dazugehörigen *Activity Set* weiter ausgeführt werden (vgl. Abschnitt 6.2.4). Auf diese Weise kann die Prozessinstanz entsprechend der vorgestellten Strategie auch innerhalb einer *Block Activity* migriert werden [ZHKK10].

Weitere zu einem Verbund zusammengefasste Kontrollflussstrukturen mit einem vordefinierten Verhalten, wie zum Beispiel bedingte Verzweigungen oder Schleifen, existieren in XPDL 1.0 – wie zum Beispiel bei WS-BPEL – nicht in Form von explizit definierten Elementen, sondern werden implizit durch eine entsprechende Modellierung der Graphstruktur festgelegt. Die hierfür verwendbaren Bedingungskonstrukte basieren auf logischen Operationen auf Prozessvariablen, deren Werte im Fall einer Migration automatisch mit dem Prozess weitergegeben werden. Die Auswertung eines Bedingungskonstrukts erfolgt als atomare Aktivität. Wird zum Beispiel durch eine Bedingung ausgedrückt, dass eine Schleife aus einer sequentiell angeordneten Menge von Aktivitäten ausgeführt werden soll, bis eine Variable V den Wert w besitzt, so ist V als *Data Field* des Prozesses definiert und der aktuelle Wert w ist in den Migrationsdaten vermerkt. Folglich liegen die für die Aus- bzw. Fortführung des Kontrollflusses erforderlichen Informationen allen (potentiellen) Ausführungseinheiten vor, so dass eine Migration auch innerhalb der

Schleife und somit zwischen einzelnen Schleifendurchläufen möglich ist. Eine solche Verteilung von unterschiedlichen Schleifendurchläufen gestaltet sich im Gegensatz dazu bei einer physischen Partitionierung des Prozesses schwierig, da auf der Ebene des Prozessmodells zunächst unter Umständen gar keine Differenzierung unterschiedlicher Schleifendurchläufe vorgesehen ist. Es müsste daher für eine gezielte Zuweisung einzelner Schleifendurchläufe an verschiedene Ausführungseinheiten eine erhebliche Änderung am Prozessmodell vorgenommen werden, z. B. um die ursprüngliche Schleife in zwei (unterschiedliche) Schleifen zu zerlegen, welche nun wiederum den Ausführungseinheiten zugeordnet werden können [ZHKK10].

Im Fall einer Verzweigung des Kontrollflusses wird das hierfür spezifizierte Bedingungskonstrukt als atomare Aktivität ausgewertet und die Menge der entsprechend des Auswertungsergebnisses auszuführenden Pfade durch das Setzen der Startaktivität(en) determiniert. Die nicht auszuführenden Pfade werden im Rahmen der *Dead Path Elimination* in den Zustand *skipped* versetzt (vgl. Abschnitt 2.7.2) und müssen daher nicht weiter betrachtet werden, da die entsprechenden Zustandsinformationen automatisch auch allen nachfolgend teilnehmenden Ausführungseinheiten vorliegen. Im Fall einer physischen Partitionierung von Prozessen werden die Partitionen und die Verteilung der Prozesspfade in vielen Fällen bereits zur Entwicklungszeit bzw. spätestens beim Aufruf des Prozesses vorgenommen. Es werden daher hierbei Prozesspfade verteilt, welche zur Laufzeit unter Umständen gar nicht ausgeführt werden müssen. Für die Zusammenführung oder Synchronisation von Kontrollflusspfaden (OR-Split bzw. -Join) ist in Folge wiederum eine Kommunikation zwischen den Partitionen des Prozesses notwendig, um zu vermitteln, wie viele ausgehende Pfade an der Verzweigung aktiviert wurden und wie viele eingehende Pfade an einer Zusammenführung bzw. einem Synchronisationspunkt entsprechend erwartet werden müssen [ZHKK10].

Sofern nur ein Pfad einer Verzweigung ausgeführt wird (XOR-Split), ist keine Zusammenführung bzw. Synchronisation des Kontrollflusses notwendig und die Migrierbarkeit der Prozessinstanz entspricht der einer sequentiellen Ausführung. Da die Split- und Join-Elemente analog zu atomaren Aktivitäten in den Kontrollfluss eingebunden sind, ist zudem eine Zuordnung dieser Elemente zu bestimmten Ausführungseinheiten möglich. Dies ermöglicht auch im Fall einer verteilten parallelen Ausführung die (statische oder dynamische) Festlegung eines Join-Elements als „Treffpunkt“ für die Synchronisation paralleler Kontrollflusspfade (vgl. Abschnitt 6.2.5). Eine solche Zuordnung stellt zwar eine Einschränkung der Flexibilität bei der Migration von Prozessinstanzen dar, ist jedoch auch bei der physischen Partitionierung von Prozessmodellen stets notwendig. Gegenüber der physischen Partitionierung stellt die Replikation des Prozesses hierbei jedoch einen Vorteil dar, da hierdurch Datenabhängigkeiten zu parallel ausgeführten Pfaden anderer Ausführungseinheiten zur Laufzeit erkannt werden können. Da nur zur Laufzeit tatsächlich auftretende Abhängigkeitskonflikte behandelt werden müssen,

	XPDL-Elemente	Migration	Physische Partitionierung
Atomare Aktivitäten	*Activity*	vor oder nach der Ausführung der Aktivität	vor oder nach der Aktivität
Strukturierte Aktivitäten	*Block Activity* mit Referenz auf *Activity Set*	vor, nach und innerhalb der Ausführung der Blockaktivität	vor und nach der Aktivität, innerhalb der Aktivität durch Auflösung des Blocks
Variablen	*Data Field*	Weitergabe (Sequenz) oder Replikation (Parallelität)	Partitionierung bzw. Replikation der für die Partition relevanten Variablen
Transitionsbedingungen	*Condition*	vor oder nach der Auswertung der Bedingung	vor oder nach der Bedingung
Verzweigungselemente	*Transition Restriction* (Split, Join)	Zusammenfassung mit der Auswertung der dazugehörigen Transitionsbedingungen als atomare Aktivität	Zusammenfassung mit dazugehörigen Transitionsbedingungen als atomare Aktivität
Impliziter Kontrollfluss	Sequenz	vor, nach und innerhalb der Ausführung	vor, nach und innerhalb der Ausführung
	Verzweigung (*XOR*)	entspricht Sequenz, nur tatsächlich ausgeführter Pfad wird ggf. verteilt	entspricht Sequenz, alle Pfade werden ggf. vorverteilt
	Verteilt parallele Ausführung (*AND*)	Replikation des Prozesses und Synchronisation bei zur Laufzeit identifizierten Abhängigkeitskonflikten und bei der Zusammenführung (statischer oder dynamischer „Treffpunkt" erforderlich)	Synchronisation bei zur Entwicklungszeit identifizierten potentiellen Datenabhängigkeiten und bei der Zusammenführung (statischer „Treffpunkt" erforderlich)
	Verzweigung (*OR*)	entspricht Sequenz oder verteilt paralleler Ausführung, nur tatsächlich ausgeführter Pfade werden ggf. verteilt, Dead Path Elimination erfolgt automatisch	entspricht Sequenz oder verteilt paralleler Ausführung, alle Pfade werden ggf. vorverteilt, Dead Path Elimination erfordert ggf. zusätzliche Kommunikation
	Schleifen	vor, nach und innerhalb der Ausführung der Schleife	vor und nach der Schleife, innerhalb der Schleife durch Auflösung der Schleife oder durch zusätzlichen Koordinationsmechanismus
Subprozesse	*Subflow*	Subprozesse werden automatisch im *Package* mit transportiert	Subprozesse werden vorverteilt
Berücksichtigung von Verteilungsinformationen im Prozessmodell	*Participant*	ja (Auswertung zur Laufzeit möglich)	ja (Auswertung vor der Ausführung und Zuweisung der Partitionen möglich)
Festlegung der Verteilung zur Entwicklungszeit	-	ja (durch benutzerdefinierte Restriktionen)	ja (durch Zuweisung der Partitionen)
Festlegung der Verteilung zur Laufzeit	-	fortwährend	ggf. einmalig nach Instantiierung
Rebinding von Ressourcen	-	Abstraktionsgrad von XPDL erlaubt Rebinding	Abstraktionsgrad von XPDL erlaubt Rebinding
Vertraulichkeit von verteilten Prozesspartitionen	-	durch zusätzliche Verschlüsselung	automatisch erfüllt

Tabelle 8.1: Migrierbarkeit von XPDL-Prozessinstanzen (Version 1.0) und Vergleich mit dem allgemeinen Konzept der physischen Partitionierung (ZHKK10)

können durch die Migration von Prozessinstanzen unnötige Synchronisationsvorgänge vermieden werden [ZHKK10].

Da Verzweigungselemente (Split, Join) und Transitionsbedingungen in XPDL unabhängig voneinander modelliert werden, sollten bei einer Verteilung des Prozesses sowohl bei der Migration als auch bei der physischen Partitionierung des Prozesses die Auswertung der Transitionsbedingungen der ausgehenden Transitionen des Verzweigungselements nicht vom Verteilungsele-

ment getrennt werden. Bei einem XOR-Split könnte sonst bei der Verzweigung nicht mehr sichergestellt werden, dass nur ein einziger ausgehender Pfad aktiviert wird. Bei einem OR-Split und einer Verteilung der ausgehenden Pfade auf unterschiedliche Ausführungseinheiten würde zudem das Wissen über die Anzahl der aktivierten Pfade verloren gehen und somit bei der Zusammenführung der Pfade nicht mehr zur Verfügung stehen. Das gleiche gilt im Allgemeinen für eine physische Partitionierung.

Da XPDL in der Version 1.0 noch über eine sehr abstrakte Struktur für die Festlegung von Prozessteilnehmern und den Aufruf von Softwareanwendungen verfügt, ist abhängig von der jeweiligen Plattform und dem in der Prozessbeschreibung gewählten Abstraktionsgrad auch ein *Rebinding* von Ressourcen möglich. Die in XPDL-Prozessmodell möglicherweise angegebenen Ressourcen oder Rollenbeziehungen (*Participants*) können zudem für eine Verteilung des Prozesses eingesetzt werden, indem das *Participant*-Element, welches mit einer Aktivität in Verbindung steht, zur Laufzeit ausgelesen wird und – unter Berücksichtigung der benutzerdefinierten Rahmenbedingungen – eine Zuordnung der entsprechenden Prozesspartition zu einer geeigneten Ausführungseinheit vorgenommen wird.

Darüber hinaus ist eine Zuweisung von Ausführungseinheiten anhand von benutzerdefinierten Rahmenbedingungen möglich, welche die Verteilung der Prozessinstanzen zur Laufzeit bzw. der Zuordnung der Prozessfragmente vor Beginn der Ausführung steuert. Durch die Verwendung statischer oder rollenbasierter Zuordnungsstrategien bei der Migration von Prozessinstanzen ist dabei das Verhalten der physischen Partitionierung imitierbar, so dass auch eine komplette Verteilungsstrategie für die Migration von Prozessinstanzen zur Entwicklungszeit festgelegt werden kann. Eine fortwährende Umverteilung des Prozesses zur Laufzeit ist hingegen für eine physische Partitionierung nicht ohne weiteres möglich [ZHKK10].

8.3.2 WS-BPEL-Prozesse

Die XML-basierte Prozessbeschreibungssprache WS-BPEL (*Web Services Business Process Execution Language*) ist ein zur Zeit durch OASIS verwalteter Standard zur Komposition von Web Services [OAS07] (vgl. auch Abschnitt 5.2.4). Die in Abschnitt 8.2.2 vorgestellte Prozess-Engine für mobile Endgeräten unterstützt die ausführbare Repräsentation von den dem Standard vorausgehenden Prozessen der Version 1.1 unter dem Akronym *BPEL4WS* [And03]. Da die hier betrachteten (grundlegenden) Kontrollflusselemente größtenteils auch in der neueren Version 2.0 beibehalten wurden, gelten die Ergebnisse entsprechend für Prozesse beider Versionen [OAS07]. Im Einzelfall wird darauf hingewiesen, wenn es zu relevanten Abweichungen in Hinblick auf die Weiterentwicklung der Prozessbeschreibung kommt.

Im Gegensatz zu XPDL-Prozessen besitzen WS-BPEL-Prozesse auf oberster Ebene eine Blockstruktur, welche grundlegend aus zwei Arten von Aktivitäten zusammengesetzt werden kann: Atomare Aktivitäten stehen für die Interak-

tion mit Web Services (*Invoke*, *Receive* und *Reply*), zur Steuerung des Kontrollflusses (*Empty*, *Wait*, *Exit*, *Throw*, *Rethrow* und *Compensate*) sowie zur Manipulation und Überprüfung von Daten (*Assign*, *Validate*) zur Verfügung. Strukturierende Aktivitäten werden verwendet, um den Kontrollfluss zwischen atomaren Aktivitäten zu definieren und diese darüber miteinander in eine zeitlich-logische Beziehung zu setzen (*Sequence*, *Switch* bzw. *If-Elseif-Else*, *Pick*, *Flow*, *While*, *Repeat until*, *For each*). Auf Basis dieser grundlegenden Eigenschaften sind die genannten Aktivitäten dem in Abschnitt 6.2.2 vorgestellten Migrationsmodell zugeordnet. Die Tabellen 8.2 und 8.3 fassen das Ergebnis der Analyse dieser Kontrollflusskonstrukte in Hinblick auf eine Verteilung durch die Migrierbarkeit von Prozessinstanzen im Vergleich mit der physischen Partitionierung von Prozessmodellen zusammen.

Hinsichtlich der atomaren Aktivitäten zeigt sich, dass insbesondere die verschiedenen Möglichkeiten zur Interaktion mit Web Services für eine Verteilung des Prozesses (sowohl bei einer Migration als auch bei der physischen Partitionierung) Herausforderungen beinhalten. Die *Invoke*-Aktivität initiiert den Aufruf eines Web Services, welcher in der Prozessbeschreibung bzw. durch ergänzende WSDL-Dokumente abstrakt oder konkret durch die Angabe eines Zugriffsendpunktes definiert ist. Eine Migration kann dabei im Allgemeinen uneingeschränkt vor dem Aufruf des Web Services geschehen, was insbesondere vorteilhaft ist, wenn der angegebene Dienst von der aktuellen Ausführungseinheit nicht erreicht werden kann, oder die Migration kann nach dem Aufruf des Web Services erfolgen. Handelt es sich um einen Aufruf ohne Rückgabewert, kann die Migration prinzipiell sofort nach dem Aufruf erfolgen, da die Ausführung der Aktivität dann beendet ist. Handelt es sich um einen synchronen Aufruf mit Rückgabewert (*Request-Response*), so ist der Erhalt der Antwortnachricht ein Teil der atomaren Aktivität und kann somit nicht durch eine zwischenzeitliche Migration abgetrennt werden. Handelt es sich um einen asynchronen Aufruf, kann eine Migration nach dem Aufruf des Web Services nur erfolgen, wenn die im Kontrollfluss folgende Aktivität nicht die Verarbeitung der Antwortnachricht darstellt [ZKML10].

Die Aktivitäten *Receive* und *Reply* stellen weitere Elemente für einerseits eine asynchrone Kommunikation mit dem Aufrufer der durch den Prozess definierten Dienstleistung und andererseits die asynchrone Kommunikation des Prozesses mit anderen Web Services dar. Die initiale *Receive*-Aktivität zur Instantiierung des Prozesses stellt in Hinblick auf eine dynamische Verteilung von Prozessinstanzen kein Problem dar, da der durch den Prozess definierte Dienst zentral bereitgestellt wird und eine Prozessinstanz erst nach der Initialisierung migriert werden kann. Der Zugriffsendpunkt des Dienstes kann daher in einem Verzeichnis veröffentlicht und somit jederzeit von potentiellen Konsumenten aufgefunden und aufgerufen werden. Das Senden einer Antwortnachricht im Rahmen einer *Reply*-Aktivität kann realisiert werden, wenn die benötigten Informationen über den Empfänger der Nachricht (d. h. insbesondere dessen Zugriffsendpunkt) und etwaige benötige Korrelationsdaten vorhanden sind. Findet zwischen der (initialen) *Receive*-Aktivität und dem Auf-

ruf der *Reply*-Aktivität eine Migration statt, so müssen diese Informationen entweder ergänzend im Log gehalten werden, oder der Prozess muss zu der Ausführungseinheit zurück migriert werden, welche diese Informationen besitzt. Stellt die *Receive*-Aktivität jedoch die Verarbeitung einer Antwortnachricht einer vorausgehenden *Invoke*-Aktivität dar, so ist der Empfänger für die Antwortnachricht nach einer zwischenzeitlichen Migration unter Umständen nicht mehr aufzufinden. Es muss daher entweder eine Migration vollständig unterbleiben, oder der Prozess muss zum Empfang der Antwortnachricht an diejenige Ausführungseinheit zurück migriert werden, welche für die Ausführung der *Invoke*-Aktivität verantwortlich war. Letzteres setzt voraus, dass die Schnittstellen zum Empfang der Antwortnachricht während der Migration des Prozesses zu einer anderen Ausführungseinheit bestehen bleiben und die dort in der Zwischenzeit eingehenden Nachrichten zwischengespeichert werden können [ZKML10]. Betrachtet man im Vergleich dazu die Möglichkeiten einer physischen Partitionierung, so ist festzustellen, dass hierbei ähnliche Einschränkungen für die Verteilung von Prozessen bestehen (vgl. [KL06]). Insbesondere die Trennung eines *Invoke-Receive*-Aktivitätspaares ist hierbei nicht ohne weiteres durchführbar. Die Einschränkung der Flexibilität für eine dynamische Verteilung des Prozesses durch Migration ist vor diesem Hintergrund also hinnehmbar.

Da stets der gesamte Prozess mit allen Instanzdaten migriert wird, ist eine Zuweisung von Variablen durch die atomare *Assign*-Aktivität von einer Migration des Prozesses unabhängig. Dasselbe gilt für die atomaren Aktivitäten *Empty* und *Wait*, welche in Bezug auf die Verteilung des Prozesses kein interessantes Verhalten aufweisen. Die Ausführung der *Terminate-* [And03] bzw. *Exit*-Aktivität [OAS07] führt zu einer Beendigung der Prozessinstanz, was bei einer sequentiell verteilten Ausführung ohne weiteren Koordinationsaufwand möglich ist. Bei einer verteilten Ausführung von parallelen Prozesspfaden müssen bei der Ausführung der *Terminate-* bzw. *Exit*-Aktivität auf einem Pfad die anderen Pfade über den Abbruch des Prozesses informiert werden. Dies erfordert sowohl für eine verteilte Ausführung durch Migration von Prozessinstanzen als auch durch physische Partitionierung zusätzliche Koordinationsmechanismen [ZKML10].

Fehlerbehandlungsmaßnahmen sowie Maßnahmen zur Kompensation werden an Gültigkeitbereiche (*Scopes*) als Zusammenfassung einer Menge von Aktivitäten gekoppelt und in speziellen Behältern (*Fault Handler, Compensation Handler, Termination Handler*) spezifiziert, welche wiederum eine Reihe von atomaren Aktivitäten beinhalten und sich somit wie strukturierte Aktivitäten verhalten. Die Erzeugung von Fehlern (*Throw-* bzw. *Rethrow*-Aktivität) oder der Aufruf zur Kompensation (*Compensate*-Aktivität) aktiviert dabei – falls vorhanden – die Ausführung der entsprechenden Aktivitätsbereiche oder beendet die Ausführung des Prozesses mit einer Fehlermeldung. Die für eine spätere Kompensation benötigten Variablenwerte eines bestimmten Gültigkeitsbereichs können bei Bedarf im Log festgehalten werden. Da im Fall einer Migration von Prozessinstanzen immer die gesamte Struk-

	WS-BPEL-Elemente	Migration	Physische Partitionierung
Atomare Aktivitäten	*Invoke* (One-Way oder Request-Response)	vor oder nach der Ausführung der Aktivität	vor oder nach der Aktivität
	Receive (Instantiierung ohne Rückgabe)	nach der Ausführung der Aktivität	nach der Aktivität
	Receive-Reply-Aktivitätspaar (Instantiierung mit Rückgabe)	vor oder nach der Ausführung des Aktivitätspaares; zwischen *Receive* und *Reply* falls die Empfängerinformationen für das Senden der *Reply*-Nachricht bei der Migration inkludiert werden können oder falls *Reply* wieder der Ausführungseinheit zugewiesen wird, welche für *Receive* verantwortlich gewesen ist	vor oder nach der Ausführung des Aktivitätspaares, zwischen *Receive* und *Reply* falls *Reply* wieder der Ausführungseinheit zugewiesen wird, welche für *Receive* verantwortlich gewesen ist
	Invoke-Receive-Aktivitätspaar	vor oder nach der Ausführung des Aktivitätspaares; zwischen *Invoke* und *Receive* falls *Receive* wieder der Ausführungseinheit zugewiesen wird, welche für *Invoke* verantwortlich gewesen ist	vor oder nach der Ausführung des Aktivitätspaares, zwischen *Invoke* und *Receive* falls *Receive* wieder der Ausführungseinheit zugewiesen wird, welche für *Invoke* verantwortlich gewesen ist
	Assign	vor oder nach der Ausführung der Aktivität	vor oder nach der Aktivität
	Wait, *Empty*	vor oder nach der Ausführung der Aktivität	vor oder nach der Aktivität
	Throw, *Rethrow* (Referenz auf *Fault Handler*)	vor oder nach der Ausführung der Aktivität	vor oder nach der Aktivität
	Compensate, *Compensate Scope* (Referenz auf *Compensation Handler*)	vor oder nach der Ausführung der Aktivität	vor oder nach der Aktivität
	Terminate bzw. *Exit*	vor oder nach der Ausführung der Aktivität, ggf. Benachrichtigung verteilt ausgeführter paralleler Pfade	vor oder nach der Aktivität, ggf. Benachrichtigung verteilt ausgeführter paralleler Pfade
Strukturierte Aktivitäten	*Sequence*	vor, nach und innerhalb der Ausführung der Sequenz	vor, nach und innerhalb der Ausführung der Sequenz
	Switch bzw. *If... elseif... else*	vor, nach und innerhalb der Ausführung der Fallunterscheidung, Auswertung der Bedingung(en) als atomare Aktivität, nur tatsächlich ausgeführter Pfad wird ggf. verteilt	vor, nach und innerhalb der Fallunterscheidung, alle Pfade werden ggf. vorverteilt
	While... / Repeat... until	vor, nach und innerhalb der Ausführung der Schleife	vor und nach der Aktivität, innerhalb der Schleife durch Auflösung der Schleife
	For each	vor und nach der Schleife, innerhalb der Schleife nur bei sequentieller Ausführung und falls der Schleifenzähler bei der Migration inkludiert werden kann	vor und nach der Aktivität, innerhalb der Schleife durch Auflösung der Schleife
	Flow	Bei verteilt paralleler Ausführung Replikation des Prozesses und Synchronisation bei zur Laufzeit identifizierten Abhängigkeitskonflikten und bei der Zusammenführung (statischer oder dynamischer „Treffpunkt" erforderlich)	Bei verteilt paralleler Ausführung Synchronisation bei zur Entwicklungszeit identifizierten potentiellen Datenabhängigkeiten und bei der Zusammenführung (statischer „Treffpunkt" erforderlich)
	Pick	Abwarten des Ereigniseintritts bzw. dem Erreichen der Zeitbeschränkung als atomare Aktivität, Ausführung des Reaktionsteils kann verteilt erfolgen	Abwarten des Ereigniseintritts bzw. dem Erreichen der Zeitbeschränkung als atomare Aktivität, Ausführung des Reaktionsteils kann verteilt erfolgen, ggf. Auflösung des Gültigkeitsbereichs zur Verteilung von Ereignis-Reaktions-Paaren

Tabelle 8.2: Migrierbarkeit von WS-BPEL-Prozessen und Vergleich mit dem allgemeinen Konzept der physischen Partitionierung (Teil 1) (ZKML10)

tur des Prozesses mit allen Informationen über Fehler, Fehlerbehandlungsmaßnahmen und Anweisungen zur Kompensation bei der fehlerverursachenden Ausführungseinheit vorliegt, kann direkt nach dem Auftreten eines Fehlers mit der Fehlerbehandlung begonnen werden. Alternativ kann der Prozess während der Fehlerbehandlung migriert werden, z. B. um die Anweisungen innerhalb eines *Compensation Handler* durch die Ausführungseinheit vornehmen zu lassen, welche auch die zu kompensierenden Aktivitäten ausgeführt haben. Im Gegensatz dazu kann es – abhängig von der gewählten Zerteilung – bei einer physischen Partitionierung des Prozessmodells notwendig werden, die Gültigkeitsbereiche mit den definierten Fehlerbehandlungen für die Partitionierung aufzulösen oder zu verändern, was in der Regel einen hohen Modellierungsaufwand bedeutet [ZKML10].

Weitere strukturierende Aktivitäten, wie *Sequence*, *Switch* [And03] bzw. *If... Elseif... Else* [OAS07], *While* und *Repeat until* stellen wiederum Behälter für (atomare oder weiter strukturierte) Aktivitäten dar, so dass deren Ausführung nicht zwingend beendet werden muss, bevor eine Migration der Prozessinstanz möglich ist (vgl. Erklärungen zu entsprechenden XPDL-Konstrukten in Abschnitt 8.3.1) [ZKML10]. Während die Schleifen- bzw. Abbruchbedingung bei den vorgenannten Konstrukten zur Abbildung von Schleifen (*While* und *Repeat until*) bei jedem Schleifendurchlauf durch die aktuelle Ausführungseinheit ausgewertet werden kann, besitzt die strukturierte Aktivität *For Each* einen eigenen Durchlaufzähler. Dieser wird bei jedem Schleifendurchlauf erhöht bis die Gesamtzahl der Elemente erreicht wird, für welche die spezifizierten Anweisungen ausgeführt werden sollen. Der Durchlaufzähler gehört jedoch nicht zu den global definierten Prozessvariablen und wird daher nur im Speicher der Ausführungseinheit verwaltet. Damit der Durchlaufzähler nicht undefiniert ist, wenn eine Prozessinstanz während der Ausführung von *For Each* migriert wird, muss dieser entweder als lokale Prozessvariable in die Migrationsdaten aufgenommen werden oder individuell im Log spezifiziert werden. Zudem besteht bei WS-BPEL optional die Möglichkeit, die Ausführung der *For-Each*-Anweisungen für jedes der definierten Elemente einer Menge parallel durchführen zulassen [OAS07]. Eine Ausführung von mehreren parallel ausgeführten Instanzen der gleichen Aktivität kann bisher nicht durch die Migration von Prozessinstanzen unterstützt werden. Eine verteilt parallele Ausführung eines solchen Kontrollflusses durch physische Partitionierung des Prozessmodells kann jedoch ebenfalls nur durch die Auflösung der *For-Each*-Schleife, die Replikation der in der Schleife enthaltenen Anweisungen auf die zu beteiligenden Ausführungseinheiten und den Aufruf der entstehenden Partitionen mit jeweils einem zu bearbeitenden Datensatz erfolgen.

Das strukturierende *Flow*-Element enthält ebenfalls Aktivitäten, welche im einfachsten Fall parallel zueinander oder in beliebiger Reihenfolge ausgeführt werden können, und entspricht somit im Wesentlichen einer Blockstruktur der in Abschnitt 8.3.1 diskutierten parallelen Verzweigung. Die *Flow*-Blockstruktur besitzt jedoch zusätzlich die Möglichkeit, explizite gerichtete Beziehungen (*Links*) zwischen parallelen Aktivitäten zu definieren, welche

	WS-BPEL-Elemente	Migration	Physische Partitionierung
Variablen	*Variable*	Weitergabe (Sequenz) oder Replikation (Parallelität)	Partitionierung bzw. Replikation der für die Partition relevanten Variablen
Impliziter Kontrollfluss	Verzweigung (*Link* mit *Transition Condition*)	Zusammenfassung der Auswertung der Transitionsbedingungen mit der Ausführung der (atomaren) Aktivität	Zusammenfassung der Auswertung der Transitionsbedingungen mit der Ausführung der (atomaren) Aktivität
Andere Elemente	*Scope*	vor, nach und innerhalb der Ausführung des *Scope*	vor und nach dem *Scope*, innerhalb des *Scopes* durch Auflösung des *Scopes*
	Fault Handler	vor, nach und innerhalb der Ausführung des *Fault Handler*	vor und nach dem *Fault Handler*, innerhalb des *Fault Handler* durch Auflösung des *Fault Handler*
	Compensation Handler	vor, nach und innerhalb der Ausführung des *Compensation Handler*, ggf. Zuweisung zu der Ausführungseinheit, welche für die Ausführung der zu kompensierenden Aktivitäten verantwortlich gewesen ist	vor und nach dem *Compensation Handler*, innerhalb des *Compensation Handler* durch Auflösung des *Compensation Handler*, ggf. Zuweisung zu der Ausführungseinheit, welche für die Ausführung der zu kompensierenden Aktivitäten verantwortlich gewesen ist
	Event Handler	vor und nach dem Gültigkeitsbereich des *Event Handler*, innerhalb des Gültigkeitsbereichs des *Event Handler* nur, wenn Ereignisse bei der Zielausführungseinheit verlustfrei re-abonniert werden können	vor und nach dem *Event Handler*, innerhalb des *Event Handler* durch Auflösung des *Event Handler*
Festlegung der Verteilung zur Entwicklungszeit	-	ja (durch benutzerdefinierte Restriktionen)	ja (durch Zuweisung der Partitionen)
Festlegung der Verteilung zur Laufzeit	-	fortwährend	ggf. einmalig nach Instantiierung
Rebinding von Ressourcen	-	nur begrenzt, i.d.R. feste Endpunkte angegeben	nur begrenzt, i.d.R. feste Endpunkte angegeben
Vertraulichkeit von verteilten Prozesspartitionen	-	durch zusätzliche Verschlüsselung	automatisch erfüllt

Tabelle 8.3: Migrierbarkeit von WS-BPEL-Prozessen und Vergleich mit dem allgemeinen Konzept der physischen Partitionierung (Teil 2) (ZKML10)

ausdrücken, dass die Zielaktivitäten (*Targets*) eines solchen *Links* erst gestartet werden dürfen, wenn die Ausführung der Quellaktivitäten (*Sources*) abgeschlossen sind. Jeder ausgehende *Link* kann dabei mit einer Bedingung (*Transition Condition*) versehen werden, so dass hierüber die Modellierung bedingter Verzweigungen in Graphstruktur möglich ist. Analog dazu können zur Zusammenführung der entstehenden Pfade Bedingungen definiert werden, welche die Art und Anzahl der zu erwartenden eingehenden Pfade angeben (*Join Condition*). Rein strukturell sind diese Kontrollflusskonstrukte jeweils innerhalb der (fachlichen) Aktivitäten gekapselt. Die Vermengung der Block- und Graphstrukturen ist daher für die Migration unkritisch, sofern die Auswertung der Bedingung (mit entsprechender Aktivierung der Zielaktivität(en) eines *Links* und etwaiger Dead Path Elimination) analog ihrer syntaktischen Strukturierung in WS-BPEL mit der fachlichen Aktivität zu einer atomaren Aktivität im Sinne des Migrationsmodells zusammengefasst wird. In diesem Fall ist der Status des Links über die Zustände der Quell- und Zielaktivitäten und der Ausführungsfortschritt über die Angabe der Startaktivität(en)

determiniert. Eine parallel verteilte Ausführung von *Flow*-Aktivitäten wurde an dieser Stelle nicht im Detail betrachtet. Es ist jedoch festzuhalten, dass hinsichtlich der Durchführung von Zusammenführungen und Synchronisationen dieselben Einschränkungen gelten wie für XPDL-Prozesse (vgl. Abschnitt 8.3.1, so dass jeweils die Festlegung eines (statischen oder dynamischen) „Treffpunktes“ zur Vereinigung der erzeugten Replikate notwendig ist. Dies gilt sowohl für die explizite Beendigung der Ausführung des *Flow*-Elements als auch für die implizit durch Graphstruktur erzeugten *Join*-Aktivitäten innerhalb des *Flow*-Elements [ZKML10].

Die strukturierende Aktivität *Pick* definiert, dass der Kontrollfluss des Prozesses das Eintreten ein oder mehrerer Ereignisse oder das Erreichen einer Zeitbeschränkung abzuwarten hat. Tritt ein spezifiziertes Ereignis ein (*on Event*) oder wird die angegebene Zeitbeschränkung erreicht (*on Alarm*), wird der mit dem Ereignis bzw. der Zeitbeschränkung verbundene Kontrollfluss ausgeführt. Da das Eintreten eines Ereignisses über das Empfangen einer (fachlichen) Nachricht erfolgt, gelten für die *Pick*-Aktivität bei einer Verteilung zunächst einmal dieselben Einschränkungen wie für die Aktivität *Receive*[2]. Die Migration von *Pick*-Aktivitäten muss daher unterbleiben oder der Prozess muss an eine Ausführungseinheit migriert werden, welche dem Sender der (Ereignis-)Nachricht als Empfänger bekannt ist. Dies hat auch den Vorteil, dass die Auswertung der Zeitbeschränkung nicht durch eine Migration unterbrochen werden kann.Diese Einschränkung der Migrierbarkeit der *Pick*-Aktivität bedeutet jedoch nicht, dass die gesamte *Pick*-Aktivität atomar durch diejenige Ausführungseinheit ausgeführt werden muss, welche das Ereignis empfängt bzw. die Zeitbedingung auswertet. Da der Inhalt der (Ereignis-)Nachricht stets in einer Prozessvariablen gespeichert wird, kann der durch das Ereignis (bzw. die Zeitbeschränkung) initiierte Kontrollfluss unter den gegebenen Rahmenbedingungen beliebig verteilt werden. Die genannten Einschränkungen und Verteilungsmöglichkeiten gelten dabei analog zu den zuvor beschriebenen *Invoke-Receive*-Aktivitätspaaren entsprechend auch für die physische Partitionierung von Prozessmodellen [ZKML10].

Eine weitere Möglichkeit zur Steuerung des Kontrollflusses besteht in der Verarbeitung von Ereignissen, welche nicht fest in den Kontrollfluss eingebunden sind, sondern an einen bestimmten Gültigkeitsbereich geknüpft sind, so dass das Eintreten des Ereignisses parallel zum aktuell ausgeführten Kontrollfluss die Ausführung einer Ereignisbehandlungsroutine anstößt (*Event Handler*). Der Gültigkeitsbereich kann den ganzen Prozess oder nur einen festgelegten Teilbereich (*Scope*) umfassen. Eingehende (Ereignis-)Nachrichten werden berücksichtigt, sobald der Kontrollfluss den Gültigkeitsbereich erreicht hat, für welchen der *Event Handler* definiert ist. Dabei wird bei jedem Empfang einer beliebigen im *Event Handler* definierten Nachricht jeweils der dort angegebene Kontrollfluss ausgeführt. Wegen der bereits bei der *Receive*- und

[2]Liegt nur eine fachliche Nachricht und keine ereignisgesteuerte Architektur zugrunde, so ist ein (Re-)Abonnement von Ereignissen an der Ereignisquelle nach einer Migration nicht ohne weiteres möglich.

der *Pick*-Aktivität diskutierten Beschränkungen kann eine Migration innerhalb von Gültigkeitsbereichen mit *Event Handler* nur erfolgen, wenn die Zielausführungseinheiten sich bei der Ereignisquelle für den Empfang der Ereignisse neu registrieren können und die Migration des Prozesses so erfolgt, dass während der Übergabe der Verantwortlichkeit für die Prozessausführung keine Ereignisse verloren gehen können. Bei der physischen Partitionierung kann eine Verteilung des Prozesses über die Auflösung des Gültigkeitsbereiches und eine statische Zuordnung von Ereignis-Reaktions-Paaren vorgenommen werden [ZKML10]. Hierbei liegen folglich nicht alle Ereignisbehandlungsroutinen allen Ausführungseinheiten vor oder müssen geeignet repliziert werden, um bei Eintritt eines Ereignisses ein zu einer nicht-verteilten Ausführung äquivalentes Verhalten zu erreichen.

Eine weitere relevante Eigenschaft von WS-BPEL ist, dass die im Prozess definierten Elemente – mit Ausnahme der Variablen (*Variables*) – nicht notwendigerweise über eine innerhalb des Prozesses eindeutige Bezeichnung verfügen müssen. Für Aktivitäten existiert das Attribut *Name*, welches für die Referenzierung der Zustandsinformationen aus den Migrationsdaten genutzt werden kann. Hierfür muss daher einmalig vor einer Verteilung des Prozesses die Eindeutigkeit der Aktivitätsnamen sichergestellt werden, z. B. indem allen Aktivitätsnamen ein eindeutiger Suffix angehängt wird. Bei einer physischen Partitionierung von Prozessmodellen ist dies nicht notwendig.

8.3.3 BPMN-Prozesse

Die Prozessbeschreibungssprache *BPMN* (*Business Process Model and Notation*) ermöglicht eine graphische Darstellung von Prozessmodellen und besitzt seit der durch die OMG standardisierten Version 2.0 auch eine ausführbare XML-Repräsentation (vgl. auch Abschnitt 5.2.4). Das Ziel von BPMN 2.0 ist im Wesentlichen die Bereitstellung einer einzigen Spezifikation als gemeinsame Notation, Metamodell und Austauschformat von Prozessen [OMG11]. In diesem Abschnitt wird die Migrierbarkeit der auch von der untersuchten Prozess-Engine *Activiti* (vgl. Abschnitt 8.2.3) unterstützten BPMN-Prozesse der Version 2.0 in XML-Repräsentation betrachtet. Da die BPMN-Spezifikation einen relativen hohen Abstraktionsgrad aufweist, werden einige Sprachelemente bei *Activiti* durch plattformspezifische Erweiterungen konkretisiert. Es wird daher im Folgenden jeweils darauf hingewiesen, wenn die Migrierbarkeit von BPMN-Prozessinstanzen durch spezielle Erweiterungen von BPMN positiv oder negativ beeinflusst wird.

BPMN-Prozesse besitzen im Allgemeinen eine Graphstruktur, können jedoch innerhalb dieser Graphstruktur auch verschiedene Blockstrukturen zur Definition von Gültigkeitsbereichen und zur Einbindung von Subprozessen aufweisen. Die durch BPMN definierten Kontrollflusselemente können in Knoten (*Flow Objects*), Transitionen (*Connecting Objects*), Zuordnungen zu Prozessteilnehmern in Form von organisatorischen Rollen und Einheiten (*Pools* und *Lanes*) sowie Artefakten (*Artifacts*) eingeteilt werden. Zu den Knoten

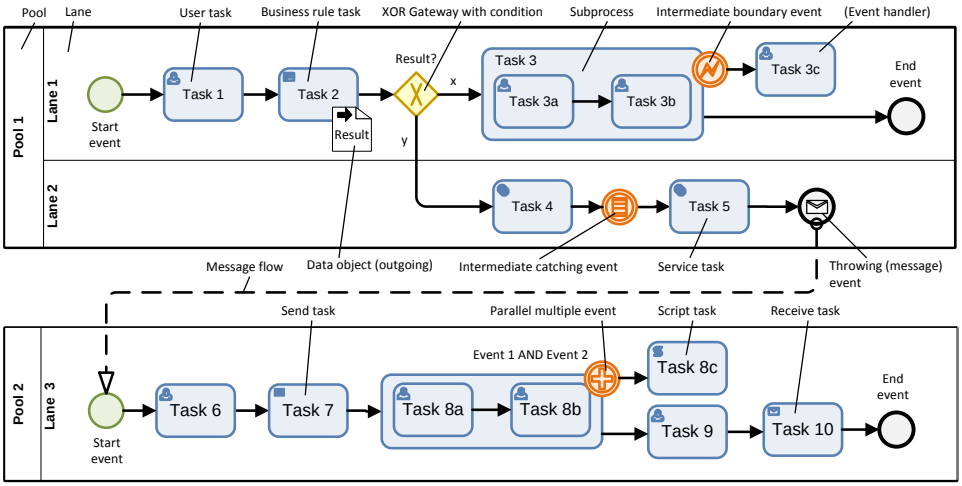


Abbildung 8.2: Graphische Repräsentation eines BPMN-Beispielprozesses (BDH+12)

zählen Aktivitäten (*Tasks*), Verzweigungsobjekte (*Gateways*) und Ereignisse (*Events*). Jeder Knoten kann sich dabei auf höchstens einer *Lane* befinden, welche angibt, welcher konkrete Prozessteilnehmer oder organisatorische Rolle für die Ausführung dieses Knotens verantwortlich ist. Prozessteilnehmer und Rollen müssen dabei nicht zwingend einer Person bzw. einer Gruppe von Personen zugeordnet sein, sondern können auch Maschinen oder IT-Systeme in Form von Hard- und/oder Software bezeichnen. Eine Menge von *Lanes* kann einem *Pool* zugeordnet sein, welcher die *Lanes* in Form einer organisatorischen Einheit umschließt. Mögliche Transitionen in Form von Kontrollflusspfaden (*Sequence Flows*) können Knoten auf derselben *Lane* oder über verschiedene *Lanes* desselben *Pools* verbinden oder können in Form von Nachrichten (*Message Flows*) Knoten von Lanes verschiedener Pools miteinander in Verbindung setzen. Artefakte umfassen Datenobjekte (*Data Objects*), Gruppierungen bzw. Gültigkeitsbereiche (*Groups*) und Annotationen (*Annotations*) [OMG11]. Eine Übersicht der genannten Elemente ist als Beispielprozess in Abbildung 8.2 dargestellt. Im Folgenden werden die wichtigsten Kontrollflusskonstrukte von BPMN im Detail in Hinblick auf die Migrierbarkeit von BPMN-Prozessinstanzen untersucht.

Tasks

Ein BPMN-Prozess besteht aus wenigstens einem *Task*, welcher die kleinste Einheit einer auszuführenden fachlichen Aufgabe in BPMN darstellt. In Hinblick auf das in Abschnitt 6.2.2 vorgestellte Migrationsmodell und im Vergleich mit XPDL und WS-BPEL entspricht ein *Task* einer atomaren Aktivität [BDH+12]. Im Gegensatz zu den durch WS-BPEL definierten Aktivitäten werden bei BPMN jedoch die fachlichen Aktivitäten nicht nach der Art ihres Kommunikationsmodells unterschieden, sondern geben an, wie die Ausführung der definierten Aufgabe durchgeführt wird. Es können dazu die folgenden Aufgabentypen unterschieden werden [BDH+12]:

- **Service Task:** Der *Service Task* ruft eine Software-Anwendung (z. B. eine Java-Klasse oder einen Web Service) auf, lässt eine vorgegebene fachliche Aufgabe durchzuführen und nimmt etwaige Ergebnisse entgegen. Die Kommunikation mit der Software-Anwendung findet dabei im Rahmen einer synchronen Kommunikation statt, welche durch die atomare Aktivität gekapselt ist. Es ist daher im Allgemeinen vor oder nach der Aktivität eine Migration des Prozesses möglich. Aufgrund des hohen Abstraktionsgrades von BPMN ist in vielen Fällen nach einer Migration auch ein Rebinding der Software-Anwendung möglich. Zum Beispiel können Referenzen auf Java-Klassen durch die Angabe eines relativen Pfades auf die Ressource angegeben werden, so dass nach einer Migration – bei Vorliegen derselben Klassenstruktur auf dem Zielausführungssystem – die Ausführung ohne Anpassungsaufwand mit der funktional äquivalenten lokalen Ressource fortgesetzt werden kann. Auf der anderen Seite können Ressourcen jedoch auch eindeutig referenziert werden. So kann zum Beispiel ein Web Service durch eine eindeutige URL angegeben werden, so dass nach einer Migration dieselbe Ressource zugegriffen werden kann (bzw. muss) wie vor der Migration.

- **User Task:** Der *User Task* bezeichnet die Interaktion mit einem menschlichen Prozessteilnehmer. Hierzu werden dem Benutzer in der Regel die Aufgabe und etwaige Ausgabedaten angezeigt und nach Abschluss der Aufgabe ggf. Eingabedaten entgegengenommen. Die hierfür erforderlichen Benutzungsschnittstellen sind nicht Inhalt der BPMN-Spezifikation, sondern werden in der Regel durch eine Referenz auf aufgabenspezifische Dokumente zur Repräsentation der Benutzungsschnittstellen (z. B. HTML-Formulare) ausgedrückt, welche im Rahmen des Deployments zusammen mit dem Prozess auf dem Prozessmanagementsystem installiert werden. Im Fall einer Migration des Prozesses müssen diese Dokumente daher mitgeführt werden, was deren Ausführbarkeit bzw. Darstellbarkeit auf der Zielausführungseinheit voraussetzt. Eine Möglichkeit zur weiteren Flexibilisierung besteht in der Anwendung abstrakter Benutzungsschnittstellen (vgl. Abschnitt 6.4). Ist die Voraussetzung der Übertragbarkeit der Benutzungsschnittstellen erfüllt, kann eine Migration vor oder nach der Benutzerinteraktion erfolgen, sofern der für die Aufgabe vorgesehene Benutzer (konkret oder im Rahmen einer Rollenbeziehung) beim Zielausführungssystem zur Verfügung steht. Ist ein individueller Benutzer nicht konkret vorgegeben und können dementsprechend mehrere Personen die Ausführung einer anstehenden Aufgabe übernehmen, verwenden Prozessmanagementsysteme in der Regel (öffentliche) Aufgabenlisten, aus denen autorisierte Prozessteilnehmer sich die Aufgabe selbst fest zuweisen können (*Checkout*). Eine Migration des Prozesses sollte dabei auch nach der benutzerdefinierten Zuweisung des Prozesses noch möglich sein, zum Beispiel falls der Prozessteilnehmer die Aufgabe auf einem mobilen Endgerät bearbeiten möchte. Um

zwischenzeitlich aber Migrationen an andere Prozessteilnehmer zu verhindern, sollte daher bei einem *Checkout* der Aufgabe in den Migrationsdaten eine neue Zuweisungsbeschränkung eingetragen werden, welche nur eine Bearbeitung durch den aktuellen Prozessteilnehmer (unabhängig von seiner Zuordnung zu einer Ausführungseinheit) erlaubt. Nach einer Migration des Prozesses würde diese Aufgabe dem betreffenden Prozessteilnehmer daher sofort wieder zugewiesen werden können, ohne erneut in einer etwaigen öffentlichen Aufgabenliste angezeigt zu werden.

- **Manual Task:** Der *Manual Task* stellt das vollständig manuelle Ausführen einer Aufgabe dar. Die manuelle Ausführung wird dabei nicht vom Prozessmanagementsystem verwaltet, so dass der *Manual Task* in der Regel nicht in einer direkt ausführbaren Repräsentation des Prozesses enthalten ist.

- **Script Task:** Der *Script Task* erlaubt das Ausführen von Programmfragmenten geringen Umfangs, welche in Form von Skripten direkt in den *Skript Task* eingebettet werden können. Bei einer etwaigen Migration werden die Skripte daher automatisch mit dem Prozessmodell zur ausgewählten Zielausführungseinheit transportiert und können – vorausgesetzt, die verwendete Skriptsprache wird bei der Ausführungseinheit unterstützt – prinzipiell auch nach einer Migration umgehend ausgeführt werden. Die Migration von beliebigem ausführbarem Programmcode birgt jedoch unter Umständen ein hohes Sicherheitsrisiko, so dass die Ausführung von *Script Tasks* auf Prozesse von vertrauenswürdiger Quelle beschränkt werden sollte (vgl. [ZHKK10]). Da *Skript Tasks* eine nach dem Migrationsmodell atomare Aktivität darstellen, kann die Ausführung von Skripten im Kontext dieser Arbeit nicht durch eine Migration unterbrochen werden.

- **Send Task** und **Receive Task:** Der *Send Task* und der *Receive Task* stellen Aktivitäten zum Nachrichtenaustausch dar. Besteht eine Aktivität nur darin, eine Information zu versenden (*Send Task*) ohne dabei später eine Antwort zu erwarten, ist eine Migration des Prozesses ohne Einschränkungen möglich. Gehören *Send Task* und *Receive Task* jedoch zu einem Aktivitätspaar zur asynchronen Kommunikation (vgl. z. B. Task 7 und Task 10 in Abbildung 8.2), so muss die Migration dahingehend eingeschränkt werden, dass beide Aktivitäten durch dieselbe Ausführungseinheit verwaltet werden, damit die Antwortnachricht für den *Receive Task* korrekt zugestellt werden kann. Dies kann zum Beispiel durch eine statische oder dynamische Verteilungsstrategie (*Assignment Strategy*) festgelegt werden (vgl. Abschnitt 6.2.2).

- **Business Rule Task:** Ein *Business Rule Task* verweist auf eine Anwendung zur Auswertung von Geschäftsregeln (vgl. Abschnitt 3.6). Die Spezifikation von Geschäftsregeln und die entsprechende Verwendung

von Geschäftsregel-Managementsystemen wird nicht durch BPMN geregelt. Die Migration von Aufgaben zur Auswertung von Geschäftsregeln setzt daher voraus, dass das Zielausführungssystem Zugriff auf ein kompatibles Geschäftsregel-Managementsystem besitzt. Zudem muss berücksichtigt werden, dass bei einem etwaigen Rebinding eines *Business Rule Task* auf eine andere Datenbasis unter Umständen andere Ergebnisse erzielt werden, welche den der Auswertung folgenden Kontrollfluss des Prozesses (gewollt oder ungewollt) beeinflussen können.

Zusammenfassend kann festgehalten werden, dass der hohe Abstraktionsgrad von BPMN einerseits eine hohe Flexibilität hinsichtlich des Rebindings oder der Migration von Ressourcen ermöglicht, anderseits aber auch durch die Notwendigkeit, nicht im Standard enthaltene Konkretisierungen durch plattformspezifische Erweiterungen zu kompensieren, eine dynamische Verteilung von Prozessinstanzen auf entsprechend kompatible Ausführungseinheiten begrenzt. Das Rebinding von Ressourcen erfordert in allen Fällen ein gemeinsames Verständnis, wie zum Beispiel durch die Definition von geeigneten Ontologien oder durch ein gemeinsames Organisationsmodell. Der Standard BPMN allein ist daher in offenen und heterogenen verteilten Systemen für eine dynamische Verteilung von Prozessen mittels Migration von Prozessinstanzen nicht ausreichend. So ist insbesondere eine ergänzende Einigung auf gemeinsame technologische Grundlagen für Organisationsmodelle, Benutzungsschnittstellen, Skriptsprachen und Geschäftsregeln empfehlenswert [BDH+12].

Teilprozesse

Ein Teilprozess (*Subprocess*) stellt in BPMN eine Blockaktivität dar, welche bei der Ausführung durch einen in der Prozessbeschreibung gekapselten (analog eines Prozesses strukturierten) Kontrollfluss ersetzt wird. In Hinblick auf das Migrationsmodell erfolgt somit eine Abbildung als strukturierte Aktivität, welche – unter Berücksichtigung der Migrationsfähigkeit der gekapselten Kontrollflusselemente – eine Migration sowohl vor, nach und während der Ausführung des Blocks erlaubt. Die Prozessbeschreibung des Teilprozesses muss dazu bei der Zielausführungseinheit vorliegen, was bei BPMN sowohl in der graphischen Darstellung als auch in der XML-Repräsentation unterstützt wird (vgl. Task 3 in Abbildung 8.2) [BDH+12].

Gateways

Ein *Gateway* stellt ein Kontrollflusselement dar, welches die (bedingte) Verzweigung oder Zusammenführung von alternativen oder parallelen Prozesspfaden einleitet. Es existieren typische Arten der Verzweigung bzw. Zusammenführung, wie *XOR-*, *OR-* und *AND-Gateways*, deren etwaige Bedingungen wie bei XPDL an die aus- bzw. eingehenden Transitionen des Gateways geknüpft sind (vgl. Abschnitt 8.3.1). Zudem existieren spezielle *ereignisbasierte Gateways*, welche auf Basis eingehender Ereignisse einen ausgehenden Kon-

trollflusspfad zur Ausführung aktivieren. Analog zu den oben genannten Bedingungen befinden sich hierzu die Ereignisse als jeweils erste dem Gateway im Kontrollfluss folgende Elemente auf den ausgehenden Pfaden.

In Hinblick auf die Migration von Prozessinstanzen ist es vorteilhaft, ein Gateway mit der Auswertung der dazugehörigen Bedingungen bzw. dem Warten auf das Eintreten der Ereignisse als atomare Aktivität zu betrachten, welche durch die Prozess-Engine selbst ausgeführt wird. Durch diese Vorgehensweise kann die „Ausführung“ des *Gateways*, d. h. im Speziellen die Auswertung der Bedingung bzw. das Erwarten eines Ereignisses, bei Bedarf einer bestimmte Ausführungseinheit zugeordnet werden. Ein *Gateway* ist demnach im Zustand *ready* sobald der Kontrollfluss das *Gateway* erreicht hat, im Zustand *executing* solange die Bedingungen ausgewertet werden bzw. die relevanten Ereignisse noch nicht eingetreten sind, und im Zustand *executed* bzw. *finished* sobald die Auswertung abgeschlossen ist und der nachfolgende Kontrollfluss aktiviert wurde (vgl. Abschnitt 6.2.4). Analog dazu werden die Zustände der nachfolgenden nicht auszuführenden Aktivitäten in den Zustand *skipped* versetzt (Dead Path Elimination) [BDH+12]. Eine Migration ist daher analog zu XPDL im Allgemeinen sowohl vor als auch nach dem *Gateway* möglich. Auf die Einschränkungen, welche unter Umständen zusätzlich aus dem Empfangen von Ereignissen resultieren, wird im folgenden Abschnitt genauer eingegangen.

Ereignisse

Ereignisse (*Events*) erlauben eine flexible Behandlung von außerhalb des sequentiell strukturierten Kontrollflusses auftretenden Situationen. BPMN unterscheidet ereignisbasierte Kontrollflussstrukturen zunächst danach, ob ein Ereignis durch eine Prozessinstanz *ausgelöst* oder durch die Prozess-Engine für eine Prozessinstanz *aufgefangen* wird. In Bezug auf die Migration von Prozessinstanzen stellt das *Auslösen* eines Ereignisses eine atomare Aktivität dar, welche von der Prozess-Engine selbst ausgeführt wird. Wird das ausgelöste Ereignis intern durch eine Ereignisbehandlungsroutine innerhalb derselben Prozessinstanz verarbeitet, so wird der entsprechende Kontrollfluss aktiviert und ausgeführt, wobei Migrationen während der Ausführung der Ereignisbehandlung wieder erlaubt sein können. Wird das Ereignis (alternativ oder zusätzlich) durch eine andere Prozessinstanz derselben oder einer anderen Ausführungseinheit empfangen, so ist eine Migration der auslösenden Prozessinstanz hierdurch ebenfalls nicht beeinträchtigt [BDH+12]. Da während einer Migration keine Ereignisse durch den Prozess selbst ausgelöst werden können, stellt das reine Auslösen eines Ereignisses daher in Bezug auf die Migration kein Problem dar.

Im Gegensatz dazu erfordert das *Auffangen* von Ereignissen einer externen Ereignisquelle (d. h. einer Ereignisquelle ungleich der betrachteten Prozessinstanz) eine detailliertere Untersuchung der zugrunde liegenden Architektur, der Integration des Ereignisses in den Kontrollfluss und der Art des Ereignisses. Dabei ist für die Migrierbarkeit von Prozessinstanzen insbesondere rele-

vant, ob es sich überhaupt um ein Ereignis im Sinne der in Abschnitt 3.7 vorgestellten ereignisgesteuerten Architektur für prozessorientierte Anwendungen handelt. Die Spezifikation von BPMN lässt dies offen. Es können insbesondere zwei unterschiedliche Sichtweisen identifiziert werden:

- **Ereignis ausschließlich auf Anwendungsebene:** Ausschließlich auf Anwendungsebene dargestellte Ereignisse sind an einen bestimmten Empfänger gerichtete Nachrichten, welche von der Prozess-Engine empfangen werden und nur im Kontext der Prozessausführung als Ereignisse abgebildet werden. Sie stellen bei der Kommunikation zwischen Ereignisquelle und Ereigniskomsument jedoch keine Ereignisse im Sinn einer ereignisgesteuerten Architektur dar. Entsprechend werden die dazugehörigen Nachrichten nicht abonniert und können dementsprechend nach einer Migration nicht ohne weiteres von einer anderen Ausführungseinheit re-abonniert werden. In diesem Fall gelten die bereits bei der Beschreibung des *Receive*-Tasks erläuterten Einschränkungen.
- **Ereignisse auf Anwendungs- und Kommunikationsebene:** Ereignisse auf Kommunikationsebene stellen Ereignisse im Sinn einer ereignisgesteuerten Architektur dar. Die Ereignisquelle emittiert ihre Ereignisse daher in Form von Nachrichten unabhängig davon, welche und wie viele Ereigniskonsumenten sich für den jeweiligen Ereignistyp registriert haben. Es ist daher nach einer Migration im Prinzip ein Re-Abonnement der relevanten Ereignistypen der vorherigen Ereignisquelle oder sogar ein Rebinding einer neuen Ereignisquelle möglich.

Wie bereits bei der Betrachtung von WS-BPEL festgestellt wurde, ist die Migration von Kontrollflussstrukturen, welche lediglich gerichtete Nachrichten als Ereignisse auf Anwendungsebene abbilden, stark eingeschränkt, da der Prozess als Empfänger der Nachricht in diesem Fall bei dem Sender der Nachricht fest vorgeschrieben ist. Ein Nachsenden der Nachrichten von einer Ausführungseinheit zur anderen wäre zwar prinzipiell möglich, würde jedoch – insbesondere bei einer mehrfachen Migration – die Zustellung der Nachrichten und somit das Auffangen des Ereignisses mehr oder weniger stark verzögern. Bei BPMN ist insbesondere das Zwischenereignis zum Empfangen einer Nachricht (*Message Event*, vgl. Tabelle 8.5) von diesem Problem betroffen, da hierbei davon ausgegangen werden muss, dass die Nachricht auf Kommunikationsebene stets an die individuelle Prozessinstanz adressiert ist. Für die übrigen Ereignisarten (vgl. Tabelle 8.5) kann auch die Möglichkeit einer zugrunde liegenden ereignisbasierten Kommunikationsinfrastruktur in Betracht gezogen werden. Hierbei ist dann für jedes von der Prozess-Engine aufzufangende Ereignis ein vorheriges Abonnement des in der Prozessbeschreibung spezifizierten Ereignistyps bei der (konkret oder abstrakt) angegebenen Ereignisquelle erforderlich.

Geht man davon aus, dass Ereignisse nicht persistent gespeichert werden, ist zur Einschränkung der Migration oder zur Übertragung des Ereignis-

Integration des Ereignisses im Kontrollfluss	Beschreibung	Spätester Zeitpunkt für Abonnement
Top level start event	Instantiierung des Prozesses	Deployment
Sub-process start event	Instantiierung des Teilprozesses	Ausführungsbeginn des aufrufenden (Teil-)Prozesses
Intermediate boundary event	Ereignisbehandlung während der Ausführung eines Teilprozesses	Ausführungsbeginn des Teilprozesses
Intermediate catching event	Fortführung des Kontrollflusses nach Warten auf den Eintritt des Ereignisses	Beginn des Wartezustands

Tabelle 8.4: Arten der Integration von Ereignissen in den Kontrollfluss von BPMN-Prozessen (BDH+12)

abonnements bei einer Migration des Prozesses zudem relevant, wie sich ein Ereignis in den Kontrollfluss des Prozesses einbettet. BPMN unterscheidet vier unterschiedliche Möglichkeiten zur Integration von Ereignissen, welche zusammenfassend in Tabelle 8.4 dargestellt sind. Ereignisse, welche eine Instantiierung des Prozesses auslösen (*Top level start events*) müssen für eine potentielle Migration der Prozessinstanz nicht betrachtet werden, da die Prozessinstanz erst bei Eintreten des Ereignisses erzeugt wird. Die Instantiierung von Teilprozessen ist grundsätzlich möglich, sofern der jeweils aufrufende Prozess instantiiert wurde (*Sub-process start event*). Die hierfür erforderlichen Ereignistypen müssen daher spätestens mit der Instantiierung des jeweilig übergeordneten Prozesses abonniert werden. Ereignisse, welche für einen bestimmten Gültigkeitsbereich definiert werden (*Intermediate boundary events*) müssen ab Ausführungsbeginn des Teilprozesses empfangen werden können, für welchen bei Eintreten des Ereignisses eine Ereignisbehandlung durchgeführt werden soll (vgl. z. B. „Task 3a“ und „Task 3b“ in Abbildung 8.2). Lokal innerhalb des Kontrollflusses definierte Ereignisse (*Intermediate catching events*) veranlassen den Kontrollfluss, an der Position der Ereignisdefinition anzuhalten, bis das definierte Ereignis eingetreten ist. Ein Abonnement bzw. ein Empfang des Ereignisses ist daher erst bei Erreichen der Ereignisdefinition im Kontrollfluss des Prozesses relevant. Das Abonnement zu einem (nicht wiederholt aufzufangenden) Ereignis kann im Allgemeinen aufgehoben werden, wenn das Ereignis bereits eingetreten ist oder die ereignisbasierte Kontrollflussstruktur vollständig durchlaufen oder übersprungen wurde [BDH+12].

Die Zusammenfassung des Abonnements, des Empfangs des Ereignisses, des Anstoßens der Ereignisbehandlung und der Beendigung des Abonnements als atomare Aktivität im Sinn des Migrationsmodells würde gewährleisten, dass keine Ereignisse während einer Migration verpasst werden können und dass Migrationen zwischen verschiedenen ereignisbasierten Kontrollflussstrukturen möglich sind. Die Behandlung eines Ereignisses als atomare Aktivität ist jedoch in Hinblick auf die flexible Verteilung des Prozesses nur für die direkt in den Kontrollfluss eingebetteten Ereignisse (*Intermediate catching events*) vorteilhaft. Für ereignisbasierte Gültigkeitsbereiche (*Sub-process start event* und *Intermediate boundary event*) würde diese Strategie im ungünstigsten Fall die Verteilung des gesamten Prozesses verhindern, zum Beispiel falls der gesam-

Ereignis	Beschreibung	Komplexes Ereignis	Prozess-internes Ereignis	Lokale Ereignis-quelle	Externe Ereignis-quelle	Migration
Compensation	Ausführung einer Kompensation	–	✓	–	–	vor oder nach dem Eintreten des Ereignisses
Error	Behandlung definierter Fehler	–	✓	–	–	vor oder nach dem Eintreten des Ereignisses
Escalation	Aufruf übergeordneter Hierarchieebene	–	✓	–	–	vor oder nach dem Eintreten des Ereignisses
Terminate	Beendigung der Prozessinstanz	–	✓	–	–	vor dem Eintreten des Ereignisses
Link	Neuer Sequenzfluss	–	✓	–	–	vor oder nach dem Eintreten des Ereignisses
Signal	Signal innerhalb oder über mehrere Prozesse	–	✓	✓	–	Re-Abonnement des Ereignistyps bzw. Rebinding der Ereignisquelle notwendig
Cancel	Abbruch (einer Transaktion)	–	✓	✓	–	Re-Abonnement notwendig
Message	Empfang einer Nachricht	–	–	✓	–	Re-Abonnement notwendig
Timer	Zeitereignis	–	–	✓	–	Re-Abonnement des Ereignistyps bzw. Rebinding der Ereignisquelle notwendig (nur bei absoluten Zeitbezugspunkten)
Conditional	Reaktion auf veränderte Bedingungen	*	*	*	*	abhängig von Ereignisquelle und Zusammensetzung des Ereignisses
Parallel/Multiple	Eintreten aller bzw. eines Ereignisse(s) aus einer definierten Menge von Ereignissen	*	*	*	*	vor dem Eintreten des ersten (Teil-)Ereignisses oder nach vollständigem Eintreten des Ereignisses

Tabelle 8.5: Arten von Ereignissen in BPMN 2.0 (BDH+12)

te Prozess durch ein *Intermediate boundary event* umschlossen ist [BDH+12]. Soll die Migration auch innerhalb von komplexen ereignisbasierten Kontrollflussstrukturen erfolgen, so muss die Zielausführungseinheit die Abonnements der relevanten Ereignistypen zur Laufzeit übernehmen können und die Ereignisverarbeitung bei Eintreten eines Ereignisses anstoßen bzw. fortsetzen können. Da jedoch auch während der Migration des Prozesses Ereignisse eintreten können, darf die Quellausführungseinheit das Abonnement der relevanten Ereignisse erst beenden, sobald die Zielausführungseinheit die Abonnements vollständig übernommen hat. In der Zwischenzeit eintretende Ereignisse müssen der Zielausführungseinheit nachgesendet werden, wobei auch eingehende Duplikate von Ereignissen geeignet ausgeschlossen werden müssen [BDH+12].

Eine weitere relevante Eigenschaft von Ereignisstrukturen in BPMN ist, dass diese einfach oder komplex sein können. Tabelle 8.5 zeigt eine Übersicht über die einzelnen Arten von Ereignissen, welche in BPMN zur Verfügung stehen. Einfache prozessinterne Ereignisse, wie das Anzeigen eines Fehlers (*Error*) oder die Auslösung einer Kompensation (*Compensation*) sind in Bezug auf eine zwischenzeitliche Migration unkritisch, da das Ereignis dabei nicht aus

mehreren Bestandteilen zusammengesetzt ist, die Ereignisverarbeitung somit automatisch als atomarer Schritt erfolgt und somit entweder auf dem Quell- oder dem Zielausführungssystem erfolgen kann. Einfache Ereignisse externer Ereignisquellen (wie möglicherweise das *Conditional*-Ereignis) stellen sich in Hinblick auf die Migrierbarkeit – falls ein Re-Abonnement oder ein Rebinding möglich ist – wie einfache prozessinterne Ereignisse dar [BDH+12]. Komplexe Ereignisse (z. B. *(Parallel) multiple events*), welche sich aus mehreren internen Ereignissen zusammensetzen, sind in Bezug auf eine Migration im Prinzip ebenfalls unproblematisch, sofern sie sich wie strukturierte Aktivitäten verhalten und das Auslösen bzw. Eintreten eines Teilereignisses in den Migrationsdaten vermerkt werden kann [BDH+12]. Wie in Abschnitt 6.2.6 dargestellt wurde, kann es bei der verteilten Verarbeitung komplexer Ereignisse, welche von externen Ereignisquellen erzeugt und durch die jeweils lokale Ausführungsumgebung (teil-)verarbeitet werden, jedoch zu unerwünschten Seiteneffekten kommen.

In der hier gewählten Lösung wird das Auffangen von Ereignissen daher als atomare Aktivität aufgefasst, welche die Prozess-Engine veranlassen, eine Ereignisverarbeitung durchzuführen. Erreicht der Kontrollfluss das Ereignis bzw. die erste Aktivität des Gültigkeitsbereichs des Ereignisses, wird diese „Ereignisaktivität“ in den Zustand *ready* versetzt. Zu diesem Zeitpunkt muss spätestens auch das Abonnement des dazugehörigen Ereignistyps erfolgt sein, damit Ereignisse aufgefangen werden können. Die „Ereignisaktivität“ nimmt den Zustand *executing* ein, sobald die Ereignisverarbeitung beginnt, d. h. sobald ein (Teil-)Ereignis eintritt. Einfache Ereignisse entsprechen in dieser Sichtweise einer atomaren Aktivität und werden in den Zustand *executed* bzw. *finished* versetzt, sobald das Eintreten des Ereignisses festgestellt und die nachfolgende Aktivität aktiviert wurde. Komplexe Ereignisse enthalten Regeln für die Erkennung von bestimmten Mustern in den eingehenden Ereignissen innerhalb eines bestimmten Zeitfensters und verbleiben im Zustand *executing* bis das Ereignis vollständig eingetreten und die Ereignisverarbeitung somit beendet ist. Die komplexe Ereignisaktivität geht nun in den Zustand *executed* bzw. *finished* über und die nachfolgenden Aktivitäten werden aktiviert. Entsprechend kann die Verarbeitung komplexer Ereignisse nicht durch eine Migration unterbrochen werden. Im Fall eines ereignisgesteuerten Gültigkeitsbereichs (*Intermediate boundary event*) besteht auf diese Weise zudem die Möglichkeit, die Ereignisaktivität in den Zustand *expired* bzw. *terminated* zu versetzen, falls die Ausführung der Aktivitäten in dem jeweiligen Gültigkeitsbereich bereits beendet wurde, bevor das Ereignis (vollständig) eingetreten ist. Die Prozess-Engine kann das Abonnement für den dazugehörigen Ereignistyp nun beenden, sofern nicht eine übergeordnete Kontrollflussstruktur ein Ereignis desselben Typs erwartet [BDH+12].

Pools und Lanes

Über die einfache Modellierung von Prozessen hinaus erlaubt es BPMN, die für die Ausführung der Prozesse notwendige Zusammenarbeit von verschiedenen Ressourcen über Organisationsgrenzen hinweg auszudrücken. In Bezug auf die dynamische Verteilung eines Prozesses kann das *Lane*-Element als Behälter für alle Aktivitäten aufgefasst werden, welche von einer bestimmten Ressource bzw. (auf Basis eines Rollenkonzepts) von einer bestimmten Gruppe von Ressourcen ausgeführt werden sollen (*ressourcenbezogene Verteilung*, vgl. Abschnitt 4.1.2). Das übergeordnete *Pool*-Element kann als Behälter für eine organisatorische bzw. technische Zuordnung ausdrücken, welche Ausführungseinheit für die Navigation des Kontrollflusses und den Aufruf der in den zugeordneten *Lane*-Elementen enthaltenen Ressourcen verantwortlich sein soll (*ausführungsbezogene Verteilung*, vgl. Abschnitt 4.1.2). Verfolgt man diesen Gedanken weiter, so repräsentieren die Nachrichtenflüsse (*Message flows*) jeweils den Übergang von einer lokal-zentralen Ausführung zu einer anderen lokal-zentralen Ausführung bzw. zu einer verteilt parallelen Ausführung. Bildet man diese Kommunikation auf die dynamisch verteilte Ausführung einer Prozessinstanz ab, so kann – unter den oben genannten Einschränkungen – anstelle des Nachrichtenflusses auch eine Migration des Prozesses erfolgen. Es ergibt sich dementsprechend eine alternative Interpretation des Kollaborationsdiagramms, welches als Grundlage für Verteilungsentscheidungen genutzt werden kann [BDH+12].

8.3.4 Zusammenfassung der Ergebnisse

Eine wesentliche Anforderung dieser Arbeit besteht darin, die als Prozess definierte fachliche Anwendungslogik für eine dynamische Verteilung der Prozessausführung nicht anpassen zu müssen, sondern Prozessmodell und Prozessinstanz zur Laufzeit unverändert auf ein anderes Prozessmanagementsystem zu übertragen. Die Untersuchung der drei standardisierten Prozessbeschreibungssprachen XPDL, WS-BPEL und BPMN hat gezeigt, dass das vorgestellte (abstrakte) Migrationsmodell für verschiedene Prozessbeschreibungssprachen anwendbar ist, jedoch für jede Prozessbeschreibungssprache eine individuelle Abbildung der Kontrollflusskonstrukte auf die Elemente des Migrationsmodells erforderlich ist. In Abhängigkeit der durch die Prozessbeschreibungssprache angebotenen Kontrollflussstrukturen müssen zudem teilweise Einschränkungen für die Migration von Prozessinstanzen hingenommen werden, sofern bestimmte Kontrollflussmuster im Prozessmodell spezifiziert sind. Hierbei sind insbesondere die verteilte parallele Ausführung von einzelnen Aktivitätsinstanzen (vgl. *For each* in Abschnitt 8.3.2), das Empfangen einer Nachricht als Antwort auf eine in einer vorausgegangenen Aktivität abgesetzten Anfrage (asynchrone Kommunikation auf Anwendungsebene, vgl. z. B. *Receive* in Abschnitt 8.3.2 oder *Receive Task* in Abschnitt 8.3.3) und die Verarbeitung komplexer Ereignisse über einen Gültigkeitsraum von mehreren Aktivitäten (vgl. z. B. *Intermediate boundary event* in Abschnitt 8.3.3) zu nennen.

Ein Vergleich mit der Verteilung derselben Kontrollflussstrukturen durch physische Partitionierung des Prozessmodells zeigt, dass die bei der Migration von Prozessinstanzen identifizierten Einschränkungen im Wesentlichen auch für diese Art der Verteilung vorliegen, so dass es durch das eingeschränkte Verteilungsspektrum im Fall der Migration in Bezug auf diese Elemente nicht zu einer Verschlechterung der Flexibilität kommt. Für die physische Partitionierung von Prozessen ist zudem für alle Verteilungsschritte die Integration weiterer Aktivitäten zur Steuerung der Verteilung notwendig. In einigen Fällen kann eine Verteilung des Prozessmodells durch Partitionierung nicht vorgenommen werden, ohne dabei *erhebliche* Änderungen an den zu verteilenden Kontrollflussstrukturen zu erfordern, welche teilweise sogar zu einer Auflösung der Kontrollflussstrukturen führen (wie zum Beispiel bei einer Verteilung innerhalb von Schleifendurchläufen). Da bestimmte Kontrollflussstrukturen durch physische Partitionierung des Prozessmodells in Folge nicht verteilt ausgeführt werden können, kann der Migration von Prozessinstanzen daher zum Teil sogar ein größeres Verteilungsspektrum zugesprochen werden als der physischen Partitionierung von Prozessen. Durch die Auswertung von Verzweigungsbedingungen und Datenabhängigkeiten zur Laufzeit des Prozesses können zudem für die individuelle Prozessinstanz unnötige Verteilungen von Prozesspartitionen zur Entwicklungszeit und die entsprechende Kommunikation zur Laufzeit vermieden werden. Allerdings entsteht durch die Migration von Prozessinstanzen auch ein Mehraufwand, da bei jedem Verteilungsschritt jeweils das gesamte Prozessmodell übertragen wird, während bei der Verteilung durch physische Partitionierung immer nur der jeweils relevante Teil des Prozessmodells an eine oder mehrere Ausführungseinheiten zugewiesen wird. Welche Verteilungsvariante vorteilhafter ist, ist daher vom Umfang des Prozessmodells, der Anzahl potentiell für die Ausführung von Prozesspartitionen in Frage kommenden Ausführungseinheiten, der Möglichkeit, die Granularität der Prozesspartitionen bereits im Vorfeld der Ausführung festzulegen und der Dynamik zur Laufzeit des Prozesses abhängig. Im Folgenden werden diese Parameter im Kontext verschiedener Anwendungsfälle und im Rahmen einer integrierten Flexibilitätsbetrachtung genauer untersucht.

8.4 Anwendungsszenarien

In Abschnitt 4.2 wurden verschiedene Anwendungsgebiete und Fallbeispiele mit ihren unterschiedlich hohen Anforderungen an eine dynamische Anpassung von verteilt ausgeführten Prozessen hinsichtlich Verteilung, Überwachung und Steuerung vorgestellt. In diesem Abschnitt wird gezeigt, wie die Anwendung der im Rahmen dieser Arbeit vorgeschlagenen Konzepte auf Basis ihrer prototypischen Implementierungen zur Unterstützung dieser Fallbeispiele und zur Erfüllung ihrer Anforderungen beitragen können. Dabei können Anwendungsgebiete mit einem besonders hohen Bedarf an Flexibilität, wie zum Beispiel die *kontextbasierte Kooperation* (vgl. Abschnitt 8.4.1) und *Process-Management-as-a-Service*-Szenarien (vgl. Abschnitt 8.4.2) durch

die Möglichkeiten zur individuellen Anpassung der Ausführung einzelner Prozessinstanzen besonders weitreichend unterstützt werden. Unabhängig vom Verteilungsmodell bestehen des weiteren Möglichkeiten zur Überwachung organisationsübergreifend ausgeführter Prozesse, welche am Beispiel eines Prozesses aus dem Projekt *eErasmus eHigher Education (eEH)* vorgestellt werden (vgl. Abschnitt 8.4.3). Die weiterführende Nutzung der ursprünglich für verteilt ausgeführte Prozesse konzipierten abstrakten Benutzungsschnittstellen auch für nicht-verteilte parallel auf heterogenen mobilen Endgeräten ausgeführte Prozessinstanzen wird am Beispiel eines Anwendungsfalls zur *prozessorientierten Datenerhebung* durch elektronische Fragebögen verdeutlicht (vgl. Abschnitt 8.4.4). Als Beispiel für eine Optimierung der Prozessausführung durch die Verteilung des Prozesses wird die Reduzierung der übertragenen Datenmenge aufgegriffen und anhand eines Anwendungsbeispiels zum Versenden und Kommentieren von Bilddaten diskutiert (vgl. Abschnitt 8.4.5). Da die in dieser Arbeit vorgestellten Konzepte zum großen Teil in mehreren Anwendungsbeispielen angewendet worden sind, wird in jedem Anwendungsszenario jeweils nur zu ausgewählten Komponenten eine genauere Betrachtung der Anwendbarkeit vorgenommen.

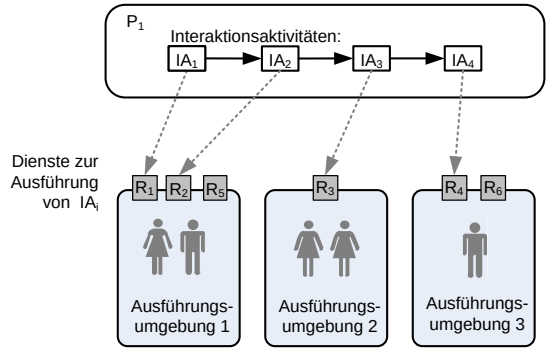
8.4.1 Weiterentwicklung der kontextbasierten Kooperation

Das Konzept der kontextbasierten Kooperation erlaubt die Ausführung von prozessorientierten Anwendungen über mehrere stationäre und mobile Ausführungssysteme, welche jeweils nur einen Teil der benötigten Anwendungsfunktionalitäten bereitstellen und zur Entwicklungszeit der prozessorientierten Anwendung nicht bekannt sind (vgl. Abschnitt 4.2.4). Die durch Migration von Prozessinstanzen verteilt auszuführenden Prozesse wurden hierzu bisher in der proprietären Prozessbeschreibungssprache DPDL ausgedrückt, für die bisher keine Unterstützung für eine graphische Modellierung oder für die Transformation von Prozessen anderer Prozessbeschreibungssprachen bereitgestellt wurde. Bereits bestehende Prozesse anderer Prozessbeschreibungssprachen, welche durch kontextbasierte Kooperation ausgeführt werden sollten, mussten daher zunächst manuell in DPDL-Repräsentation re-modelliert bzw. übersetzt werden. Insbesondere in Hinblick darauf, dass Entscheidungen über die Verteilung von Prozessen in Abhängigkeit des aktuellen Kontextes dynamisch auch zur Laufzeit erfolgen sollten, ist eine je nach verwendeter Prozessbeschreibungssprache mehr oder weniger aufwändige Re-Modellierung nicht vorteilhaft. Die Auswahl von Ausführungseinheiten, welche potentiell für eine kontextbasierte Kooperation zur Verfügung stehen, ist zudem durch die notwendige Unterstützung der proprietären Prozessbeschreibungssprache begrenzt. Es besteht dementsprechend bisher ein hoher Anfangsaufwand für potentielle Teilnehmer, da jeweils die Anschaffung und der Betrieb einer kompatiblen Prozess-Engine erforderlich ist, welche jedoch nur für Prozesse der kontextbasierten Kooperation, nicht aber für sonstige prozessorientierte Anwendungen des Teilnehmers genutzt werden kann [ZKL09b, ZKL09a].

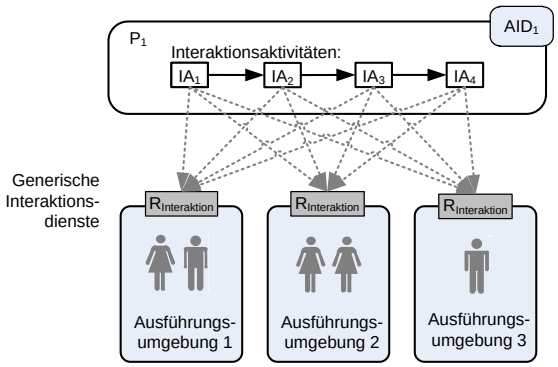
Durch die Anwendung der im Rahmen dieser Arbeit vorgeschlagenen Strategien zur Flexibilisierung der verteilten Prozessausführung ist das Konzept der kontextbasierten Kooperation weiterentwickelt worden. Zum einen können bereits bestehende und innerhalb einer lokalen Ausführungsumgebung eingesetzte Prozessmanagementsysteme durch die Bereitstellung der vorgeschlagenen Schnittstelle (mit einmaligem Entwicklungsaufwand) für eine kontextbasierte Kooperation nutzbar gemacht werden (vgl. Abschnitt 8.2). Zum anderen können auf diese Weise auch bereits bestehende Prozesse, welche zum Beispiel in einer standardisierten Prozessbeschreibungssprache ausgedrückt sind (vgl. Abschnitt 8.3), dynamisch im Rahmen einer kontextbasierten Kooperation ausgeführt werden. Dies gilt sowohl für bereits bestehende Prozessmodelle als für einzelne Prozessinstanzen, deren Ausführung bereits auf einer lokalen Prozess-Engine begonnen wurde. Durch die Möglichkeit zur Nutzung von standardisierten Prozessbeschreibungssprachen ist zudem in der Regel eine breite Auswahl an Modellierungswerkzeugen verfügbar, um die Erstellung von neuen prozessorientierten Anwendungen für die kontextbasierte Kooperation zu unterstützen. Durch die Weiterverwendung bestehender Infrastrukturen, der Übernahme von bereits bestehenden Prozessmodellen und der Möglichkeit zur Nutzung von standardisierten Prozessbeschreibungssprachen eröffnet sich somit ein größeres Potential an Teilnehmern, welche über das Konzept der kontextbasierten Kooperation Funktionalitäten bereitstellen können. Da die Maximierung der Anzahl potentieller Teilnehmer für die kontextbasierte Kooperationen einen wichtigen Faktor darstellt (vgl. [Kun08]), kann bereits durch die vorgeschlagene prozessbeschreibungssprachenunabhängige Weiterentwicklung des Migrationsmodells eine wesentliche Verbesserung erzielt werden.

Betrachtet man die inhaltliche Weiterentwicklung des Migrationsmodells, so ist mit der Identifikation nicht trennbarer Kontrollflussstrukturen, der Behandlung von Datenabhängigkeiten auf parallelen Kontrollflusspfaden und der potentiellen Migration von ereignisbasierten Gültigkeitsbereichen innerhalb von prozessorientierten Anwendungen der Handlungsspielraum für die Verteilung von Prozessen vergrößert und in Hinblick auf die korrekte Ausführung des Prozesses abgesichert worden. Die für eine verteilte Ausführung besonders relevante parallele Ausführung sowie die Ausschöpfung von Flexibilisierungsmöglichkeiten auf Anwendungsebene sind daher nun auch für die kontextbasierte Kooperation zugänglich und erlauben somit ein breiteres Spektrum an verteilt auszuführenden Anwendungen abzudecken.

Darüber hinaus war die kontextbasierte Kooperation bisher auf die reine Ausführung von Prozessen beschränkt. Eine weitergehende Überwachung der verteilten Ausführung, das Sammeln von Daten für eine spätere Auswertung oder die Rückführung der Prozessinstanz an den Initiator nach Abschluss der Prozessausführung wurden bislang nicht unterstützt. Die Defizite hinsichtlich Sicherheit auf Anwendungsebene in Verbindung mit zuverlässigen Mechanismen zur Protokollierung der Prozessausführung und einer nicht abstreitbaren Zurechnung von Ausführungsverantwortlichkeiten erschwerten bislang auch



(a) Interaktion über vordefinierte Dienste für die Interaktionsart der Interaktionsaktivität IA_i



(b) Interaktion über abstrakte Interaktionsbeschreibungen (AID) und generischen Interaktionsdienst

Abbildung 8.3: Flexibilitätsgewinn durch generische Interaktionsdienste: Mögliche Ausführungen der Interaktionsaktivitäten IA_i des Prozessmodells P_1 (ohne Berücksichtigung fachlicher bzw. personeller Einschränkungen)

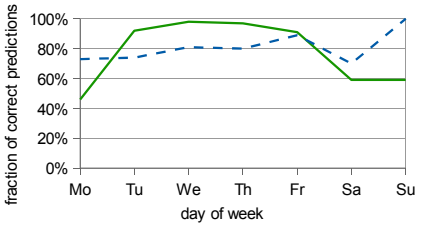
die Konzeption eines kommerziellen Einsatzes der kontextbasierten Kooperation [ZKL09b, ZKL09a]. Durch die Trennung von funktionaler Prozessbeschreibung und Verteilungsanweisungen ist durch das hier vorgestellte Migrationsmodell die Durchsetzung von grundlegenden Sicherheitseigenschaften vereinfacht worden (vgl. Abschnitt 6.2.7). Zusammen mit den durch die kontextbasierten Kooperation bereits bereitgestellten Sicherheitsmechanismen auf Kommunikationsebene ist daher nun auch die Vertraulichkeit und Integrität des Prozesses auf Anwendungsebene nach benutzerdefinierten Vorgaben direkt umsetzbar. Eine Überwachung des verteilt ausgeführten Prozesses erfolgt durch die Bereitstellung der beteiligten Prozessmanagementsysteme als verwaltbare Ressource mit einer (für mobile Systeme vereinfachten) WSDM-Schnittstelle und einem (optionalen) lokalen Managementdienst (vgl. auch [Böh09]). Ein auch auf die kontextbasierte Kooperation anwendbarer Ansatz auf Basis der vorgestellten Strategien wird in Abschnitt 8.4.2 vorgestellt.

Des weiteren konnten Benutzerinteraktionen im Rahmen der kontextbasierten Kooperation bislang nur über vordefinierte Ein- und Ausgabedienste erfolgen [Kun08], da die durchzuführende Interaktion mit den durch DPDL bereitgestellten Sprachelementen nicht ausgedrückt werden konnte. Die Wieder-

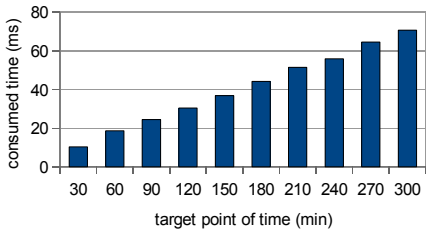
verwendung von spezialisierten Ein- und Ausgabediensten mit Benutzungsschnittstellen für die Durchführung alternativer Interaktionsaktivitäten ist jedoch im Allgemeinen stark beschränkt (vgl. Abschnitt 5.8.1). Daher musste bei der kontextbasierten Kooperation bisher für jede Aktivität mit Benutzerinteraktion ein eigener Dienst implementiert und dieser in angemessener Form für die jeweilige Ausführungseinheit bereitgestellt werden. Da die Art und der Inhalt der benötigten Interaktion nicht von den Ausführungseinheiten vorhergesehen werden kann und die zur Ausführungszeit zur Verfügung stehenden Ausführungseinheiten zur Entwicklungszeit des Prozesses nicht bekannt sind, war die Flexibilität der Verteilung bei der Durchführung von Benutzerinteraktionen bislang stark eingeschränkt. Durch die Möglichkeit zum Aufruf eines generischen Interaktionsdienstes, welcher abstrakte Benutzungsschnittstellen als Ergänzung zur migrierten Prozessbeschreibung verarbeiten kann (vgl. Abschnitt 6.4), ist die individuelle Bereitstellung von Diensten zur Benutzerinteraktion nicht mehr notwendig. Folglich besitzt nun jede Ausführungseinheit für die Anbindung menschlicher Prozessteilnehmer maximal einen *generischen Interaktionsdienst*, welcher jeweils auf die von der Ausführungseinheit unterstützte Benutzungsschnittstelle zugeschnitten ist, oder ruft alternativ einen zentralen Interaktionsdienst auf, welcher die benötigte Benutzungsschnittstelle dynamisch erzeugen kann. Die Anzahl der für eine beliebige Benutzerinteraktion erforderlichen Dienste ist somit stark reduziert und zugleich die freie Auswahl aller Ausführungseinheiten mit generischem Interaktionsdienst als mögliche Zielausführungseinheiten des interaktiven Prozesses ermöglicht worden [ZVBK09]. Abbildung 8.3 veranschaulicht diesen Flexibilitätsgewinn graphisch.

Während für die Auswahl von funktionalen Diensten bei der kontextbasierten Kooperation nicht-funktionale Parameter angegeben werden können (vgl. [Kun08, ZKL09b, ZKL09a]), erfolgte die Migration des Prozesses selbst bislang relativ ungerichtet. Im einfachsten Fall wird eine zum Routing von Paketen nach der *Hot-Potatoe*-Technik analoge Strategie angewendet, welche einen Prozess auf eine beliebige andere Ausführungseinheit migriert, sobald die anstehende Aktivität von der lokalen Ausführungseinheit nicht erfolgreich ausgeführt werden kann [ZKL09b, ZKL09a]. Durch die Bereitstellung der Aus- bzw. Weiterführung von Prozessinstanzen als Dienst kann die bereits bestehende Funktionalität zur Auswahl von Diensten auch für die Migration von Prozessen genutzt werden. Die Prozesse können somit gezielt einer Ausführungseinheit zugewiesen werden, welche z. B. die nächste Aktivität oder möglichst viele der aktuell anstehenden Aktivitäten ausführen kann. Durch die zusätzliche Prognose von Dienstverfügbarkeiten (vgl. Abschnitt 6.5) kann die Migrationsentscheidung auch dann verbessert werden, wenn aktuell keine Ausführungseinheiten in der direkten Umgebung des Prozesses aufzufinden sind, welche die Prozessausführung direkt voranbringen können.

Die Ergebnisse einer exemplarischen Anwendung der strukturierten Kontextdatenprognose zur Verbesserung der Migrationsentscheidungen für die kontextbasierte Kooperation auf Basis der Prognose von Dienst-



(a) Anteil korrekter Prognosen pro Tag zur Verfügbarkeit eines Druckdienstes (*grüne Linie*) und eines Dienstes zum Dateiaustausch (*blaue Linie*)



(b) Dauer einer Prognose der Dienstverfügbarkeit mit 70 Prognoserunden, jede Prognosedauer gemittelt über 176 Prognosen

Abbildung 8.4: Evaluation des Ansatzes der strukturierten Kontextdatenprognose am Beispiel der Prognose zur Dienstverfügbarkeit (Mei09, MZL10, ZML11)

verfügbarkeiten sind Abbildung 8.4 zu entnehmen. Hierbei wurde in einem Zeitraum von sieben Tagen (Montag bis Sonntag) bei Feststellen von Dienstverfügbarkeiten die Art des verfügbaren Dienstes als UUID, die Netzgröße, die Tageszeit und die geographische Position des Benutzers mit der mobilen Ausführungseinheit erhoben. Zu Testzwecken wurden zwei Dienste mit unterschiedlichen Charakteristika in der Verfügbarkeit gewählt. Ein stationärer Druckdienst (*Printer Service*) ist regelmäßig am Arbeitsplatz des Benutzers verfügbar (d. h. tagsüber Montags bis Freitags). Ein Dienst zum Dateiaustausch (*File Sharing Service*) wird unregelmäßig durch verschiedene andere Benutzer in der mobilen Umgebung angeboten und ist daher weniger häufig verfügbar. Die dargestellten Ergebnisse zur erreichten Genauigkeit (vgl. Abbildung 8.4a) begründen sich auf Prognosen über die Verfügbarkeit der beiden Dienste als boolesche Variable zu einem bestimmten Zeitpunkt in der Zukunft und dem Vergleich der Prognose mit deren tatsächlicher Verfügbarkeit. Da der Dateiaustauschdienst meistens nicht verfügbar ist, kann diese Regelmäßigkeit schnell „erlernt“ werden und bereits die ersten Prognosen zeigen weitestgehend korrekte Ergebnisse. Im Fall des Druckdienstes kann anhand der Abhängigkeit von der Tageszeit und dem Ort bereits am zweiten Tag korrekt vorausgesagt werden dass der Dienst erreichbar ist, wenn der Benutzer sich mit dem mobilen Endgerät am Arbeitsplatz befindet [Mei09, MZL10, ZML11].

Durch den Einsatz der gewählten Prognosemethoden (Wahrscheinlichkeitstabellen und lineare Regression) ist der Bedarf an Speicherplatz für die dargestellten Prognosen relativ begrenzt, da nicht die Rohdaten des Kontextmanagementsystems, sondern nur das hiervon abgeleitete Wissen gespeichert wird. Dieses wird in Wahrscheinlichkeitstabellen erfasst, so dass der Speicherplatzbedarf lediglich von der Anzahl der Variablen und der Anzahl ihrer jeweils möglichen Werte (bzw. Wertebereiche) abhängig ist. Es müssen daher für die oben dargestellte einfache Prognose von Dienstverfügbarkeiten nur etwa 20 KB Speicherplatz aufgewendet werden. Die für die Ableitung von neuem Wissen erforderliche Rechenleistung ist mit deutlich unter 1% CPU-Auslastung (bei einem Notebook mit 1,5 GHz und Pentium M Prozessor) zu

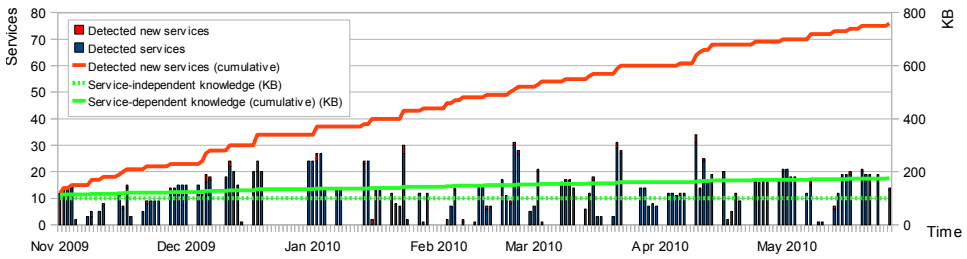


Abbildung 8.5: Anwendung des Prognosedienstes zur Vorhersage von Dienstverfügbarkeiten über sechs Monate (ZML11)

vernachlässigen. Abbildung 8.4b zeigt, dass die Zeitdauer einer Prognose (neben der Anzahl an Variablen im Prognosemodell) linear von der Anzahl der Prognoserunden und der Entfernung vom aktuellen Zeitpunkt bis zu dem Zeitpunkt, für den die Prognose durchgeführt werden soll, abhängt. Dabei wird ersichtlich, dass auch relativ komplexe Prognosen in deutlich weniger als einer Sekunde durchgeführt werden können und somit Prognosen auch für Endgeräte mit geringerer Rechenleistung (wie zum Beispiel Smartphones) in annehmbarer Zeit durchgeführt werden können (vgl. [Mei09, MZL10, ZML11] zu Details).

Verbesserte Ergebnisse hinsichtlich der Genauigkeit können erreicht werden, indem Prognosemethoden mit der geringsten Unsicherheit automatisch bevorzugt werden. Zum Beispiel ist die Netzgröße (d. h. insbesondere die Anbindung zum Internet) zur Ableitung der Verfügbarkeit des Dateiaustauschdienstes ein wichtigerer Aspekt als z. B. die Tageszeit oder die geographische Position. Im Fall des Druckdienstes ist eine Verbesserung insbesondere durch die Integration des Wochentages in das Prognosemodell zu erreichen, da in diesem Fall auch am Wochenende korrekte Ergebnisse geliefert werden [Mei09, MZL10, ZML11]. Abbildung 8.5 zeigt die Anwendung des Prognosedienstes mit einem erweiterten Prognosemodell über sechs Monate, wobei nun eine zufällige Menge von beliebigen Diensten an das Prognosesystem gemeldet wird, welche entweder bisher unbekannt sein können (rote Blöcke) oder durch ihre UUID als bereits bekannte (und somit wiederholt verfügbare) Dienste erkannt werden (blaue Blöcke). Wie in Abschnitt 6.5.5 dargestellt wurde, besteht die Möglichkeit, das bereits abgeleitete Wissen innerhalb des Prognosemodells für verschiedene Dienste wiederzuverwenden. Der Speicherplatz für das dienstunabhängige Wissen ist daher von der Menge der Dienste unbeeinflusst (gestrichelte grüne Linie). Der erforderliche Speicherplatz für das dienstabhängige Wissen (durchgezogene grüne Linie) und somit für das Sammeln von Wissen zu neu erkannten Diensten steigt daher im Verhältnis zu der Menge der insgesamt verwalteten Dienste (rote Linie) weitaus weniger stark an. Älteres Wissen über nur unregelmäßig verfügbare Dienste führt zudem in der Regel zu nicht verwertbaren Prognosen. Zum Beispiel ist das einmalige Auftreten eines Dienstes im November, für welchen in Folge keine weiteren Verfügbarkeiten gemeldet wurden, für eine Prognose im Mai nicht mehr

relevant. Durch das regelmäßige Löschen solcher nicht mehr nützlichen Daten kann daher insgesamt auch das dienstabhängige Wissen auf einen nahezu konstanten Speicherplatzbedarf begrenzt werden. Die Vorhersage von Dienstverfügbarkeiten kann somit auch für eine relativ große Menge von verschiedenartigen Diensten bzw Dienstklassen effizient eingesetzt werden und bietet somit für die kontextbasierte Kooperation eine geeignete Unterstützung [ZML11]. Weitere Details zur Evaluation der strukturierten Kontextdatenprognose sind vertiefenden Veröffentlichungen [Mei09, MZL10, ZML11] zu entnehmen.

8.4.2 (Distributed) Process-Management-as-a-Service

Wie in Abschnitt 4.2.6 motiviert wurde, sind die Anschaffung und der Betrieb eines Prozessmanagementsystems in der Regel erst ab an einer gewissen Anzahl von (teil-)automatisiert auszuführenden Prozessen wirtschaftlich. Mit dieser Anzahl an Prozessen ist – insbesondere bei leistungsfähigen Systemen – in der Regel nicht gleichzeitig auch die Auslastungsgrenze der Ausführungsumgebung erreicht, und die durchschnittliche Auslastung von mobilen Ausführungssystemen muss aufgrund ihrer zumeist hohen Spezialisierung noch deutlich geringer eingeschätzt werden. Können freie Kapazitäten von Prozessmanagementsystemen inklusive der zugeordneten lokalen Ressourcen flexibel auch an externe Prozessinitiatoren freigegeben bzw. veräußert werden, ergibt sich somit für mehrere Organisationen oder Organisationseinheiten (analog zu einer organisierten gemeinsamen Nutzung von z. B. Produktionsanlagen) die Möglichkeit zu einer gemeinschaftlichen Nutzung von Prozessmanagementsystemen. Ergänzend hierzu besteht bereits ein Markt für das kommerzielle Anbieten von Leistungen des Prozessmanagements, wo Prozessmanagementsysteme nach einem Pay-per-Use- oder Mietmodell gezielt für Konsumenten mit nur unregelmäßiger oder gelegentlicher Nutzung bereitgestellt werden [FG08]. Dynamisch verteilt auszuführende Prozesse konnten auf diese Weise bisher jedoch nicht unterstützt werden. Auf Basis der in dieser Arbeit vorgeschlagenen Strategien wird in diesem Abschnitt eine Architektur für ein *verteiltes Process-Management-as-a-Service*-Szenario [ZL10] vorgestellt.

In einem verteilten *Process-Management-as-a-Service*-System (kurz *PMaaS*-System) stellen alle (kommerziellen oder privaten) Anbieter stationärer oder mobiler Ressourcen die Möglichkeit bereit, eine dynamisch zu bestimmende Prozesspartition eines Prozessmodells in einer unterstützten Prozessbeschreibungssprache durch ihre lokale Prozess-Engine ausführen zu lassen. Für die dynamische Verteilung des Prozesses wird die in den Abschnitten 6.2 und 7.5 vorgeschlagene Migration von Prozessinstanzen zugrunde gelegt. Ein *PMaaS-Anbieter* wird daher als mobile oder stationäre Ausführungseinheit definiert, welche eine Schnittstelle zum Deployment von Prozessmodellen und zur Übernahme von Prozessinstanzdaten nach dem vorgeschlagenen Migrationsmodell besitzt und diese Schnittstellen potentiellen *PMaaS-Konsumenten* zur Verfügung stellt, um deren Prozesse bzw. Prozesspartitionen aus- bzw.

fortzuführen. Dabei können die genannten Rollen wie in einem Peer-to-Peer-System auch dynamisch wechseln, d. h. ein PMaaS-Anbieter kann auch als Konsument auftreten, falls er die PMaaS-Funktionalitäten anderer Ausführungseinheiten zur (partiellen) Ausführung seiner Prozesse in Anspruch nimmt. Es können abhängig von der zur Verfügung stehenden Kapazität und Leistungsfähigkeit der Ausführungsumgebung drei verschiedene Arten von PMaaS-Anbietern unterschieden werden [ZL10]:

- **Vollwertige Anbieter** stellen die Möglichkeit zur Ausführung von kompatiblen Prozessen bzw. Prozesspartitionen bereit. Hierzu ist zumindest eine gekapselte Prozess-Engine erforderlich.
- **Back-up-Anbieter** überwachen die Ausführung von Prozessen durch einen vollwertigen PMaaS-Anbieter. Hierzu ist eine Anwendung zur Verarbeitung der bei der Überwachung anfallenden Informationen und ggf. die Einleitung von Reaktionen auf erkannte Abweichungen erforderlich.
- **Proxy-Anbieter** stellen einfache Träger eines aus Prozessmodell und Prozessinstanzdaten bestehenden Prozesses dar, um diese an einen oder mehrere vorgegebene vollwertige Anbieter weiterzuvermitteln. Hierzu ist lediglich Speicherplatz für die Prozesse und ein Zugang zu den Schnittstellen der relevanten vollwertigen PMaaS-Anbieter erforderlich.

Für ein kommerzielles Szenario besteht eine wichtige Anforderung in der Möglichkeit zur Abrechnung und Bezahlung der erbrachten Leistungen (vgl. [ZKL09b, ZKL09a]). Die verteilte PMaaS-Systemarchitektur für ein kommerzielles Szenario besteht daher zusätzlich zu den genannten Akteuren aus mindestens einem *zentralen PMaaS-Verwalter*, welcher für die Verwaltung und Abrechnung der verteilt ausgeführten Prozesse verantwortlich ist. Abbildung 8.6 zeigt einen Vorschlag für ein mehrstufiges Konzept zur kommerziellen Integration von PMaaS-Anbietern und -Konsumenten. Die erste Phase kann dabei als eine Art Vorbereitungsphase verstanden werden, um die einmalige Registrierung von Anbietern und Konsumenten zu ermöglichen und für jeden neu registrierten Teilnehmer ein Konto anzulegen (vgl. Phase 1a in Abbildung 8.6). Zum Zeitpunkt der Registrierung können dabei bei Bedarf außerdem die öffentlichen Schlüssel und Identitätsnachweise der Teilnehmer ausgetauscht werden, so dass diese bei der späteren Verwendung von Sicherheitsmechanismen und zur Nachvollziehbarkeit der Prozessausführungen zur Verfügung stehen (vgl. Abschnitt 6.2.7). Um eine korrekte Abrechnung der Prozessausführung auch bei einer Offline-Ausführung des Prozesses und in mobilen Ad-hoc-Netzwerken zu ermöglichen, können potentielle PMaaS-Konsumenten an dieser Stelle eine frei wählbare Anzahl an digitalen Münzen (*Digital Coins*) erwerben, welche nach dem Prepaid-Prinzip einen bestimmten Wert repräsentieren und zur Vorbeugung von Fälschungen eine Signatur des ausgegebenen zentralen PMaaS-Verwalters aufweisen [ZL10] (vgl. Phase 1b in Abbildung 8.6). Das Erwerben von digitalen Münzen ist nur notwendig, wenn die PMaaS-Funktionalitäten ohne direkte Netzwerkverbindung zum

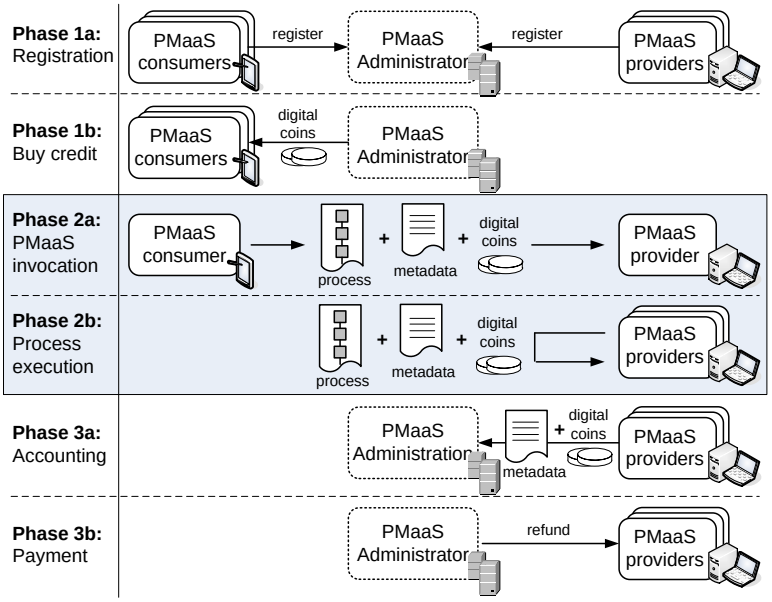


Abbildung 8.6: Überblick über das Konzept zum verteilten Process-Management-as-a-Service (kommerzielles Szenario) (ZL10)

zentralen PMaaS-Verwalter per Prepaid-Prinzip und dadurch gegenüber den einzelnen PMaaS-Anbietern anonym in Anspruch genommen werden sollen (vgl. [Gün09]).

Die zweite Phase umfasst die Interaktionen der registrierten Teilnehmer zur Laufzeit von (verteilt) ausgeführten Prozessen. Die Prozesmodell- und -instanzdaten werden dabei zusammen mit integrierten benutzerdefinierten Anweisungen zur Verteilung, Überwachung und Steuerung des Prozesses sowie der relevanten Menge an digitalen Münzen an einen ausgewählten PMaaS-Anbieter übermittelt (vgl. Phase 2a in Abbildung 8.6). Durch den Erhalt der vom zentralen PMaaS-Verwalter signierten Münzen kann der ausgewählte PMaaS-Anbieter sicherstellen, dass sein Beitrag an der Ausführung des Prozesses nach Abschluss der Leistung vergütet wird. Kann der PMaaS-Anbieter den Prozess nicht vollständig ausführen, kann er (unter Berücksichtigung der benutzerdefinierten Rahmenbedingungen für die Ausführung des Prozesses) gegenüber anderen PMaaS-Anbietern als Konsument auftreten und die verbleibende Prozessausführung erneut delegieren (vgl. Phase 2b in Abbildung 8.6). Der PMaaS-Konsument lässt sich zudem die Übernahme des Prozesses durch den gewählten PMaaS-Anbieter quittieren und die entstehende Zuständigkeitskette wird innerhalb der Log-Datei des Prozesses protokolliert. Falls Unstimmigkeiten auftreten, kann die im Prozessprotokoll enthaltene Zuständigkeitskette nach Abschluss der Ausführung mit den Protokolldateien der einzelnen Teilnehmer verglichen werden [ZL10].

Die dritte und letzte Phase beinhaltet die optionale Übergabe der individuellen Protokolldateien, der Log-Datei des Prozesses und der digitalen Münzen an den zentralen PMaaS-Verwalter (vgl. Phase 3a in Abbildung 8.6). Bei erfolgter

Ausführung der in der Prozessbeschreibung enthaltenen Aktivitäten wird der Beitrag der einzelnen PMaaS-Anbieter durch eine Gutschrift auf deren jeweiliges Konto vergütet. Schlägt die Ausführung fehl, erfolgt dementsprechend keine oder nur eine teilweise Abrechnung [ZL10] (vgl. Phase 3b in Abbildung 8.6).

Die in dieser Arbeit vorgeschlagenen Strategien zur Flexibilisierung der verteilten Prozessausführung unterstützen insbesondere die zweite Phase des dargestellten Szenarios. Dabei werden die Konzepte zur Spezifikation und Auswertung von Verteilungsstrategien (*Assignment Strategies*), die Spezifikation von Management-Regeln (*Management Rules*) und deren lokale Ableitung von Aktionen zur Steuerung der Prozessausführung sowie die vorgeschlagene Spezifikation und Durchsetzung von Sicherheitseigenschaften auf Prozessebene (*Security Policies*) integriert. Ein Überblick über die – zusätzlich zum funktionalen Prozessmodell – verwendeten Metadaten für das verteilte PMaas-Szenario ist in Abbildung 8.7 dargestellt. Um die Elemente des Prozessmodells aus den Metadaten zu referenzieren, wird die Konformität der Prozessbeschreibung zu dem in Abschnitt 6.2.2 vorgeschlagenen minimalen Prozessmetamodell vorausgesetzt. Die Instanzdaten und die Verteilungsstrategien für eine dynamische Verteilung der Prozesse werden durch die in Abschnitt 7.5.1 definierte Struktur für Migrationsdaten ausgedrückt. Als weitere Ergänzung des dort vorgeschlagenen Prologs kann optional eine Priorität (*Priority*) der Prozessausführung angegeben werden. Bei Angabe einer hohen Priorität wird der Prozess bei einer vollständigen Auslastung der bevorzugten Prozessmanagementsysteme nicht in eine Warteschlange eingereiht, sondern durch eine temporäre Aussetzung eines niedriger priorisierten Prozesses sofort zur Ausführung gebracht. Für das Ausführen von Prozessen mit einer hohen Priorität kann dabei abhängig von den Ausführungsrichtlinien der PMaaS-Anbieter natürlich auch ein höherer Preis verlangt werden [ZL10].

Um die durch die Delegation verlorengegangene Kontrolle über die Prozessausführung möglichst zu kompensieren, können optional auf Basis des Modells des Prozessmanagementsystems als verwaltbare Ressource zur Laufzeit dynamisch weitere Anweisungen abgesetzt werden, um die entfernte Ausführung des Prozesses zu überwachen und ggf. anzupassen. Da nicht jedem Teilnehmer des PMaaS-Systems automatisch Zugriff auf die bereitstehenden Managementfunktionalitäten gewährt werden soll, kann in den Management-Metadaten eine Liste von Teilnehmern (*Authorized Participant*) mit den ihnen erlaubten Managementaktionen und den zum Abonnement freigegebenen Ereignissen zu der betreffenden Prozessinstanz vermerkt werden (*Allowed management actions and events*). Desweiteren kann definiert werden, welchen PMaaS-Teilnehmern die Ergebnisse der Prozessausführung in Form der Protokolldateien und der finalen Prozessinstanzdaten zugestellt werden sollen (*Receiver*). In der Regel werden die zuvor genannten Funktionalitäten nur für den PMaaS-Konsumenten freigeschaltet werden, welcher die Prozessausführung initiiert hat. Es besteht aber durchaus die Möglichkeit, auch die Rolle der überwachenden Partei bzw. der Partei, welche die Ergebnisses

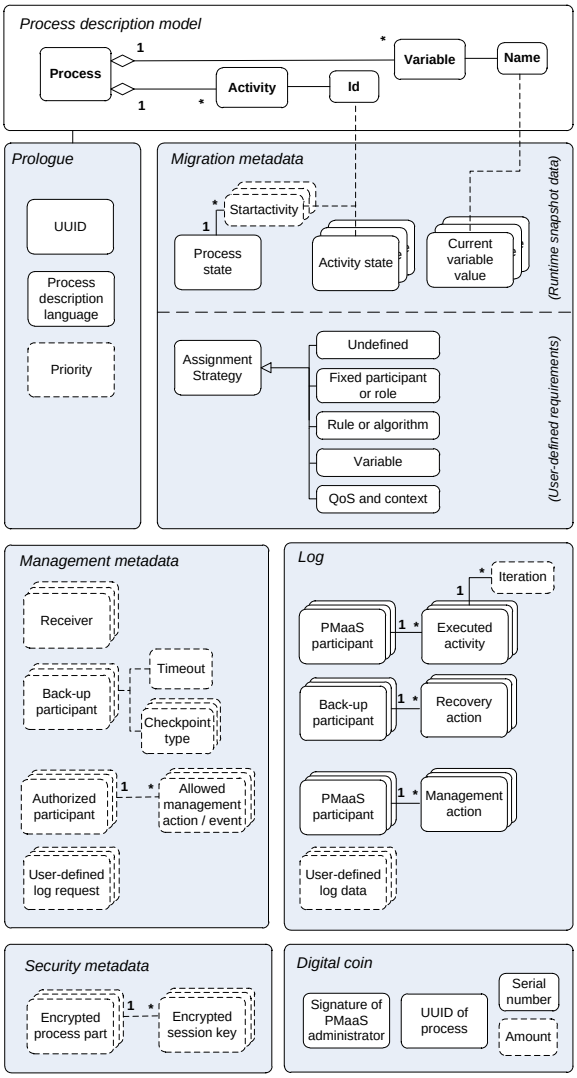


Abbildung 8.7: Gruppierung von PMaaS-Metadaten (vereinfacht) (ZL10)

der Prozessausführung erhält, zu delegieren oder durch multiple Teilnehmer des PMaaS-Systems wahrzunehmen bzw. wahrnehmen zu lassen. Dies ist insbesondere vorteilhaft, wenn es sich auch auf Anwendungsebene um eine kollaborative Prozessausführung durch mehrere gleichberechtigte Teilnehmer handelt, welche ein gemeinsames Interesse an dem ausgeführten Prozess und dessen Ergebnissen haben [ZKL09b, ZKL09a, ZL10].

Eine weitere Möglichkeit der Überwachung besteht in der Inanspruchnahme von Back-up-Teilnehmern, welche die Prozessausführung anstelle des PMaaS-Konsumenten kontrollieren. Diese Art der Überwachung ist insbesondere bei einer Prozessausführung in lokalen Netzwerken, wie etwa bei der (partiellen) Ausführung durch mobile Endgeräte, vorteilhaft. Werden Back-up-Teilnehmer spezifiziert, so darf der Prozess nur dann in einer Prozessausführungsumgebung ausgeführt werden, wenn dort die vorgeschriebene Anzahl von Back-up-Teilnehmern zur Verfügung steht, um den aktuellen Zu-

stand der Prozessausführung zu festgelegten Arten von Meilensteinen (*Checkpoint Types*) zu sichern und bei Verlust des Kontakts zu der zu überwachenden Ausführungseinheit nach einem *Timeout* die Prozessausführung am letzten Sicherungspunkt wieder aufzusetzen. Eine verteilte Infrastruktur für eine solche Überwachung mit Vermeidung von Duplikaten der Prozessausführung ist der Arbeit von BÖHLING [Böh09] zu entnehmen. Die an den Sicherungspunkten erhobenen Daten können dabei vollständig durch eine Momentaufnahme der Migrationsdaten erfasst werden. Die notwendigen Schnittstellen für die benötigten Managementoperationen werden durch das Prozessmanagementsystem als verwaltbare Ressource bereitgestellt (vgl. [Böh09, ZKL09b, ZKL09a, ZL10]).

Schließlich wird durch die Management-Metadaten festgelegt, welche Informationen der Prozessausführung erhoben und in den Log-Daten der Prozessinstanz gespeichert werden sollen. Für die Abrechnung der erbrachten Leistungen durch die PMaaS-Anbieter werden die an der Prozessausführung aktiv beteiligten Teilnehmer sowie ihre Zuordnung zu den Aktivitäten, für die sie während der Ausführung verantwortlich waren, protokolliert (*Executed Activity*). Werden Aktivitäten mehrmals ausgeführt (z. B. innerhalb einer Schleife oder aufgrund eines Rollbacks), so wird zudem festgehalten, welcher Teilnehmer für welche *Iteration* dieser Aktivität verantwortlich war. Die Übernahme des Prozesses durch Proxy-Teilnehmer wird ohne Zuordnung zu Aktivitäten protokolliert. Wurden während der Ausführung Managementfunktionalitäten zur Überwachung und Steuerung der Prozessausführung in Anspruch genommen, so werden der zugreifende Teilnehmer, die Zugriffszeit und die durchgeführte Managementaktion erfasst. Die Überwachung der Prozessausführung und ggf. das Eingreifen in die Prozessausführung durch einen Back-up-Teilnehmer werden gesondert protokolliert (*Recovery Action*). Zusätzlich zu diesen bereits automatisch erhobenen Daten kann der PMaaS-Konsument auf Basis des Modells der verwaltbaren Ressource weitere eigene Anforderungen an die Sammlung von Ausführungsinformationen hinzufügen (*User-defined log requests* bzw. *User-defined log data*). Beispiele sind das Sammeln von Informationen zur Ausführungsdauer einzelner Aktivitäten oder zu geographischen Positionen der Ausführung [ZL10].

Für die Abrechnung der erbrachten Leistungen kann zur Laufzeit eine Kommunikation mit dem zentralen PMaaS-Verwalter erfolgen oder das hier vorgeschlagene Konzept der digitalen Münzen eingesetzt werden, welche auch das anonyme Nutzen von PMaaS-Funktionalitäten in gänzlich oder teilweise mobilen Umgebungen ohne dauerhaften Netzwerkanbindung an den zentralen PMaaS-Verwalter unterstützt. Die digitalen Münzen enthalten hierzu initial die Signatur des PMaaS-Verwalters zum Beweis ihrer Echtheit und optional einen Betrag, welcher durch die Münze repräsentiert wird. Durch die Ergänzung der UUID der Prozessinstanz, welche auch im Prolog der Migrationsdaten enthalten ist, wird die Münze zur Laufzeit an eine konkrete Prozessausführung gekoppelt und die Münze anschließend mit der Signatur des PMaaS-Konsumenten versehen. Die Münze kann somit nicht durch den

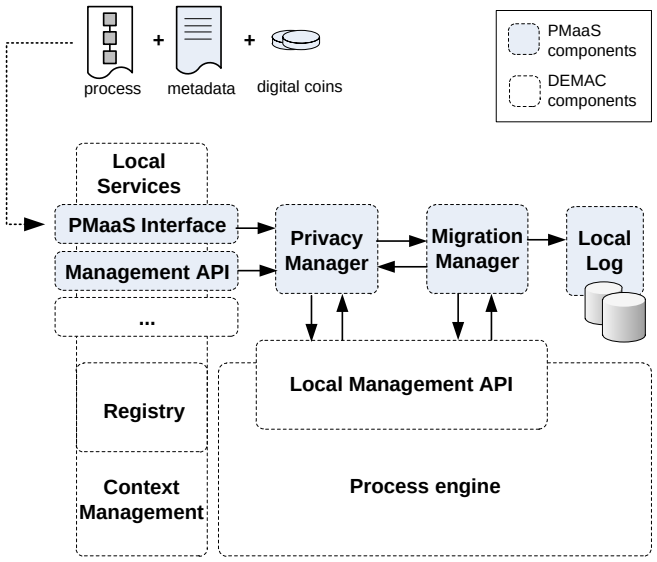


Abbildung 8.8: Architektur eines PMaaS-Anbieters (Übersicht) (ZL10)

PMaaS-Anbieter „gestohlen“ und für eine andere Prozessausführung verwendet werden (*One-Way-Coin*). Um zu verhindern, dass der PMaaS-Konsument seine Münzen mehrfach einsetzt, existiert des weiteren eine Seriennummer, welche nur bei Mißbrauch über das Konzept von *Gruppensignaturen* (vgl. [HT05]) eine Zuordnung zum PMaaS-Konsumenten erlaubt [Gün09, ZL10].

Abbildung 8.8 zeigt die Weiterentwicklung eines Teilnehmers der kontextbasierten Kooperation zu einem Anbieter in einer verteilten *Process-Management-as-a-Service*-Infrastruktur. Dabei wurden die wie in den Abschnitten 8.2.1 und 8.2.2 beschrieben erweiterten Prozess-Engines und die weiteren bestehenden Komponenten zur Realisierung der kontextbasierten Kooperation (insbesondere das angegliederte Kontextmanagementsystem und der dezentrale Verzeichnisdienst) zugrunde gelegt (vgl. [KZTL08]). Jeder vollwertige Anbieter stellt dabei eine leicht erweiterte Form der in Abschnitt 7.5.1 vorgestellten PMaaS-Schnittstelle und die Management-Schnittstelle der verwaltbaren Ressource auf Basis von WSDM zu Verfügung. Über die PMaaS-Schnittstelle werden Prozessmodell, Metadatendokument und ggf. digitale Münzen entgegen genommen und der Erhalt durch eine digital signierte Quittung bestätigt. Um auch auf bestimmte Funktionalitäten spezialisierte mobile Endgeräte zu integrieren, wird eine leichtgewichtige Variante der WSDM-Schnittstelle über die in Abschnitt 7.2 vorgestellte mobile Web-Service-Architektur verwendet.

Proxy-Teilnehmer benötigen lediglich eine kompatible Schnittstelle zum Aufruf der PMaaS-Funktionalitäten und einen Teil des Migrationsmanagers, um etwaige Anweisungen zur Weiterleitung des Prozesses interpretieren und ausführen zu können. Es werden durch sie nur die im Prolog der Migrationsdaten enthaltenen Daten ausgewertet, während der gesamte Rest ignoriert und lediglich lokal zwischengespeichert wird. Dasselbe gilt für die Back-up-

PMaaS-Konsumenten:	
Initiator der Prozessausführung:	
Unterstützte Plattformen:	J2ME MIDP/PP, Java SE
Mindestens zu unterstützende Protokolle:	TCP, HTTP oder DEMAC Overlay-Protokoll; ASN.1 oder SOAP; WSDL
PMaaS-Komponenten:	Client-Anwendung
Umfang des Softwarepakets (JAR):	$<$ 100 KByte
PMaaS-Anbieter:	
Proxy-Teilnehmer:	
Unterstützte Plattformen:	J2ME MIDP/PP, Java SE
PMaaS-Komponenten:	MPaaS-Schnittstelle, tlw. Migrationsmanager
Umfang des Softwarepakets (JAR):	$<$ 100 KByte (ohne Module zur Migrationsentscheidung)
Back-up-Teilnehmer:	
Unterstützte Plattformen:	J2ME MIDP/PP, Java SE
Mindestens zu unterstützende Protokolle:	kompatibel zu prozessausführender Partei
PMaaS-Komponenten:	Management-Client
Umfang des Softwarepakets (JAR):	$<$ 300 KByte
Vollwertiger MPaaS-Anbieter:	
Unterstützte Plattformen:	J2ME PP, Java SE
Unterstützte Prozessbeschreibungssprachen:	XPDL, WS-BPEL, DPDL
PMaaS-Komponenten:	alle
Umfang des Softwarepakets (JAR):	$<$ 3 MByte (mit jeweils einer der in den Abschnitten 5.5.5 und 8.2.2 genannten leichtgewichtigen Prozess-Engines)

Tabelle 8.6: Eigenschaften der prototypischen Implementierung der PMaaS-Teilnehmer (ZL10)

Teilnehmer, welche jedoch darüber hinaus eine Client-Anwendung zum Aufruf der Managementoperationen und zum Anlegen von Sicherungspunkten besitzen müssen. Der PMaaS-Konsument besteht schließlich im einfachsten Fall aus einer Client-Anwendung zum Aufruf der MPaaS-Schnittstelle und zur Entgegennahme etwaiger Ergebnisse der Prozessausführung. Er kann analog zum Back-up-Teilnehmer um Anwendungsfunktionalität zur Überwachung erweitert werden. Um eine (verteilte) Prozessausführung zu initiieren, muss der PMaaS-Konsument mindestens einen Proxy-Teilnehmer finden, welcher über eine kompatible Kommunikationsinfrastruktur zum Aufruf des PMaaS-Dienstes verfügt [ZL10]. Die wesentliche Konfiguration der prototypischen Implementierung und ihre Eigenschaften sind in Tabelle 8.6 zusammengefasst. Anhand des Umfangs der Softwarepakete für die einzelnen Plattformen zeigt sich zudem eine gute Anwendbarkeit für verschieden leistungsfähige Endgeräte.

Das verteilte *Process-Management-as-a-Service*-System skaliert insbesondere mit der Anzahl an Teilnehmern, welche sowohl als Anbieter als auch als Konsumenten auftreten. Treten verstärkt Anbieter auf, wird eine größere Menge an Ressourcen zu unter Umständen günstigeren Preisen angeboten, wodurch potentielle Konsumenten stärker angezogen werden. Eine größere Menge an Konsumenten verspricht für potentielle Anbieter eine größere Aussicht auf Profit, so dass angenommen werden kann, dass damit auch die Bereit-

schaft zur Bereitstellung von bislang nicht vollständig ausgenutzten Ressourcen steigt [ZL10]. Können durch die Bereitstellung der Funktionalität einer (dynamischen) Prozessausführung neben der Erzielung von Synergieeffekten auch Einnahmen erzielt werden, wirkt sich dies positiv auf das Kosten-Nutzen-Verhältnis des einmaligen Anpassungsaufwands zur Einrichtung der vorgeschlagenen Middleware-Komponenten aus.

8.4.3 Überwachung organisationsübergreifender Prozesse

Die Bereitstellung von Prozessmanagementsystemen als verwaltbare Ressource auf Basis von WSDM (vgl. Abschnitte 6.3 und 7.3) erlaubt eine flexible Anwendung von Überwachungs- und Steuerungsmaßnahmen für verteilt oder entfernt ausgeführte Prozesse bzw. Prozesspartitionen auch unabhängig vom zugrunde liegenden Verteilungsmodell. In diesem Abschnitt wird die Anwendung der in dieser Arbeit vorgeschlagenen Strategie für eine nicht-invasive Überwachung und Steuerung von Prozessen am Beispiel des in Abschnitt 4.2.2 vorgestellten Fallbeispiels aus dem Projekt *eErasmus eHigher Education (eEH)* vorgestellt. Die wesentlichen Beobachtungen hinsichtlich Flexibilität und Aufwand werden dabei mit zwei bestehenden Ansätzen zur ausschließlich ereignisbasierten (nicht-invasiven) Überwachung von Prozessen (vgl. z. B. [vA09]) und zur invasiven Überwachung durch die Integration von Monitoring-Aktivitäten (vgl. z. B. [BG05]) verglichen.

Für die Anwendung und den Vergleich der Überwachungs- und Steuerungsmaßnahmen werden zwei verschiedene Arten von Anforderungen zugrunde gelegt [ZBH+10]:

- Bereits vor der Ausführung der Prozessinstanz bekannte Anforderungen an die Überwachung und Steuerung des Prozesses (*Designtime Requirements*, kurz DR_i)
- Erst während der Laufzeit des Prozesses auftretende Anforderungen an die Überwachung und Steuerung des Prozesses (*Runtime Requirements*, kurz RR_i)

Das Fallbeispiel umfasst, wie bereits in Abschnitt 4.2.2 beschrieben, die prozessorientierte Zusammenarbeit zwischen einer Heimat-Universität und einer Gast-Universität zur Verwaltung von Auslandsaufenthalten von Studierenden (vgl. Abbildung 8.9a).

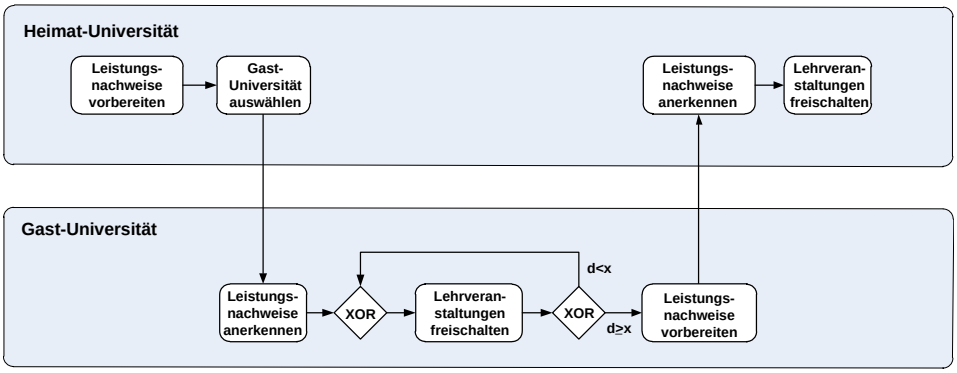
Designtime Requirements: Exemplarisch sei bereits vor dem Beginn der Zusammenarbeit bekannt, dass die Gast-Universität für jeden betreuten Studierenden für die Zeit des Auslandsaufenthalts von der Heimat-Universität in geringer Höhe Finanzmittel zur Kompensation des hierdurch entstehenden Verwaltungsaufwands gestellt bekommt. Hierfür soll (technisch) die Dauer jeder von der Gast-Universität ausgeführten Aktivität des modellierten Prozesses erhoben werden (DR_1). Um mögliche Schwierigkeiten bei der Weiterführung des Studiums an der Gast-Universität zu vermeiden, soll eine

Bestätigung über die erfolgreiche Aufnahme bzw. Weiterführung des Prozesses durch die Gast-Universität erbracht werden (DR_2). Überschreitet die Aktivität „Leistungsnachweise vorbereiten“ eine kritische Dauer (d. h. eine auf dem Zielausführungssystem überdurchschnittlich lange Ausführungszeit), soll diese exemplarisch abgebrochen werden, damit der Kontrollfluss des Prozesses in jedem Fall zur Heimat-Universität zurückkehren kann, um weitere Fehlerbehandlungsmaßnahmen einzuleiten (nicht im Prozess dargestellt). Hierzu soll (technisch) die bei der jeweiligen Gast-Universität übliche (d. h. durchschnittliche) Ausführungszeit erhoben werden (DR_3) und bei Vorliegen einer bestimmten Überschreitung (z. B. um 20%) die Aktivität abgebrochen und übersprungen werden (DR_4). Da es aufgrund von unterschiedlichen Regelungen in den Prüfungsordnungen vorkommen kann, dass die Frist zur erfolgreichen Absolvierung einer Lehrveranstaltung von der Gast-Universität verlängert wird (z. B. aufgrund von nachgewiesener Krankheit des Studierenden) und sich damit auch der ursprünglich geplante Auslandsaufenthalt verlängert, soll die Heimat-Universität über solche Ausnahmefälle informiert werden (DR_5) [ZBH+10].

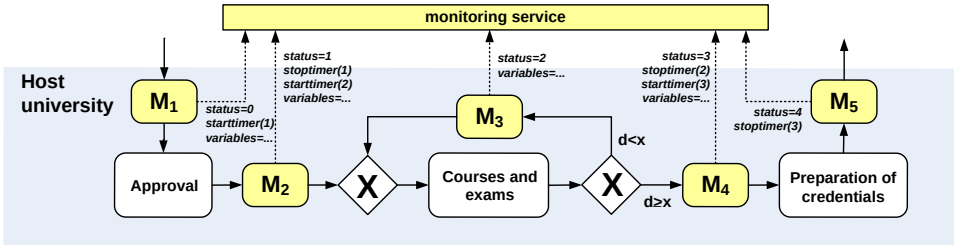
Runtime Requirements: Da es sich bei der entfernt ausgeführten Prozesspartition um einen eher lang andauernden Teilvorgang über mehrere Monate handelt, ist es nicht unwahrscheinlich, dass während des Auslandsaufenthalts von Studierenden weitere, zuvor unerwartete Anforderungen auftreten. Exemplarisch sei in diesem Fall erst zur Laufzeit einer bestimmten Instanz dieses Teilprozesses bekannt, dass eine Organisation zur Studienbeihilfe die Heimat-Universität um Auskunft zum Fortschritt des Studiums einer ihrer Beihilfeempfänger bittet. Hierfür ist (technisch) eine Abfrage des Zustands der entfernt ausgeführten Prozesspartition notwendig (RR_1). Desweiteren soll als Beispiel einer organisationsübergreifenden Änderung angenommen werden, dass der Studierende heiratet und dementsprechend sein Name als eine wesentliche Prozessvariable des Prozesses in allen Prozesspartitionen angepasst werden muss (RR_2) [ZBH+10].

Abbildung 8.9 illustriert die exemplarische Umsetzung der überwachten Prozessinstanz jeweils durch die Integration von Monitoring-Aktivitäten, der (ausschließlich) ereignisbasierten Überwachung und dem hier vorgestellten Ansatz der Bereitstellung des Prozessmanagementsystems als verwaltbare Ressource. Die Ergebnisse des Vergleichs sind in Tabelle 8.7 zusammengefasst.

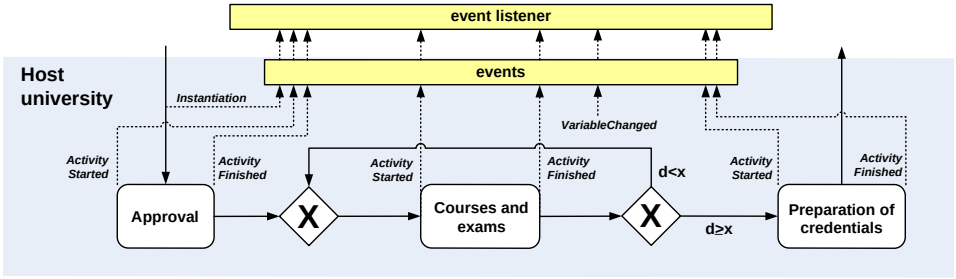
Vergleicht man die Umsetzung der Aspekte, deren Überwachung bereits vor der Ausführung des Prozesses bekannt sind (DR_1 bis DR_5), so zeigt sich, dass in einigen Fällen zur Entwicklungszeit des Prozesses oder kurz vor der Verteilung der entfernt auszuführenden Prozesspartition die Integration von Monitoring-Aktivitäten möglich ist, welche diese Aufgaben während der Ausführung übernehmen (vgl. Abbildung 8.9b). Für die Erhebung der Ausführungsdauer (DR_1), die Bestätigung über die erfolgreiche Übernahme der Prozessinstanz (DR_2) und die Beobachtung von Variablenwerten (DR_5)



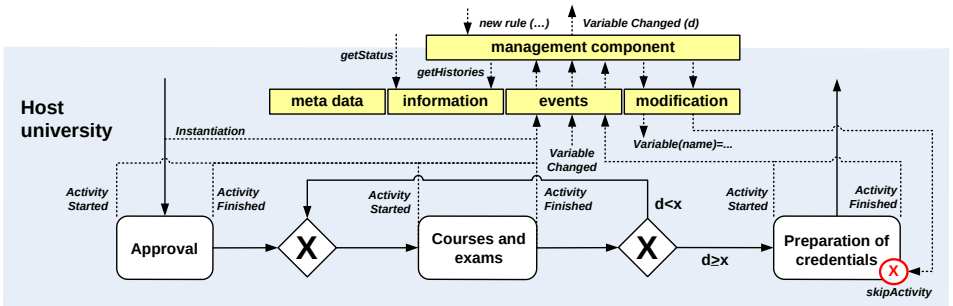
(a) Beispielprozess zum eErasmus-Projekts (ZBH+10)



(b) Überwachung und Steuerung durch Integration von Monitoring-Aktivitäten



(c) Überwachung durch (ausschließlich) ereignisbasiertes Monitoring



(d) Überwachung und Steuerung durch Bereitstellung des Prozessmanagementsystems als verwaltbare Ressource

Abbildung 8.9: Fallbeispiel eErasmus: Veranschaulichender Vergleich verschiedener Ansätze zur Überwachung und Steuerung entfernt ausgeführter Prozesspartitionen (ZBH+10)

können die zwischen den fachlichen Aktivitäten eingefügten Monitoring-Aktivitäten bei einem entsprechenden Fortschritt des Prozesses den jeweiligen Prozesszustand an einen (zentralen) Monitoring-Dienst weiterleiten, so dass der Beobachter der entfernten Prozessausführung (in diesem Fall die Heimat-Universität) die jeweils erforderlichen Daten erhält.

Analog dazu kann die Heimat-Universität bei der Gast-Universität Ereignisse zum Beginn der Ausführung des Prozesses (Instantiierung), zum Beginn und zum Ende der Ausführung von Aktivitäten und zur Veränderung von Variablenwerten abonnieren (vgl. Abbildung 8.9c). Der ereignisbasierte Ansatz ist insofern vorteilhafter, da der fachliche Kontrollfluss nicht durch die Ausführung von Monitoring-Aktivitäten unterbrochen und ggf. (z. B. bei einem temporär nicht verfügbaren Monitoring-Dienst) verzögert werden kann. Zudem muss der Zustand des Prozesses nicht immer, sondern nur im Fall von relevanten (abonnierten) Veränderungen übertragen werden. Das Prozessmanagementsystem als verwaltbare Ressource ermöglicht ebenfalls den ereignisbasierten Ansatz. Über die Möglichkeit einer komplexen Ereignisverarbeitung auf Seiten der Gast-Universität (z. B. über den in Abschnitt 7.4 vorgeschlagenen Management-Dienst) ist darüber hinaus das Filtern und Sammeln von Informationen (z. B. von Informationen zur Ausführungsdauer von Aktivitäten) möglich, so dass nicht jede Information einzeln zur Laufzeit des Prozesses übertragen werden muss, sondern auch eine gesammelte Auswertung bei Erreichen eines definierten Zwischen- oder Endzustands der entfernt ausgeführten Prozesspartition erreicht werden kann. Die Anforderungen DR_1, DR_2 und DR_5 können also – unabhängig von Aufwand oder Qualität – prinzipiell von allen drei Ansätzen erfüllt werden [ZBH+10].

Die Bereitstellung historischer Werte und darauf basierend die Abfrage von Durchschnittswerten (DR_3) erfordert Kenntnis der auf diesem Prozessmanagementsystem vorhergehenden Ausführungen der relevanten Aktivität. Dies ist generell nur möglich, sofern diese Aktivität überhaupt zuvor auf diesem Prozessmanagementsystem ausgeführt wurde. Ist dies der Fall, erfordern die Ansätze der Integration von Monitoring-Aktivitäten und des ereignisbasierten Monitorings zusätzlich, dass bereits Erfahrungswerte durch die der beobachtenden Partei selbst gesammelt wurden, da die Eigenschaften vorhergehender Prozessinstanzen in der aktuell überwachten Prozessinstanz nicht sichtbar sind. Die Bereitstellung des Prozessmanagementsystem als verwaltbare Ressource erlaubt die Abfrage solcher Eigenschaften, sofern sie durch den Anbieter bereitgestellt werden. In Kombination mit Möglichkeiten zur Verarbeitung von Regeln bzw. komplexen Ereignissen (z. B. durch den oben genannten Management-Dienst) ist darüber hinaus eine direkte Ableitung von Anpassungsbedarf in Form der vorgesehenen Abbruchreaktion (DR_4) möglich. Das Abbrechen bzw. Überspringen von als kritisch identifizierten Aktivitäten ist für die Ansätze der Integration von Monitoring-Aktivitäten und des ereignisbasierten Monitorings ebenfalls problematisch. Das ereignisbasierte Monitoring erlaubt nur eine passive Überwachung der Prozessausführung und ermöglicht somit gar kein aktives Eingreifen. Die Integration einer Aktivität zum Abbruch

Anforderungen	Verwaltbare Ressource	Ereignisbasiertes Monitoring	Integration von Monitoring-Aktivitäten
(DR_1) Ausführungsdauer von Aktivitäten	+	+	+
(DR_2) Bestätigung der erfolgreichen (Wieder-) Aufnahme der Prozessinstanz	+	+	+
(DR_3) Feststellung kritischer Ausführungsdauer	+	o	o
(DR_4) Abbruch von Aktivitäten	+	-	o
(DR_5) Beobachtung von Variablenwerten	+	+	+
(RR_1) Abfrage von Zustandsdaten der Prozessinstanz	+	o	o
(RR_2) Modifikation von Prozessvariablen	+	-	-

Tabelle 8.7: Erfüllung verschiedener (technischer) Anforderungen an die Überwachung und Steuerung von Prozessinstanzen (ZBH$^+$10)

der Ausführung einer anderen (atomaren) Aktivität ist auch durch die Integration von zusätzlichen technischen Anweisungen in das Prozessmodell nicht ohne weiteres möglich, da etwaige zusätzliche Kontrollflusselemente, welche den Abbruch erlauben würden, durch den fehlenden Fortschritt des Prozesses unter Umständen gar nicht erreicht werden können. Abhängig von der Ausdrucksmächtigkeit der Prozessbeschreibungssprache ist hier nur die Modellierung einer ereignisbasierten Kontrollflussstruktur und die Integration einer geeigneten Ereignisbehandlungsroutine möglich, um einen Abbruch der Prozessausführung im Einzelfall herbeiführen zu können [ZBH$^+$10].

Der Ansatz der Bereitstellung des Prozessmanagementsystems als verwaltbare Ressource stellt insbesondere in Hinblick auf unerwartete Management-Anforderungen zur Laufzeit des Prozesses große Vorteile dar. Die ungeplante Abfrage von Zustandsinformationen (RR_1) kann dabei auf Basis des vorgestellten Ressourcenmodells über einen einmaligen Abruf der entsprechenden Eigenschaft(en) der betrachteten Prozessinstanz erfolgen. Sowohl der Ansatz der Integration von Monitoring-Aktivitäten als auch der (ausschließlich) ereignisbasierte Ansatz können die angefragten Zustandsinformationen zur Laufzeit nur bereitstellen, sofern die Prozessausführung bis zu diesem Zeitpunkt lückenlos überwacht wurde. Dies ist im Allgemeinen nur möglich, wenn zu Beginn der Prozessausführung vor bzw. nach jeder fachlichen Aktivität eine Monitoring-Aktivität eingefügt wurde bzw. alle verfügbaren Ereignisse abonniert wurden. Ist dies nicht gegeben, ist es mehr oder weniger Zufall, ob eine zur Laufzeit auftretende Anforderung durch den zur Entwicklungszeit geplanten Informationsbedarf erfüllt werden kann [ZBH$^+$10].

Die spontane Modifikation von Variablenwerten (RR_2) ist bei einem rein ereignisbasierten Ansatz aufgrund von fehlenden Operationen zur Modifikation ebenfalls nicht möglich. Bei einer Integration von Monitoring-Aktivitäten, welche jeweils auch die Veränderung der an den Monitoring-Dienst übermittelten Prozessvariablen erlaubt, wäre eine Modifikation nur dann möglich, wenn die potentielle Modifikation auch im Voraus geplant wäre. Die Bereitstellung des Prozessmanagementsystems als verwaltbare Ressource erlaubt prinzipiell die Modifikation ausgewählter Parameter auch während der Laufzeit. Hierbei ist

selbstverständlich zu berücksichtigen, dass die Integrität des Prozesses durch die durchgeführten Maßnahmen nicht gefährdet werden darf. In der vorgestellten Vorgehensweise können Modifikationen nur nach einem geordneten Anhalten der Prozessinstanz vorgenommen werden. Die Berücksichtigung der fachlichen Korrektheit des Prozesses nach der Durchführung von Änderungen bleibt hiervon gleichermaßen für alle drei Ansätze unberührt [ZBH+10].

Betrachtet man auch die nicht-funktionalen Aspekte der drei möglichen Ansätze, so zeigt sich deutlich, dass die gewünschte Trennung zwischen fachlicher Ausführungslogik und technischen Anweisungen zur Überwachung und ggf. Steuerung der Prozessausführung nur durch den ereignisbasierten Ansatz und durch das Prozessmanagementsystem als verwaltbare Ressource umgesetzt wird. Im Gegensatz dazu besitzt der invasive Ansatz der Integration von Monitoring-Aktivitäten den Vorteil, dass hierfür keine Erweiterungen der verwendeten Prozessmanagementsysteme vorgenommen werden müssen. Durch den Aufruf der Monitoring-Aktivitäten kann es jedoch – in Abhängigkeit ihrer Art und Anzahl – zu Verzögerungen der Prozessausführung kommen. In Bezug auf die möglichst geringe Beeinflussung der fachlichen Prozessausführung ist insbesondere der rein ereignisbasierte Ansatz vorteilhaft. Spontane Management-Anforderungen sind jedoch nur zu erfüllen, sofern nach jeder Aktivität alle Informationen des Prozesses an den Beobachter übertragen werden, was zu einem dauerhaft hohen und – sofern zur Laufzeit gar keine spontanen Anforderungen eintreten – ggf. auch zu einem unnötigen Nachrichtenaufkommen führt [ZBH+10].

Mit Möglichkeiten zur nicht-invasiven Überwachung und Steuerung stellt der Ansatz auf Basis des Prozessmanagementsystems als verwaltbare Ressource einen Kompromiss dar, welcher den Ansatz zum ereignisbasierten Monitoring subsumiert, jedoch gleichzeitig Möglichkeiten zur (intendierten) Beeinflussung der Prozessausführung bereitstellt. Werden Ereignisse blockierend abonniert und nicht direkt beim ausführenden Prozessmanagementsystem gefiltert bzw. verarbeitet, tritt hinsichtlich der Beeinflussung eine ähnliche Problematik wie im Fall der Integration von Monitoring-Aktivitäten auf. Management-Anforderungen, welche erst zur Laufzeit des Prozesses auftreten, können über die stets verfügbare Management-Schnittstelle durch eine minimale Anzahl an Nachrichten erfüllt werden. Die Bereitstellung des Prozessmanagementsystems als verwaltbare Ressource bietet zusammengefasst daher nicht nur hinsichtlich im Voraus unbekannter Management-Anforderungen einen größeren Handlungsspielraum, sondern stellt auch in Bezug auf bereits zur Entwicklungszeit bekannte Anforderungen keine schlechtere Lösung dar als die untersuchten bestehenden Ansätze [ZBH+10].

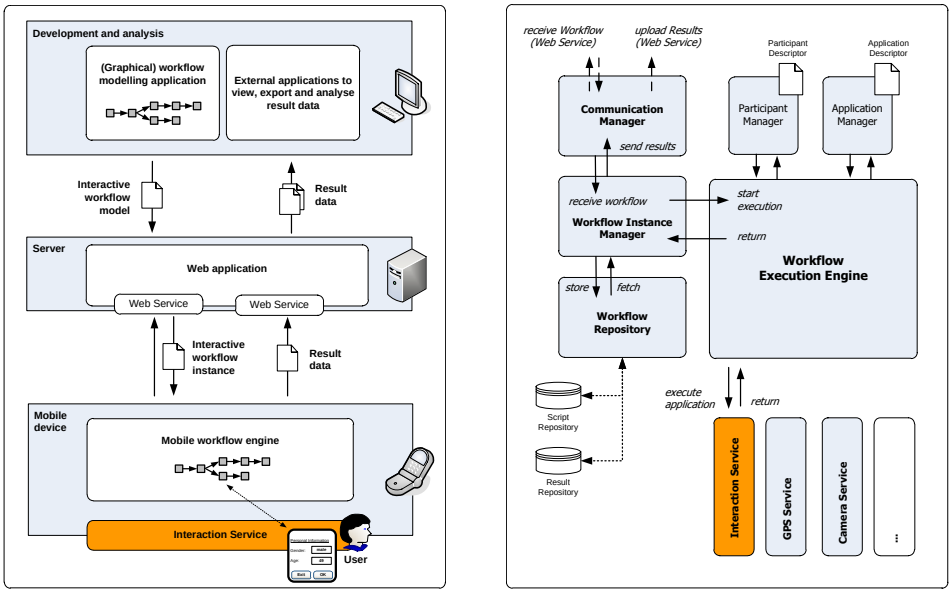
Durch die Möglichkeit zu einer lokalen Ereignisverarbeitung im Rahmen des optionalen Management-Dienstes kann als Ergänzung zur verwaltbaren Ressource die Anzahl der auszutauschenden Nachrichten weiter reduziert und damit auch die Reaktionszeit zur Ableitung notwendiger Steuerungsmaßnahmen verbessert werden. Da es sich in diesem Fall um eine lokale Berechnung handelt, ist die Reaktionszeit durch die Anzahl und Komplexität der

zu überprüfenden Verarbeitungsregeln dominiert. Entsprechende Benchmarks können den für die in der prototypischen Realisierung eingesetzten *Esper*-Plattform veröffentlichten Ergebnissen entnommen werden [Esp07].

8.4.4 Prozessorientierte Datenerhebung durch mobile Geräte

Das in Abschnitt 6.4 beschriebene Konzept der abstrakten Benutzungsschnittstellen für prozessorientierte Anwendungen und der entsprechend in Abschnitt 7.6 vorgestellte Interaktionsdienst zur kontext- und plattformabhängigen Repräsentation von Interaktionselementen können nicht nur für dynamisch verteilt ausgeführte Prozesse eingesetzt werden, sondern können auch eine parallele (nicht-verteilte) Ausführung von Prozessinstanzen desselben Prozessmodells auf verschiedenartigen (mobilen) Endgeräten angemessen unterstützen. Ein Beispiel für eine solche Anwendung stellt die in Abschnitt 4.2.3 beschriebene prozessorientierte Datenerhebung im Rahmen einer breit angelegten Marktforschungsstudie dar. Hierbei werden elektronische Fragebögen als Abfolge von Benutzerinteraktionen in Form von Prozessen spezifiziert und parallel von verschiedenen Benutzern auf ihren (untereinander heterogenen) mobilen Endgeräten ausgeführt. Jedes teilnehmende mobile Endgerät wird dazu einmalig mit einer plattformabhängig aufbereiteten Prozess-Engine ausgestattet, welche beliebige in der jeweils unterstützten Prozessbeschreibungssprache spezifizierte Fragebögen anzeigen kann. Stehen weitere Anwendungen des mobilen Endgeräts bereit, welche für die Durchführung der Datenerhebung genutzt werden können, so können diese über die Prozess-Engine integriert und über die auszuführenden Prozesse aufgerufen werden. Im Rahmen dieser Arbeit erfolgte eine prototypische Umsetzung für Mobiltelefone (J2ME-Plattform, CLDC-Profil), PDA (J2ME-Plattform, CDC-Profil) und Notebook (J2SE-Plattform) mit einer einfachen XPDL-Prozess-Engine und der exemplarischen Anbindung der geräteinternen Kamera und des GPS-Empfängers (vgl. [ZBV09] zu Details).

Die übergeordnete Architektur für die prozessorientierte Datenerhebung ist in Abbildung 8.10a dargestellt. Zunächst wird ein elektronischer Fragebogen modelliert, d. h. der Inhalt und die Abfolge der zu präsentierenden und der einzugebenden Daten werden festgelegt und ggf. im Kontrollfluss mit weiteren Anwendungen des mobilen Endgeräts integriert. Dieser Schritt erfolgt auf einem zentralen und in der Regel stationären System (hier ein Desktop-PC). Die Modellierung der Benutzerinteraktionen wird hierbei durch TERESA [MPS04] unterstützt (vgl. Abschnitt 6.4.3 und [Vil08]). Resultat ist ein XPDL-Prozess, welcher beliebige Interaktionsaktivitäten enthält, welche jeweils eine abstrakte Beschreibung der Repräsentation der Benutzungsschnittstelle referenzieren. Das Paket aus XPDL-Prozessbeschreibung und abstrakter Beschreibung der Benutzungsschnittstellen wird auf einem Server abgelegt, wo es von den zur Datenerhebung ausgewählten Personen über einen Web Service heruntergeladen werden kann [ZBV09].



(a) Systemarchitektur zur Verteilung und Durchführung prozessorientierter Datenerhebungen

(b) Ausführungsumgebung und Integration des Interaktionsdienstes

Abbildung 8.10: Fallbeispiel prozessorientierte Datenerhebung: Systemarchitektur und Integration des Interaktionsdienstes (ZBV09)

Auf der Seite des mobilen Endgeräts wird das Paket aus XPDL-Prozessbeschreibung und abstrakter Beschreibung der Benutzungsschnittstellen in einem lokalen Repository gespeichert und für die Durchführung der Datenerhebung auf der lokalen Prozess-Engine des mobilen Endgeräts installiert und instantiiert. Wird im Kontrollfluss eine Interaktionsaktivität aufgerufen, so wird durch die Prozess-Engine der Interaktionsdienst aufgerufen, welcher auf Basis der abstrakten Beschreibung eine für dieses Endgerät geeignete Benutzungsschnittstelle erzeugt, die Eingabe des Benutzers entgegen nimmt und den Kontrollfluss zurück an die Prozess-Engine übergibt (vgl. Abbildung 8.10b). Da die Prozessbeschreibung sowohl den Kontroll- als auch den Datenfluss vollständig lokal verwaltet, ist für die gesamte Datenerhebung keine Netzwerkverbindung erforderlich. Nach Abschluss der Datenerhebung werden die Ergebnisse in Form der Instanzdaten des Prozesses auf dem mobilen Endgerät zwischengespeichert, bis eine Übertragung der Daten zum Server möglich und der Datentransfer über den hierfür bereitgestellten Web Service abgeschlossen ist. Auf dem Server werden die (homogenen) Daten aller mobilen Benutzer gesammelt und stehen für eine Weiterverarbeitung zur Verfügung [ZBV09].

Aufgrund der Ausgestaltung als Dienst ist der Interaktionsdienst lose gekoppelt in die Ausführungsumgebung integriert und kann so unabhängig von der Prozess-Engine einmalig für jedes zu verwendende mobile Endgerät angepasst werden. Bei der Modellierung von elektronischen Fragebögen müssen die potentiell heterogenen Endgeräte als Schnittstelle zu den Benutzern da-

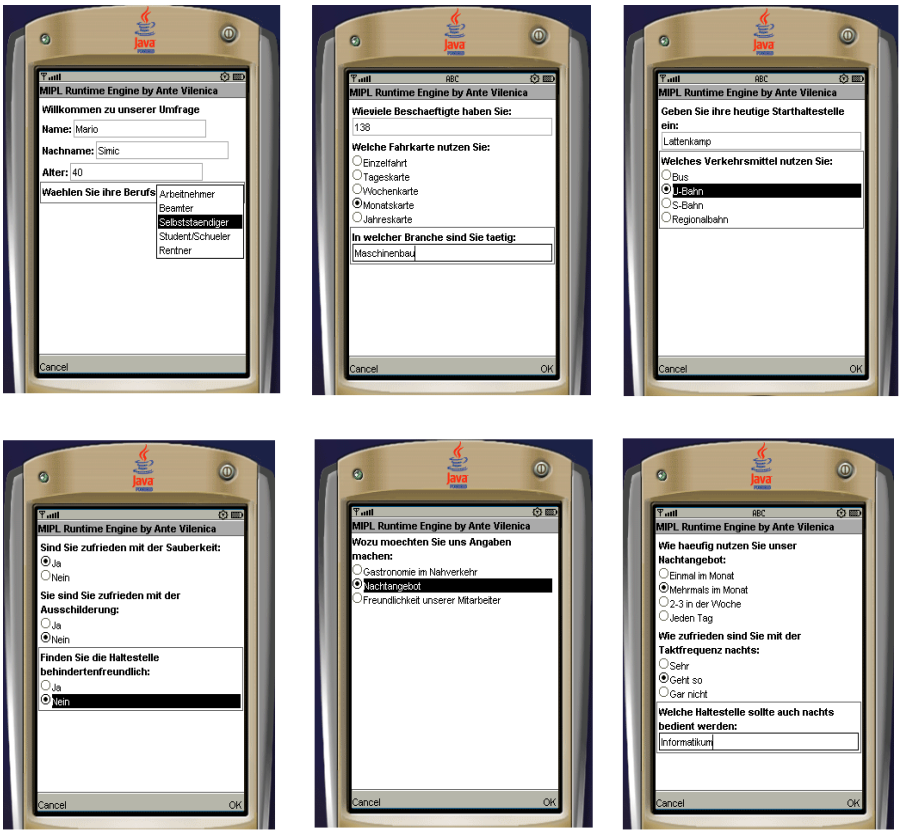


Abbildung 8.11: Anwendungsszenario elektronische Verkehrsumfrage (Vil08)

durch nicht mehr berücksichtigt werden. Auch bei einer großen Menge der an der Datenerhebung beteiligten Personen muss der Fragebogen somit nur einmal modelliert werden, was zu einer deutlich schnelleren Reaktionszeit bei der Durchführung von elektronisch gestützten Datenerhebungen führt.

Die aus den unterschiedlichen Implementierungen resultierenden Benutzungsschnittstellen wurden zunächst mittels zweier Emulatoren (*Sun Wireless Toolkit* und *Nokia SDK*) getestet, welche für die Simulation verschiedener PDAs und Mobiltelefone mit unterschiedlichen Einstellungen (z. B. Abmessungen und Art des Displays, Bildschirmauflösung und Eingabemedien) und Leistungsmerkmalen (z. B. Prozessorleistung und Bandbreite für Datenübertragung) konfiguriert werden können. Da die Emulatoren die Benutzungsschnittstellen dementsprechend unterschiedlich umsetzen, kann auf diese Weise basierend auf einer Basisimplementierung des Interaktionsdienstes eine geeignete Abbildung der abstrakten Interaktionsobjekte (AIO) auf die konkreten Interaktionsobjekte vorgenommen werden (vgl. Abschnitt 6.4.4). In einem zweiten Schritt wurde die J2ME-Implementierung auf einer ausgewählten Menge von „echten“ Mobiltelefonen verschiedener Hersteller getestet. Als Ergebnis hat sich herausgestellt, dass die Basisimplementierung bereits ohne größere Anpassungen für eine Vielzahl von javafähigen Endgeräten eingesetzt werden kann. Die resultierenden Benutzungsschnittstellen unter-

scheiden sich dabei aufgrund der herstellerspezifischen Interpretation der konkreten Interaktionsobjekte leicht, setzen jedoch in fast allen Fällen die bei der Modellierung intendierten Benutzungsschnittstellen um. Ein Beispiel für das Ergebnis der Umsetzung einer prozessgestützten elektronischen Verkehrsumfrage kann Abbildung 8.11 entnommen werden.

8.4.5 Verteilung zum Umgang mit großen Datenmengen

In Abschnitt 4.2.5 wurde ein Szenario vorgestellt, in dem eine verteilte Ausführung auf Basis von Dienstgütekriterien, wie insbesondere der Vermeidung von (mehrfachen) umfangreichen Datenübertragungen, vorteilhaft ist. Ist der Umfang der Daten bereits vor Beginn der Prozesslaufzeit bekannt, so ist eine verteilte Prozessausführung zum Beispiel auf Basis einer physischen Partitionierung des Prozessmodells und der Zuweisung der entstehenden Prozesspartitionen zu den teilnehmenden Ausführungseinheiten möglich. Können die für die Verteilung relevanten Parameter wie die teilnehmenden Ausführungseinheiten, die Granularität und die Zuordnung der Prozesspartitionen erst zur Laufzeit des Prozesses festgelegt werden, kann die Migration von Prozessinstanzen für eine dynamische Verteilung des Prozesses angewendet werden.

Die dynamische Anpassung der Prozessausführung ist unter den in Abschnitt 6.2 genannten Einschränkungen und etwaigen benutzerdefinierten Rahmenbedingungen für die Verteilung prinzipiell jederzeit möglich. Ein wesentliches Problem zum dynamischen Umgang mit großen Datenmengen besteht jedoch darin, dass ein unerwarteter umfangreicher Datentransfer unter Umständen erst festgestellt werden kann, sobald er angelaufen bzw. abgeschlossen ist. Sind die Daten bereits (teilweise) übertragen, so müssten diese bei einer Migration des Prozesses mit den Prozessinstanzdaten (z. B. als Wert einer Prozessvariablen) für die Weiterführung der Ausführung erneut übertragen werden – wodurch der Vorteil der Prozessmigration als Möglichkeit zum *Function Shipping* relativiert würde. Die Herausforderung bei diesem Anwendungsfall besteht also darin, die dynamische Verteilung des Prozesses herbeiführen zu können, *bevor* es zur Ausführung der Aktivität mit der resultierenden Übertragung der großen Datenmengen kommt. Mögliche Ansäze hierzu werden im Folgenden diskutiert.

Liegen die umfangreichen Daten auf Seiten des prozessausführenden Systems vor, besteht für eine (Neu-)Verteilung des Prozesses zur Optimierung des Datentransfers kein Anlass. Liegen die umfangreichen Daten auf der Seite einer entfernten Ressource vor, welche durch das aktuell prozessausführende System aufgerufen wird, so kann eine dynamische Verteilung des Prozesses zur Vermeidung der Datenübertragung entweder über Hinweise aus dem fachlichen Inhalt des Prozesses oder über die Feststellung unzureichender Dienstgütekriterien (z. B. Übertragungsbandbreite) dieser Ressource von der aktuellen Ausführungseinheit angestossen werden. Alternativ kann die dynamische Verteilung des Prozesses gänzlich von der Seite der Ressource initiiert

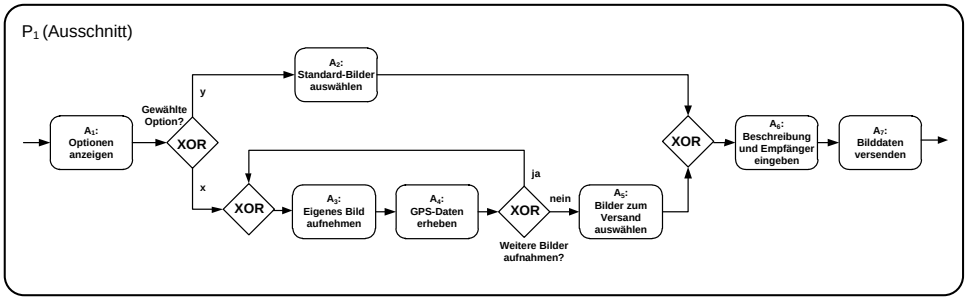


Abbildung 8.12: Anwendungsszenario zur Vermeidung mehrfachen Datentransfers mit umfangreichen Datenmengen (Beispiel)

werden. Letzteres ist in der Regel nur bei einer Benutzerinteraktion möglich, bei welcher dem Benutzer die Problematik der Datenübertragung bewusst und der Umfang der zu übertragenden Daten bekannt ist.

Abbildung 8.12 zeigt den Ausschnitt eines Prozessmodells als Beispiel für die Auswahl und den Versand von Bilddaten und ergänzenden Beschreibungen. Typische Anwendungsfälle hierfür sind die Erhebung und Mitteilung von wissenschaftlichen Daten oder das Versenden von virtuellen Postkarten im Bereich von (mobilen) sozialen Netzwerken. Die serverbasierte Anwendung erlaubt es einem Benutzer, entweder eine Menge von bereits auf dem Server gespeicherten Bildern (z. B. anhand von Miniaturansichten) auszuwählen oder selbst (z. B. während einer Dienst- oder Urlaubsreise) Bilder mit der Kamera eines Mobiltelefons inklusive Positionsangaben aufzunehmen, um daraus später eine Untermenge für den Versand auszuwählen. Als veranschaulichendes Beispiel sei hierbei angenommen, dass die in Aktivität A_3 aufgenommenen Bilder jeweils zum prozessausführenden System transferiert und dort in einer Prozessvariable (z. B. einer Liste) gespeichert werden, und dass die Menge der gespeicherten Bilder in Aktivität A_5 zurückübertragen wird, um dem Benutzer eine Auswahl zu ermöglichen.

Um einen Teil der Prozessausführung zu übertragen, wird sowohl für die physische Partitionierung des Prozessmodells als auch für die Migration von Prozessinstanzen vorausgesetzt, dass alle an der Ausführung teilnehmenden Parteien über kompatible Ausführungseinheiten verfügen (vgl. Abschnitt 4.2.5). Für eine dynamische Verteilung des Prozesses sind nun die folgenden Alternativen möglich, um die Menge der bei der Prozessausführung zu transferierenden Daten zu reduzieren:

- **Variablenbasierte Zuweisungsstrategie:** Die Migrationsentscheidung wird an den Wert der Variablen „Option“ gebunden und die Prozessausführung wird an die Ausführungseinheit des (mobilen) Benutzers delegiert, wenn eigene Bilder aufgenommen und ausgewählt werden sollen („Option = y“) (vgl. Abschnitt 7.5.3 zu *variablenbasierter Zuordnung*).
- **Dienstgütebasierte Zuweisungsstrategie:** Die Migrationsentscheidung wird aufgrund der nicht-funktionalen Eigenschaften der zur Ausführung von A_3 ausgewählten Ressource vorgenommen. Handelt es

sich hierbei um einen mobilen Prozessteilnehmer, welcher nur über eine leistungsschwache oder kostenpflichtige Kommunikationsverbindung verfügt, so wird in der Vorbereitungsphase der Ausführung von A_3 (bei der Auswahl der Ressource) eine Migration des Prozesses zu der Ausführungseinheit angestossen, welche diese Ressource lokal zugreifen kann (vgl. [HSZ11]).

- **Manuelle Zuweisung:** Bekommt ein Benutzer die Aufgabe A_3 zur Ausführung zugewiesen und ist dem Benutzer der voraussichtliche Umfang der Daten bekannt, so kann über die PMaaS-Schnittstelle bei Vorliegen der benötigten Berechtigungen auch von extern eine Migrationsentscheidung initiiert werden. Voraussetzung hierfür ist jedoch, dass die Ausführung der Aktivität A_3 noch nicht begonnen wurde. Dies ist zum Beispiel der Fall, wenn die Aufgabe in der persönlichen Aufgabenliste des Benutzers angezeigt, jedoch durch diesen noch nicht zur Bearbeitung geöffnet wurde (vgl. Abschnitt 8.3.3 zu *User Tasks*).

Bei allen drei Möglichkeiten wird zur Laufzeit eine dynamische Verteilung des Prozesses ermöglicht. Für die Anwendung der variablenbasierten und der dienstgütebasierten Zuordnungsstrategien muss aber zumindest im Voraus bekannt sein, dass es aufgrund der Prozessmodellierung (Anordnung und Inhalt der Aktivitäten A_3 und A_5) und der potentiell mobilen Prozessteilnehmer zu einem Bedarf an dynamischer Verteilung während der Ausführung kommen kann. Zudem muss im Voraus bekannt sein, wie dieser Anpassungsbedarf zur Laufzeit festgestellt werden kann (Auswertung der Variablen „Option“ bzw. Auswertung eines geeigneten Dienstgütekriteriums). In beiden Fällen kann zudem nicht sichergestellt werden, dass die später zu übertragende Datenmenge bei nicht-verteilter Ausführung höher ist als die bei der Migration der Prozessinstanz anfallende Datenmenge.

Die durch den Benutzer initiierte manuelle Zuweisung der Prozessausführung erlaubt prinzipiell den Umgang mit im Voraus unbekannten Problemen, wie der zu berücksichtigenden Menge an transferierten Daten. Diese Variante erfordert jedoch ein gewisses Anwendungswissen bei Prozessteilnehmern, für welche die Ausführung des Prozesses bis auf die zugewiesene Aufgabe(n) jedoch eigentlich transparent sein sollte. Über die zugewiesene Aufgabe hinausgehendes Anwendungswissen kann nicht in allen Fällen vorausgesetzt werden. Aus diesem Grund ist die Anwendbarkeit einer dynamischen Verteilung des Prozesses für den Umgang mit erst zur Laufzeit auftretenden Anforderungen an die Reduzierung der zu übertragenden Datenmenge nur eingeschränkt möglich. Im Folgenden werden die durch die Anwendung der vorgeschlagenen Konzepte gesammelten Erfahrungen und Erkenntnisse in einer integrierten Aufwands- und Flexibilitätsbetrachtung zusammengefasst und die für eine allgemeine Bewertung interessanten Aspekte vertieft.

8.5 Aufwands- und Flexibilitätsbetrachtung

In Abschnitt 8.1 wurden verschiedene Kriterien definiert, um den im Allgemeinen nicht quantifizierbaren Aufwand für die Durchführung dynamischer Anpassungen sowie den konkreten Nutzen der Anpassungsfähigkeit zu untersuchen und zu bewerten. Im Folgenden werden die in dieser Arbeit vorgeschlagenen Konzepte auf Basis der praktischen Erfahrungen mit der prototypischen Implementierung und ergänzenden theoretischen Überlegungen untersucht und bewertet. Zunächst wird der Aufwand betrachtet, welcher durch die permanente Möglichkeit zu einer spontanen Überwachung, Verteilung oder anderweitigen Steuerung der Prozessausführung entsteht. Hierzu zählen der Entwicklungsaufwand für die Anpassung der Prozessmanagementsysteme zur Nutzung der hier vorgeschlagenen Middleware-Komponenten (vgl. Abschnitt 8.5.1), die nur in Sonderfällen notwendige Anpassung von Prozessmodellen (vgl. Abschnitt 8.5.2) und die Betrachtung einer etwaigen Beeinflussung nicht überwachter und nicht verteilt ausgeführter Prozessinstanzen auf der für die dynamische Verteilung angepassten Ausführungseinheit (vgl. Abschnitt 8.5.3). Im Hauptteil der Aufwandsbetrachtung werden die generellen Einflussfaktoren für den Aufwand der Überwachung einzelner Prozessinstanzen (vgl. Abschnitt 8.5.4) und für die Reaktionszeit zur Durchführung ihrer dynamischen Verteilung beschrieben (vgl. Abschnitt 8.5.5). Die verschiedenen Arten von Aufwand und ihre Zuordnung zu Entwicklungs- und Laufzeit prozessorientierter Anwendungen sind vorab zur Veranschaulichung in Abbildung 8.13 dargestellt.

Dem notwendigen Aufwand für die Anpassbarkeit der verteilten Prozessausführung wird im Folgenden der erreichbare Umfang des Anpassungsspielraums (insbesondere das Verteilungsspektrum) gegenüber gestellt (vgl. Abschnitt 8.5.6). Im Vergleich dazu wird diskutiert, welchen Aufwand der Ansatz der physischen Partitionierung von Prozessen verursachen würde, um den gleichen Umfang des Anpassungsspielraums zu erreichen (vgl. Abschnitt 8.5.7).

Die Aufwands- und Flexibilitätsbetrachtungen für die ergänzend vorgeschlagenen Maßnahmen zur regel- und ereignisbasierten Überwachung und zur proaktiven Anpassung der Prozessausführung auf Grundlage von Kontextdatenprognosen sind aufgrund der Optionalität der Komponenten separat in Abschnitt 8.5.8 zusammengefasst.

8.5.1 Entwicklungsaufwand

Der hier vorgestellte Ansatz erfordert einmaligen Aufwand für die notwendige Anpassung zur Einrichtung von (ergänzenden) Middleware-Komponenten, wobei der Aufwand einmalig zur Entwicklungszeit eines (potentiell teilnehmenden) Prozessmanagementsystems auftritt. Bei Standardprodukten kann die Entwicklung z. B. durch den Hersteller des Prozessmanagementsystems vorgenommen werden und im Rahmen eines Upgrades oder Updates ausgeliefert

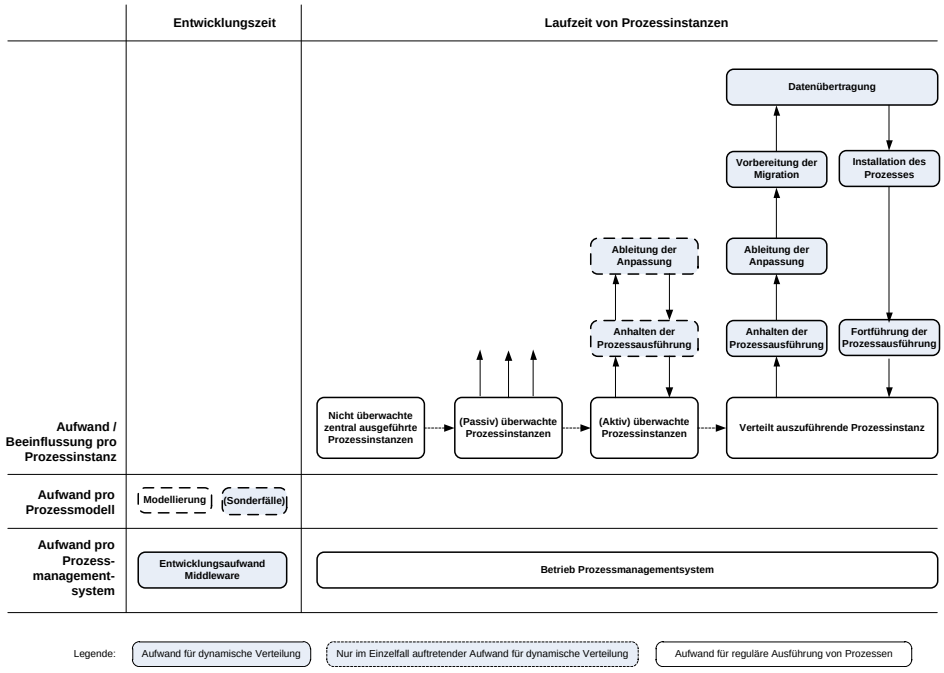


Abbildung 8.13: Verschiedene Arten von Aufwand für die Ermöglichung von individuell dynamisch überwachbaren und verteilt ausführbaren Prozessinstanzen

werden, so dass der Entwicklungsaufwand nicht auf Seiten jedes Betreibers einzeln entsteht. Für eine prototypische Implementierung verschiedenen Umfangs lag der Entwicklungsaufwand für die Anpassung der drei betrachteten Prozessmanagementsysteme zwischen 150 und 168 Personenstunden (vgl. Abschnitt 8.2). Der Entwicklungsaufwand für die hier vorgestellten Middleware-Komponenten (vgl. Abschnitt 7) ist dabei nicht inkludiert.

Dem Aufwand für die einmalig notwendige Anpassung zur Einrichtung der Middleware-Komponenten steht zunächst kein direkter Flexibilitätsgewinn gegenüber. Es handelt sich hierbei um eine Art „Investition“, deren Verhältnismäßigkeit von der Art und Anzahl der zukünftig mit dieser Middleware dynamisch verteilt ausgeführten Prozesse abhängt. Hierbei erweist sich der gewählte *Process-Management-as-a-Service*-Ansatz als wichtiger Aspekt, da hierdurch nicht nur eigene Prozesse verteilt ausgeführt werden können, sondern insbesondere auch durch die (teilweise) Ausführung von Prozessen anderer Organisationen bzw. Organisationseinheiten Synergieeffekte erzielt werden können (vgl. Abschnitt 8.4.2).

8.5.2 Aufwand für die Anpassung von Prozessmodellen

Wie bei der Untersuchung der Migrierbarkeit von Prozessinstanzen verschiedener (standardisierter) Prozessbeschreibungssprachen in Abschnitt 8.3 dargelegt wurde, ist bei dem hier vorgestellten Ansatz eine Anpassung bestehender Prozessmodelle für die Verteilung des Prozesses prinzipiell nicht notwendig. Die Verteilung des Prozesses muss abhängig von den durch die

Prozessbeschreibungssprache bereitgestellten Kontrollflusskonstrukten unter Umständen beschränkt werden, um die fachlich-korrekte Ausführung nicht zu gefährden. Diese Einschränkungen gelten jedoch weitestgehend auch für andere Verteilungsmodelle. Auch für die Überwachung und Steuerung des Prozesses muss aufgrund der Bereitstellung von Managementfunktionalitäten durch das Prozessmanagementsystem kein Eingriff in den fachlichen Prozess vorgenommen werden. Es konnten jedoch insgesamt die folgenden Einschränkungen (Sonderfälle) identifiziert werden:

- Die Prozessbeschreibung muss eine eindeutige Identifikation ihrer Kontrollflusselemente erlauben, um diese durch Migrationsdaten zu referenzieren. Ist dies nicht der Fall, muss die Benennung einmalig angepasst werden.

- Entscheidungen über die dynamische Verteilung des Prozesses, welche auf Informationen basieren, die nicht durch das Prozessmanagementsystem als verwaltbare Ressource bereitgestellt werden können, erfordern unter Umständen weiterhin die Integration von fachlichen oder technischen Aktivitäten zur Erhebung der benötigten Daten (vgl. z. B. Abschnitt 8.4.5).

- Für die Verwendung der Beschreibung von abstrakten Benutzungsschnittstellen ist einmalig eine Referenz auf die abstrakte Schnittstellenbeschreibung aus der Aktivität, in welcher die Benutzerinteraktion durchgeführt werden soll, notwendig.

Die Verteilung bzw. Überwachung der von einem Prozessmodell zukünftig abgeleiteten Prozessinstanzen ist von diesen Einschränkungen weitestgehend unabhängig. Geht man von dem Extremfall dynamisch verteilt ausgeführter Prozesse aus, in welchem jede Prozessinstanz unterschiedlich verteilt ausgeführt bzw. überwacht werden soll, so ist trotzdem nur einmalig eine (einfache) Anpassung pro Prozessmodell notwendig. Mit Ausnahme dieser Sonderfälle ist bei dem hier vorgeschlagenen Lösungsansatz keine Anpassung von Prozessmodellen erforderlich, so dass somit hierfür in der Regel auch kein Aufwand zu berücksichtigen ist.

8.5.3 Beeinträchtigung nicht überwachter Prozessinstanzen

Bei der in Abschnitt 3.2 diskutierten Effizienz der Anpassung steht im Vordergrund, dass Anpassungen vorgenommen werden können, ohne dass die Leistung des Systems während der Anpassung wesentlich beeinträchtigt wird [GP00]. Wie in Abschnitt 7.5.3 gezeigt wurde, werden durch die Spezifikation von benutzerdefinierten Vorgaben oder lokalen Richtlinien die mit der angegebenen Strategie verknüpften Softwaremodule (*Migration Decision Modules*) aktiviert, um die Ausführung der relevanten Prozessinstanzen zu überwachen, damit der richtige Zeitpunkt zur Migration nicht verpasst

wird. In dem vorgestellten Ansatz ist daher eine Beeinträchtigung der Prozessausführung im Allgemeinen abhängig von der Art der Integration der Sensor- und Effektorschnittstelle, von der Art und Anzahl der eingesetzten Überwachungsmechanismen sowie von der Lokalität und Komplexität der Migrationsentscheidungen.

Da die Sensor- und Effektorschnittstelle bei der hier vorgestellten Implementierung auf Prozessinstanzebene arbeitet, werden sowohl Prozessinstanzen desselben Prozessmodells als auch die Prozessinstanzen anderer Prozessmodelle nicht durch Maßnahmen zur etwaigen Verteilung oder Überwachung einer bestimmten anderen Prozessinstanz beeinträchtigt. Bei der im Rahmen der prototypischen Implementierung gewählten ereignisbasierten Integration der Sensorschnittstelle wird diese erst aktiviert, sobald sich mindestens ein Empfänger für die durch diese Schnittstelle erzeugten Ereignisse registriert. Sofern eine Prozessinstanz nicht überwacht wird, sind die Sensor- und Effektorschnittstellen daher nicht aktiv und der Programmcode zur Durchführung von Migrationsentscheidungen muss nicht durchlaufen werden. Der für die Auswertung einer booleschen Variablen pro navigiertes Kontrollflusselement entstehende Rechen- und Zeitaufwand ist mit deutlich weniger als 1 Millisekunde (Notebook mit 1,3 GHz und Pentium M Prozessor) zu vernachlässigen und die normale Ausführung des Prozesses wird daher nicht beeinflusst.

Ein *nicht überwachter* Prozess kann dennoch jederzeit (auch während seiner Ausführung) durch die Erzeugung oder Änderung von Migrationsdaten oder lokalen Richtlinien zu einem *überwachten* Prozess werden. Abbildung 8.13 zeigt neben den dargestellten Arten von Aufwand auch die Evolution einer nicht-überwachten, zentral ausgeführten Prozessinstanz zu einer überwachten, steuerbaren und verteilt auszuführenden Prozessinstanz. Somit können unvorhergesehene Ereignisse, welche eine spontane Überwachung, Verteilung oder anderweitige Steuerung dieses Prozesses notwendig machen, berücksichtigt werden. Der entstehende Aufwand wird im Folgenden betrachtet.

8.5.4 Aufwand für die Überwachung von Prozessinstanzen

Werden benutzerdefinierte Rahmenbedingungen oder lokale Richtlinien für die Überwachung einzelner Prozessinstanzen definiert, um Prozesse ggf. zu migrieren oder anders steuernd auf die Prozessausführung einzuwirken, so sind die Sensor- und Effektorschnittstellen bei diesen Prozessinstanzen aktiviert und es werden Ereignisse erzeugt, welche von den Softwaremodulen zur Durchführung von Migrationsentscheidungen (*Migration Decision Modules*) oder durch beliebige andere Konsumenten abonniert und verarbeitet werden können. Bei Empfang eines relevanten Ereignisses wird der Programmcode zur Durchführung von Entscheidungen über Anpassungsmaßnahmen durchlaufen. Da das Ereignis nicht zwingend auftreten muss (z. B. bei Abonnement eines Fehler-Ereignisses) oder sogar mehrmals pro navigiertem Kontrollfluss-

element auftreten kann (z. B. bei Abonnement aller Zustandsereignisse zur Erhebung der Ausführungsdauer einer Aktivität), ist der hierfür erforderliche Rechen- und Zeitaufwand stark von den jeweiligen Anpassungsstrategien abhängig (vgl. Beispiele in Abschnitt 7.5.4).

Aufgrund der ereignisbasierten Integration kann der Programmcode zur Durchführung von Entscheidungen über Anpassungsmaßnahmen bei einer rein *passiv* überwachten Prozessinstanz parallel bzw. nebenläufig zur Prozessausführung ausgeführt werden. Es entsteht daher in diesem Fall auf Seiten der Prozessausführung nur Aufwand für das Absetzen der Ereignisse auf lokaler Ebene, was hinsichtlich des hierfür erforderlichen Rechen- und Zeitaufwands von weniger als 1 Millisekunde (Notebook mit 1,3 GHz und Pentium M Prozessor) pro Ereignis vernachlässigt werden kann.

Zu einer Beeinflussung der betrachteten Prozessinstanz kommt es immer dann, wenn die Prozessausführung für die Durchführung von Entscheidungen über Anpassungsmaßnahmen temporär angehalten werden muss. Das Anhalten (und ggf. die Fortführung) kann dabei über das Abonnement blockierender Ereignisse oder durch explizite Steuerungsanweisungen geschehen. Die Verzögerung der Ausführung entspricht dann der Rechenzeit für die Berechnung der Entscheidung über Anpassungsmaßnahmen (welche auch die Entscheidung eines Benutzers integrieren und somit lang andauernd sein kann) plus der Zeit für das Anhalten und ggf. Wiederaufsetzen der Prozessausführung. Da unter Umständen auf die Ausführung von Aktivitäten auf parallelen Pfaden gewartet werden muss, bevor die Ausführung vollständig angehalten werden kann, ist dieser Aspekt nicht zu vernachlässigen. Gleiches gilt für das Wiederaufsetzen der Prozessausführung, da ggf. bereits allozierte Ressourcen für anstehende Aktivitäten in der Zwischenzeit freigegeben wurden und neu akquiriert werden müssen. Im schlechtesten Fall wird die Entscheidung über Anpassungsmaßnahmen durch eine externe Komponente durchgeführt, welche über ein Netzwerk (mit ggf. nur geringer Kommunikationsbandbreite) angeschlossen ist. In diesem Fall muss auch noch die Nachrichtenlaufzeit berücksichtigt werden.

Analog zu Abbildung 8.13 soll im Folgenden unterschieden werden, ob eine Überwachung des Prozesses zu einer Verteilung der Prozessinstanz oder zu einer sonstigen Steuerungsmaßnahme (z. B. zur inhaltlichen Anpassung des Prozesses) führt. Sei zusammenfassend T_A die Zeit für das Anhalten der Prozessausführung inklusive des Wartens auf die Beendigung von Aktivitäten, T_E die Zeit für die Durchführung der Entscheidung über Anpassungsmaßnahmen, T_N die Nachrichtenlaufzeit, T_S die Zeit zur Durchführung von sonstigen Steuerungsmaßnahmen und T_W die Zeit für das Wiederaufsetzen der angehaltenen Prozessausführung, so entsteht für jeden Vorgang einer aktiven Überwachung, welcher zu einer Verteilung (Migration) des Prozesses führt, ein Zeitaufwand $T_{UM} = T_A + 2T_N + T_E$ und für jeden Vorgang einer aktiven Überwachung, welcher nicht zu einer Verteilung, sondern zu einer sonstigen Steuerungsmaßnahme führt, ein Zeitaufwand $T_{US} = T_A + 2T_N + T_E + T_S + T_W$. Ist die Entscheidung über die Durchführung von Anpassungsmaßnahmen negativ, so ergibt sich T_U

mit $T_S = 0$ (vgl. auch Abschnitt 8.5.8). Wird die Entscheidung durch eine lokale Komponente herbeigeführt, so ergeben sich T_{UM} und T_{US} mit $T_N = 0$.

Wie in Abschnitt 7.5.4 vorgestellt wurde, ist zumindest für die hier vorgestellten Arten von Migrationsentscheidungen eine derartige Beeinflussung der Prozessausführung nur in den Fällen notwendig, in denen auch tatsächlich eine Verteilung oder eine sonstige Steuerung des Prozesses durchgeführt werden soll. Die Migrationsentscheidung wird hierbei bereits durch das Abonnement des blockierenden Ereignisses getroffen, so dass die Verteilung bei Eintreten des Ereignisses und nach Anhalten der Prozessausführung unverzüglich durchgeführt werden kann (vgl. auch Abbildung 8.13).

8.5.5 Aufwand für die Verteilung von Prozessinstanzen

Kommt es zu einer Anpassung der Prozessausführung in Form einer Migration der Prozessinstanz zu einer anderen Ausführungseinheit, so sind neben den oben genannten Verzögerungen für das Anhalten der Prozessausführung und das Durchführen der Migrationsentscheidung weitere Maßnahmen für den Transfer des Prozesses zu berücksichtigen. Zunächst müssen – falls nicht bereits nebenläufig zur Prozessausführung geschehen – die aktuellen Instanzdaten des Prozesses erhoben und serialisiert werden. Der Aufwand hierfür ist abhängig von der Anzahl der Kontrollflusselemente und Prozessvariablen des Prozessmodells, den Datentypen der Prozessvariablen im Prozessmodell und den konkreten Werten der Prozessvariablen der jeweils betrachteten Prozessinstanz. Es entsteht für die Vorbereitung des Prozesses für den Transfer ferner Aufwand für die Serialisierung von Zuordnungsstrategien und die Anwendung von Sicherheitsmechanismen (hier nicht betrachtet), wobei letztere ebenfalls abhängig von der Anzahl der Kontrollflusselemente und Prozessvariablen sowie den Werten der Prozessvariablen sind. Da der Kontrollfluss für den Export der Instanzdaten nicht erneut navigiert werden muss, steigt der erforderliche Rechen- und Zeitaufwand für die Vorbereitung der Migration weitestgehend linear mit dem Umfang des Prozessmodells (d. h. der Anzahl der Kontrollflusselemente und Prozessvariablen). Hinzu kommt ein vom Umfang der Prozessbeschreibung weitestgehend unabhängiger Aufwand für die Aufbereitung der übrigen Migrationsdaten, welcher bei weniger umfangreichen Prozessen stärker ins Gewicht fällt als bei Prozessen mit einer hohen Anzahl an Kontrollflusselementen und Prozessvariablen (vgl. Abbildung 8.14). Zusätzlich entsteht ein anwendungsabhängiger Rechen- und Zeitaufwand für die Serialisierung von Variablenwerten, welcher insbesondere bei umfangreichen Objekten in komplexen Datenstrukturen oder bei Binärdateien gesondert zu berücksichtigen ist. Zum Beispiel entsteht für die zusätzliche Verarbeitung einer in Base64 kodierten Bilddatei mit einer Größe von 60 KB als Wert einer Prozessvariablen des in Abbildung 8.14 dargestellten Prozesses (*Prozess 1*) anstelle der dargestellten 15 ms für die Verarbeitung eines einstelligen Wertes vom Typ Integer für die Vorbereitung der Migration bereits ein Zeitaufwand von insgesamt 110 ms.

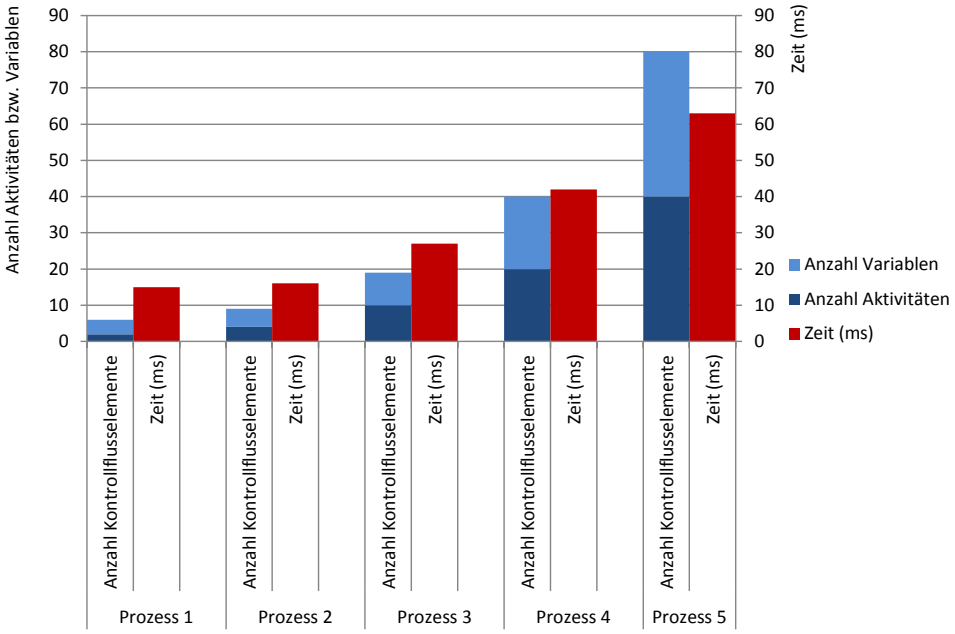


Abbildung 8.14: Entwicklung der Rechenzeit zur Vorbereitung der Migration in Abhängigkeit der Anzahl von Aktivitäten und Prozessvariablen verschiedener XPDL-Prozessmodelle (Notebook mit 1,3 GHz und Pentium M Prozessor)

Im weiteren Verlauf der Verteilung besteht (unabhängig von der Prozessinstanz) Aufwand für das dynamische Auffinden von Ausführungseinheiten sowie die Auswahl einer geeigneten Zielausführungseinheit. Der Aufwand ist abhängig von der Anzahl potentiell in Frage kommender Ausführungseinheiten sowie der Komplexität der Auswahlentscheidung. Kann keine geeignete Ausführungseinheit aufgefunden werden, so entsteht zudem eine Verzögerung der Prozessausführung, da diese unter Umständen (temporär) nicht fortgesetzt werden kann. Weiterer Zeitaufwand entsteht durch das Senden des Prozesses über das Netzwerk, wobei abhängig von der Übertragungsbandbreite auch die Belastung des Netzwerks durch die Übertragung des Prozesses berücksichtigt werden muss. Bei einer Migration werden immer sowohl das Prozessmodell als auch die Instanzdaten gesendet. Der Umfang der Prozessbeschreibung ist neben der Anzahl an Kontrollflusskonstrukten und Prozessvariablen abhängig von der gewählten Prozessbeschreibungssprache und der zusätzlich enthaltenen Elemente zur Spezifikation von Anwendungsschnittstellen und Prozessteilnehmern. Der Umfang der Instanzdaten ist wiederum abhängig von der Anzahl der Kontrollflusskonstrukte und Prozessvariablen. Hinzu kommt ein variabler Anteil für die Werte der Prozessvariablen, welche je nach Anwendungsfall verschiedenen Umfang annehmen können. In den meisten Fällen sind nicht-komplexe Daten wie kurze Zeichenketten, Zahlenwerte und boolesche Werte zu erwarten. Gerade im Umfeld mobiler Anwendungsbereiche sind jedoch auch Multimediadaten, wie z. B. Fotos, möglich, welche analog zu den zuvor genannten Bear-

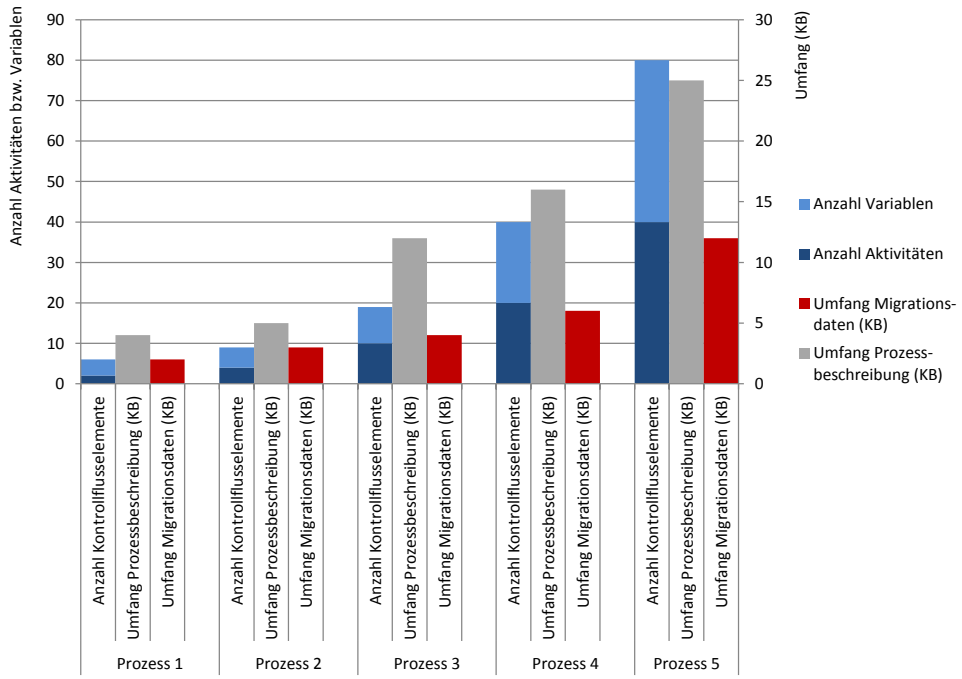


Abbildung 8.15: Entwicklung des Umfangs der Migrationsdaten in Abhängigkeit der Anzahl von Aktivitäten und Prozessvariablen verschiedener XPDL-Prozessmodelle (ohne Berücksichtigung der Werte der Prozessvariablen)

beitungszeiten auch zu erheblich größeren Datenmengen führen. Dabei ist es insbesondere von Nachteil, dass bei der Migration Daten übertragen werden, welche (aufgrund der Unvorhersehbarkeit) auf der Zielausführungseinheit für die Ausführung der zugewiesenen Prozesspartition unter Umständen gar nicht benötigt werden.

Ohne den Anteil für Variablenwerte liegt der Umfang der bei einer Migration zu übertragenden Instanzdaten bei den durchgeführten Tests stets deutlich unter dem Umfang der Prozessbeschreibung des dazugehörigen Prozessmodells (vgl. Abbildung 8.15). Dies gilt ebenfalls für die im Normalfall angenommenen nicht-komplexen Variablenwerten wie kurze Zeichenketten, Zahlenwerte und boolesche Werte. Weitere optionale Bestandteile, wie Programmfragmente, Beschreibungen für Benutzungsschnittstellen (unabhängig von ihrer Repräsentation), Protokolldaten und andere anwendungsabhängige Daten (wie z. B. Daten für Bezahlverfahren, vgl. Abschnitt 8.4.2) müssen bei der Migration ebenfalls inkludiert werden. Der in Abbildung 8.15 dargestellte *Prozess 3* beinhaltet neben dem Prozessmodell z. B. eine Beschreibung für Benutzungsschnittstellen, welche hier dem Umfang der Prozessbeschreibung zugerechnet wird. Auf den Umfang der Migrationsdaten hat dies keinen Einfluss.

Auf der Empfängerseite (d. h. auf Seiten der Zielausführungseinheit) müssen die erhaltenen Daten nach dem Transfer wiederum auf der lokalen Prozess-Engine installiert werden, was analog zur Vorbereitung der Migration vom Umfang der Prozessbeschreibung und der übertragenden Instanz-

daten abhängig ist. Es ergeben sich daher zusammenfassend die folgenden Abhängigkeiten:

Sei T_{UM} der in Abschnitt 8.5.4 genannte Zeitaufwand für das Anhalten der Prozessbeschreibung und die Durchführung der Migrationsentscheidung, T_V der vom Umfang der Prozessbeschreibung und der Variablenwerte abhängige Zeitaufwand zur Vorbereitung der Migration, T_F der Zeitaufwand für das Auffinden und die Auswahl einer geeigneten Zielausführungseinheit, T_D der Zeitaufwand für die Übertragung der genannten Daten, T_I der ebenfalls vom Umfang abhängige Aufwand zur Reinstallation des Prozesses auf der Zielausführungseinheit und T_W der in Abschnitt 8.5.4 genannte Zeitaufwand für das Wiederaufsetzen der angehaltenen Prozessinstanz. Dann ist die gesamte Reaktionszeit für die Anpassung der Verteilung definiert durch $T_R = T_M + T_V + T_F + T_D + T_I + T_W$.

Der dargestellte Aufwand für die Verteilung des Prozesses ist (ungeachtet sich verändernder Variablenwerte) bei jedem Verteilungsschritt der betrachteten Prozessinstanz identisch, so dass die in Abschnitt 8.1 genannten Bedingungen für eine fortwährende Anpassbarkeit der Prozessausführung erfüllt sind. Der durch die Verteilung insgesamt entstehende Aufwand ist schließlich abhängig von der Anzahl der Migrationen, welche vom Beginn bis zum Ende der Prozessausführung durchgeführt werden. Damit ist der Aufwand für eine vollverteilte Prozessausführung am höchsten, während bei einer zentral ausgeführten Prozessinstanz kein zusätzlicher Aufwand entsteht. Da der hier beschriebene Aufwand für die verteilte Ausführung jeder Prozessinstanz erneut anfällt, ist bei der Auswahl des zu anzuwendenden Verteilungsmodells zudem die Anzahl der voraussichtlich verschiedenartig auszuführenden Prozessinstanzen zu berücksichtigen. Dieser Aspekt wird in Anschnitt 8.5.7 erneut aufgegriffen.

8.5.6 Umfang des Anpassungsspielraums

Der Umfang des Anpassungsspielraums bezeichnet nach GOLDEN und POWELL [GP00] die *Vielfältigkeit an Möglichkeiten*, welche einem System zur Verfügung stehen, um auf erwartete Situationen reagieren zu können. Analog bezeichnet die *Robustheit* den Grad der Anpassungsfähigkeit an unerwartete Situationen (vgl. Abschnitt 8.1).

Für die reine Durchführung der Verteilung ist es bei dem hier vorgestellten Lösungsansatz nicht von Bedeutung, ob die Notwendigkeit zur Anpassung bereits im Voraus bekannt war oder erst zur Laufzeit des Prozesses aufgetreten ist. Da jede lokal ausgeführte Prozessinstanz durch den Aufruf der bereitgestellten Managementfunktionalitäten zu jedem Zeitpunkt auch spontan überwacht und in der Art ihrer Ausführung angepasst werden kann, kann sowohl auf erwartete als auch auf unerwartete Situationen reagiert werden. Die Ergebnisse der Untersuchung der Migration von Prozessinstanzen unterschiedlicher standardisierter Prozessbeschreibungssprachen (vgl. Abschnitt 8.3) lassen darauf schließen, dass eine spontane Vertei-

lung der Prozessausführung zur Laufzeit des Prozesses möglich ist, sofern dabei eine Reihe von Einschränkungen berücksichtigt werden, um die fachlich korrekte Ausführung des Prozesses zu erhalten. Diese Einschränkungen begrenzen den für Anpassungen zur Verfügung stehenden Handlungsspielraum jedoch unabhängig von der Vorhersehbarkeit des Anpassungsbedarfs.

Erwartete Situationen können zudem in Form von benutzerdefinierten Rahmenbedingungen oder lokalen Richtlinien der betrachteten Ausführungseinheit von vornherein automatisch behandelt werden. Unerwartete Situationen können entweder ebenfalls über (abstrakt definierte) Verteilungsstrategien oder erweiterte Management-Regeln aufgefangen werden oder können durch eine Erzeugung oder Anpassung der benutzerdefinierten Rahmenbedingungen oder lokalen Richtlinien gezielt behandelt werden. Ob das Eingreifen dabei manuell oder automatisch geschieht, ist im Wesentlichen von der Möglichkeit zur Erhebung von Informationen und zur (automatischen oder manuellen) Ableitung von Anpassungsbedarf abhängig.

Die Erhebung von Informationen ist durch ihre Verfügbarkeit und die Bereitschaft zur Bereitstellung begrenzt. Alle im Rahmen des Prozessmanagementsystems als verwaltbare Ressource bereitgestellten Informationen können dabei (geplant oder ungeplant) jederzeit zugegriffen werden, auch wenn eine Überwachung der betrachteten Prozessinstanz zur Entwicklungs- oder Verteilungszeit nicht beabsichtigt gewesen ist. Dabei ist auch die Anpassung der Datenerhebung selbst während der Laufzeit des Prozesses bzw. der Datenerhebung möglich, z. B. indem Abonnements für Ereignisse dynamisch nachgefragt oder aufgehoben werden.

Probleme entstehen jedoch immer dann, wenn die benötigten Informationen für eine Anpassung zur Laufzeit nicht dynamisch zugegriffen werden können (vgl. z. B. Abschnitt 8.4.5). Ist die Notwendigkeit der Erhebung solcher Informationen vor Beginn der Ausführung bekannt, so kann unter Umständen durch die Anpassung des fachlichen Prozessmodells und/oder einer Erweiterung der Ausführungsumgebung Abhilfe geschaffen werden. Ist jedoch nicht bekannt, welche Informationen benötigt werden, um Anpassungsbedarf abzuleiten, so ist naturgemäß auch kein automatisierter Anpassungsvorgang möglich. Es bleibt jedoch in diesem Fall die Möglichkeit des manuellen Ableitens von Anpassungsbedarf und einer entsprechenden Anpassung der Prozessausführung auf Basis der in dieser Arbeit vorgeschlagenen Konzepte.

Zusammenfassend ergeben sich in Bezug auf den zur Verfügung stehenden Anpassungsspielraum die folgenden Ergebnisse:

- Eine Verteilung des Prozesses ist – abhängig von der verwendeten Prozessbeschreibungssprache – unter der Berücksichtigung der Untrennbarkeit bestimmter Kontrollflusskonstrukte jederzeit zur Laufzeit der Prozessinstanz möglich. Granularität und Zielorte der Verteilung können dabei dynamisch zum Zeitpunkt jeder Verteilung für die noch nicht ausgeführte Prozesspartition (neu) festgelegt werden.

- Eine Überwachung des Prozesses ist – abhängig von der Verfügbarkeit der relevanten Informationen – jederzeit zur Laufzeit der Prozessinstanz möglich. Die Art der Überwachung und die Art und Anzahl der relevanten Informationen können dabei dynamisch zu jedem Zeitpunkt der Überwachung (neu) festgelegt werden.

- Eine Automatisierung des gesamten Anpassungsvorgangs ist nur dann vollständig möglich, wenn der Zusammenhang zwischen Informationen, Anpassungsbedarf und Anpassung vor dem Eintreten der als Auslöser der Anpassung eintretenden Situation bekannt ist.

Im Gegensatz zu anderen Arten des Umgangs mit verteilt ausgeführten Prozessen besteht die Flexibilität in dem hier vorgestellten Ansatz *a-priori* (vgl. Anschnitt 3.3). Dies bedeutet, dass bei der Ausführung jeder Prozessinstanz prinzipiell das gesamte Verteilungsspektrum offen steht und dieses bei Bedarf von den an der Prozessausführung beteiligten Parteien sinnvoll eingeschränkt werden kann. Dazu muss jede *Einschränkung* explizit modelliert werden. Andere Ansätze besitzen diese A-priori-Flexibilität in der Regel nicht, so dass hier jede *Handlungsalternative* explizit modelliert werden muss. Ist die Handlungsalternative zur Laufzeit des Prozesses nicht vorhanden, muss eine *A-posteriori-Anpassung* mit erhöhtem Aufwand vorgenommen werden. Im Folgenden wird diese wichtige Unterscheidung am Beispiel eines (theoretischen) Vergleichs mit der physischen Partitionierung von Prozessen verdeutlicht.

8.5.7 Vergleich mit physischer Partitionierung von Prozessen

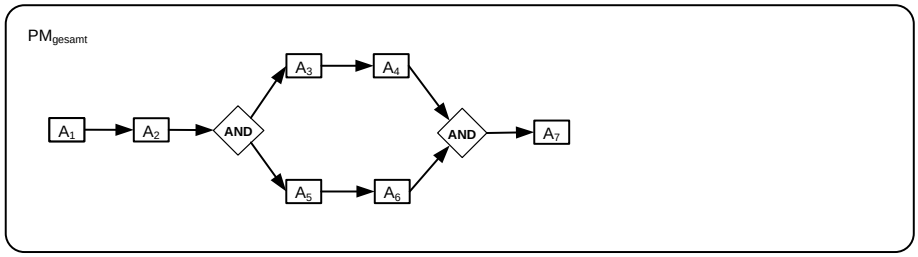
Unter Berücksichtigung der Untrennbarkeit bestimmter Kontrollflusskonstrukte kann durch die Migration von Prozessinstanzen zur Laufzeit des Prozesses eine beliebige Granularität der Verteilung und eine beliebige Zuordnung der Prozesspartitionen zu Ausführungseinheiten erreicht werden, ohne dass die Prozessausführung außerhalb der Migration hierdurch beeinträchtigt wird. Ein statisch verteilter Prozess, bei dem durch die physische Partitionierung des Prozessmodells (vgl. Abbildung 8.16a) einzelne Prozesspartitionen an bestimmte Ausführungseinheiten zugewiesen werden und durch das Einfügen zusätzlicher Aktivitäten und Schnittstellen eine Übergabe des Kontrollflusses hergestellt wird (vgl. Abbildung 8.16b), besitzt diesen Anpassungsspielraum zunächst einmal nicht. Soll für die Ausführung einer einzelnen Prozessinstanz nun eine andere Verteilungskonfiguration gewählt werden, so ist eine *A-posteriori-Anpassung* der Prozessmodelle der relevanten verteilten Partitionen notwendig. Hierzu kann entweder das Prozessmodell jeder betreffenden Partition temporär modifiziert werden oder eine neue Version der Prozesspartitionen verteilt werden. In jedem Fall ist dabei eine inhaltliche Veränderung des (verteilten) Prozessmodells notwendig, da die Aktivitäten und Schnittstellen zur Übergabe des Kontrollflusses bei einer abweichenden Aufteilung des Prozesses neu erzeugt, gelöscht oder verändert und anschließend neu in den Prozess eingeordnet werden müssen (vgl. Abbildung 8.16c). Der Auf-

wand für eine solche Re-Partitionierung steigt in beiden Fällen mit der Anzahl der betroffenen Prozesspartitionen und der Anzahl der teilnehmenden Ausführungseinheiten, wobei alle betroffenen Ausführungseinheiten zur Anpassung kontaktiert werden müssen. Nach der Ausführung der abweichenden Prozessinstanz entsteht zudem Aufwand für die Rückführung der Prozesspartitionen auf die ursprüngliche Version.

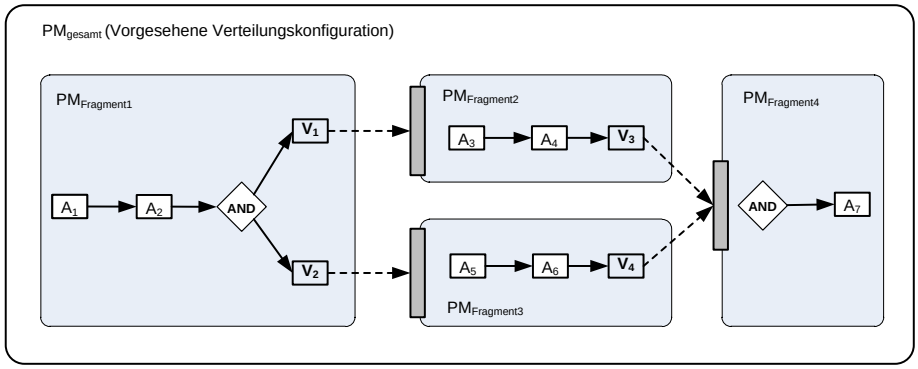
Die hier beschriebene Vorgehensweise ist bei einer *A-posteriori-Anpassung* für jede Prozessinstanz mit abweichender Verteilungskonfiguration durchzuführen. Sofern kein zusätzliches System zur Versionierung von Prozessmodellen zur Verfügung steht, müssen zunächst alle laufenden Prozessinstanzen der ursprünglichen Verteilungskonfiguration den kritischen Abschnitt der Anpassung durchlaufen haben oder vor dem kritischen Abschnitt angehalten werden, bevor die Re-Partitionierung durchgeführt werden kann. Die Ausführung der abweichend verteilt ausgeführten Prozessinstanz muss entsprechend warten, bis die genannten Vorbereitungsmaßnahmen abgeschlossen sind. Gleiches gilt entsprechend für die Rückführung der Prozesspartitionen auf die ursprüngliche Version. Es kommt daher hierbei neben einer Verzögerung der abweichend verteilt auszuführenden Prozessinstanz auch zu einer negativen Beeinflussung der anderen (normal) ausgeführten Prozessinstanzen desselben Prozessmodells. Prozessinstanzen anderer Prozessmodelle werden nicht beeinträchtigt. Eine A-posteriori-Anpassung der Prozessausführung durch physische Partitionierung des Prozessmodells besitzt folglich eine schlechte Effizienz der Anpassung und kann daher in dynamischen Umgebungen als nicht vorteilhaft angesehen werden.

Um mit physischer Partitionierung von Prozessen eine ähnliche *A-priori-Flexibilität* der Verteilung zu erzielen wie bei der hier vorgestellten Migration von Prozessinstanzen, müsste jedes Prozessmodell an jeder möglichen (erlaubten) Position im Kontrollfluss partitioniert, mit Schnittstellen für die Übergabe des Kontrollflusses versehen werden und die resultierenden Fragmente auf alle potentiell für die Ausführung in Frage kommenden Ausführungseinheiten verteilt werden. Im einfachsten Fall kann dies durch eine Vollverteilung des Prozesses durch die Bildung der maximalen Anzahl von Partitionen erreicht werden (vgl. Abbildung 8.16d). Hierbei müssen für eine Anzahl von n Aktivitäten in einem sequentielle Prozessmodell wenigstens $n - 1$ Aktivitäten zur Steuerung der Verteilung eingefügt werden, welche sich bei Verzweigungen ggf. weiter erhöht. Wird eine Prozessinstanz zentral ausgeführt, müssen die Aktivitäten zur Steuerung der Verteilung trotzdem durchlaufen werden, so dass in diesem Fall fast doppelt so viele Aktivitäten navigiert werden müssen, wie im ursprünglichen Prozessmodell angegeben wurden. Zudem wird der Umfang der gesamten Prozessbeschreibung hierdurch stark erhöht, was sich ebenfalls auf den Umfang der gesamten zu übertragenden Datenmenge bei der Zuweisung von Prozesspartitionen an die ausgewählten Ausführungseinheiten auswirkt.

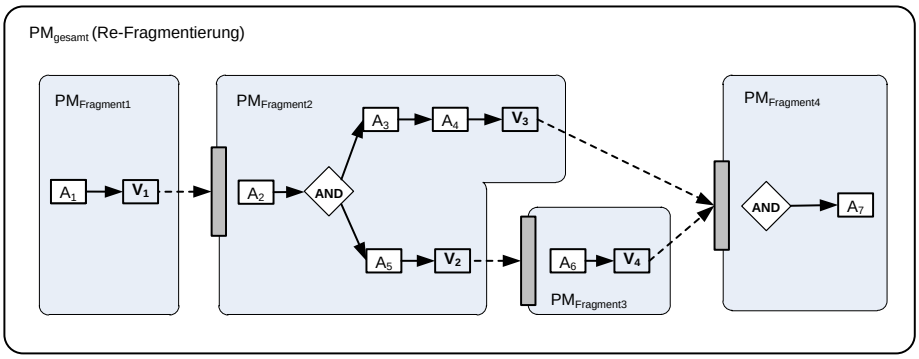
Bei einer verteilten Ausführung soll zudem zwischen verschiedenen Ausführungseinheiten gewählt werden können, so dass an jeder Aktivität



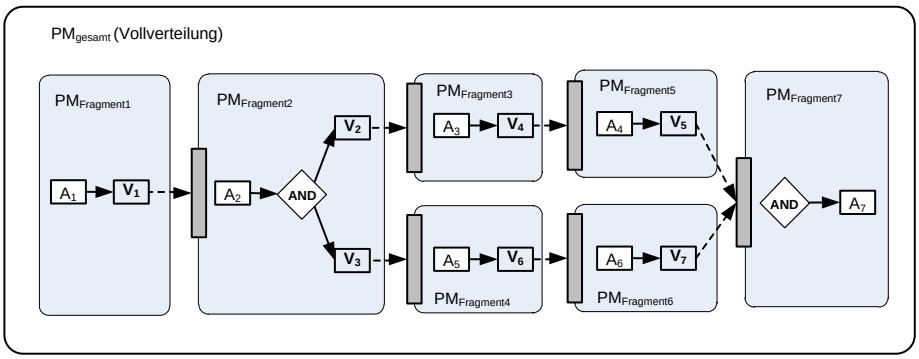
(a) Beispielprozess ohne physische Partitionierung des Prozessmodells PM_{gesamt}



(b) Exemplarische physische Partitionierung des Prozessmodells PM_{gesamt}

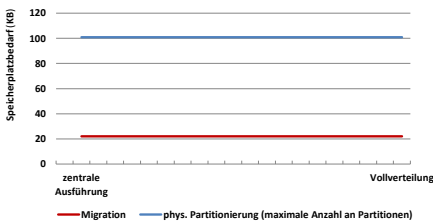


(c) A-posteriori-Anpassung: Re-Partitionierung des Prozessmodells PM_{gesamt}

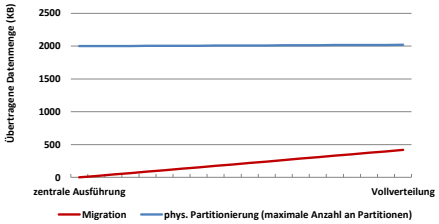


(d) A-priori-Anpassung: Vollverteilung des Prozessmodells PM_{gesamt} durch maximale Partitionierung

Abbildung 8.16: Physische (Re-)Partitionierung von Prozessen am Beispiel des Prozessmodells PM_{gesamt} mit den fachlichen Aktivitäten A_i und den Aktivitäten zur Steuerung der Verteilung V_i (Übergabe des Kontrollflusses)



(a) Speicherplatzbedarf pro Ausführungseinheit in Abhängigkeit der Anzahl an Verteilungsschritte bzw. Prozesspartitionen (für Beispielprozess PM_4)



(b) Gesamter Umfang zu übertragender Daten für die Ausführung einer Prozessinstanz in Abhängigkeit der Anzahl an Verteilungsschritte bzw. Prozesspartitionen (für Beispielprozess PM_4)

Abbildung 8.17: Vergleich von Migration von Prozessinstanzen und physischer Partitionierung von Prozessen: Speicherplatzbedarf und Umfang zu übertragender Daten bei gleichem Anpassungsspielraum am Beispiel von Prozess PM_4 (vgl. Abbildung 8.15)

zur Steuerung der Verteilung entschieden werden können muss, welche Ausführungseinheit die Prozessausführung fortsetzen soll. Bei einer physischen Partitionierung muss also jede Ausführungseinheit die relevanten Prozesspartitionen besitzen, um diese im Bedarfsfall ausführen zu können. Dies führt wiederum zu dem Umstand, dass Prozesspartitionen zur Entwicklungszeit an Ausführungseinheiten verteilt werden, welche zur Laufzeit unter Umständen gar nicht an der Ausführung teilnehmen. Die Anzahl der nicht durchlaufenden Prozesspartitionen erhöht sich folglich mit der Anzahl von Ausführungseinheiten, welche potentiell an der Prozessausführung teilnehmen können sollen. Der je Ausführungseinheit für die Datenhaltung erforderliche Speicherplatz ist gegenüber der Migration von Prozessinstanzen nur dann reduziert, wenn nicht alle physischen Partitionen des ursprünglichen Prozessmodells permanent im Speicher gehalten werden müssen. Die Situation verschärft sich jedoch, wenn für jedes Prozessmodell alle möglichen Partitionen auf allen (potentiell teilnehmenden) Ausführungseinheiten gehalten werden müssen. In diesem Fall ist die genannte Vorgehensweise – bei gleichem Anpassungsspielraum für die verteilte Ausführung – erst ab einer gewissen Anzahl von verteilt ausgeführten Prozessinstanzen je Prozessmodell in Bezug auf die zu übertragende Datenmenge vorteilhafter als die Migration von Prozessinstanzen.

Abbildungen 8.17, 8.18 und 8.19 zeigen die Ergebnisse einer Evaluation des Speicherplatzbedarfs und des Umfangs der für eine (potentiell) verteilte Ausführung zu übertragenden Daten in Abhängigkeit der Anzahl der Verteilungsschritte, der Anzahl an Prozessinstanzen desselben Prozessmodells und der Anzahl an (potentiell) teilnehmenden Ausführungseinheiten. Für die Erhebung der Daten wurde exemplarisch das bereits in Abschnitt 8.5.5 dargestellte Prozessmodell PM_4 zugrunde gelegt. Das Prozessmodell PM_4 ist in der Prozessbeschreibungssprache XPDL modelliert und besitzt 20 sequentiell angeordnete Aktivitäten, welche nacheinander eine von 6 unterschiedlichen, in der Prozessbeschreibung spezifizierten Anwendungsfunktionalitäten zugrei-

fen. Hierbei werden 20 unterschiedliche Prozessvariablen vom Typ Integer und String lesend und schreibend zugegriffen. Die Prozessbeschreibung ist insgesamt 16 KB groß. Bei einer Migration des Prozesses werden Migrationsdaten in der Größe von 6 KB erzeugt (vgl. Abbildung 8.15). Die bei jeder Migration übertragene Datenmenge, welche ebenfalls während der Ausführung von der jeweils verantwortlichen Ausführungseinheit gespeichert werden muss, beträgt daher 22 KB. Für eine Vollverteilung des Prozesses müssen daher bei Verteilung der Prozessausführung durch Migration nach dem hier vorgestellten Ansatz 418 KB übertragen werden (Worst-Case-Szenario mit 19 Verteilungsschritten).

Für einen Vergleich mit der physischen Partitionierung von Prozessen wurde das Prozessmodell PM_4 in 20 Partitionen zerlegt. Jeder Partition außer der letzten wurde eine weitere Aktivität hinzugefügt, um die nächste Partition aufrufen zu können (vgl. Aktivitäten V_i in Abbildung 8.16d). Folglich besteht nun jede Partition aus einer fachlichen Aktivität und einer Steuerungsaktivität. Für die fachliche Aktivität ist zudem die jeweils aufzurufende Anwendungsfunktionalität in der Prozessbeschreibung enthalten. Für die Steuerungsaktivität wird auf einen Dienst verwiesen, welcher die Verteilungsentscheidung durchführt und den Zielort der als nächstes aufzurufenden Ausführungseinheit zurückgibt. Für die fachliche Aktivität und die Steuerungsanweisung werden zusammen 4 Prozessvariablen pro Prozesspartition benötigt. Jede der 20 Prozesspartitionen (außer der letzten) besitzt daher nun einen Umfang von durchschnittlich 5 KB. Da auch bei der physischen Partitionierung zur Laufzeit Instanzdaten in Form des Aufrufs der nächsten Ausführungseinheiten übertragen werden müssen, werden pro Verteilungsschritt weitere 1 KB veranschlagt. Aufgrund des hohen Skalenniveaus ist der Aufwand für die Übertragung der Instanzdaten den Abbildungen 8.17 bis 8.19 jedoch nicht direkt zu entnehmen.

Um zur Laufzeit denselben Anpassungsspielraum für eine Verteilung des Prozesses zu erhalten, wurden für die verteilte Prozessausführung durch physische Partitionierung alle wie beschrieben ausgestatteten Partitionen an alle verfügbaren Ausführungseinheiten verteilt. Da eine Vollverteilung bei 20 fachlichen Aktivitäten 20 Ausführungseinheiten erfordert und für den Vergleich jede Ausführungseinheit in der Lage sein muss, jede mögliche Partitionierung des Prozesses zu bearbeiten, entsteht bei Partitionen von je etwa 5 KB hierfür ein Speicherplatzbedarf von etwa 100 KB pro Ausführungseinheit (vgl. Abbildung 8.17a). Durch die Notwendigkeit, die Prozesspartitionen einmalig an alle Ausführungseinheiten zu senden, entsteht insgesamt ein Aufwand von etwa 2000 KB Übertragungsvolumen (vgl. Abbildung 8.17b). Im Gegensatz zu der Migration fällt dieser Aufwand auch an, wenn die Prozessausführung schließlich doch zentral erfolgt oder wenn gar kein Prozess ausgeführt wird.

Wie Abbildung 8.17b weiter zu entnehmen ist, ist der Übertragungsaufwand für eine einzelne Prozessinstanz sogar bei Vollverteilung (Worst-Case der Migration) bei der physischen Partitionierung deutlich höher als bei der Migration (ungeachtet der Möglichkeit von Kompensationen und Mehrfach-

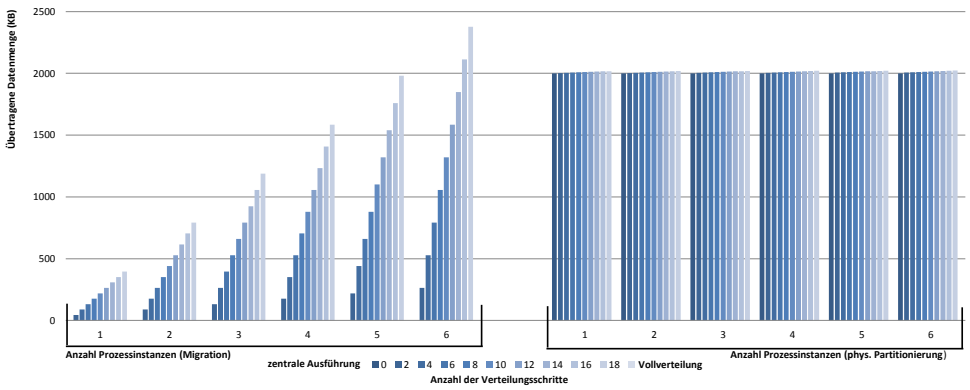


Abbildung 8.18: Vergleich von Migration von Prozessinstanzen und physischer Partitionierung von Prozessen: Gesamter Umfang zu übertragender Daten für Beispielprozess PM_4 bei gleichem Handlungsspielraum (mehrfache Ausführung, 20 Ausführungseinheiten)

ausführungen von Aktivitäten). Für einen nur einmalig verteilt ausgeführten Prozess stellt die Partitionierung bei Unvorhersehbarkeit der Verteilungskonfiguration daher die schlechtere Lösung dar. Betrachtet man jedoch, dass die Distribution der Prozesspartitionen bei der physischen Partitionierung pro Prozessmodell nur einmalig erfolgen muss und zur Laufzeit für jede weitere verteilt ausgeführte Prozessinstanz nur noch die geringeren Instanzdaten ausgetauscht werden müssen, so verursacht der Verteilungsansatz der Migration nach einer gewissen Anzahl von Prozessinstanzen desselben Prozessmodells einen höheren Übertragungsaufwand, insbesondere wenn es zu einer hohen Anzahl an Verteilungsschritten (z. B. durch Vollverteilung) kommt (vgl. Abbildung 8.18). Bei der Vollverteilung von PM_4 ist dies bei 20 beteiligten Ausführungseinheiten ab der sechsten Prozessinstanz der Fall.

Bis hierher wurde zudem angenommen, dass die Anzahl der potentiellen Ausführungseinheiten konstant ist. Abbildung 8.19 zeigt den Übertragungsaufwand für den Fall, dass eine beliebige Menge an Ausführungseinheiten an der Ausführung der Prozessinstanzen teilnehmen kann. Hierbei wird bei der physischen Partitionierung ersichtlich, dass der für jede weitere (potentiell) zu integrierende Ausführungseinheit erforderliche Übertragungsaufwand unabhängig von der ausgeführten Menge an Prozessinstanzen und den hierfür durchgeführten Verteilungsschritten steigt, während bei der Migration nach der Ausschöpfung aller möglichen Verteilungsschritte unabhängig von der Menge der Ausführungseinheiten kein weiterer Aufwand entsteht. Hieraus lässt sich insbesondere ableiten, dass in dynamischen Umgebungen, in denen die an der Prozessausführung potentiell teilnehmenden Ausführungseinheiten vor der Laufzeit des Prozesses unbekannt sind, die Verteilung des Prozesses durch Migration wiederum Vorteile bietet. Dasselbe gilt in Umgebungen, in denen aufgrund der Netzgröße mit einer großen Anzahl an potentiellen Ausführungseinheiten zu rechnen ist (z. B. im Internet).

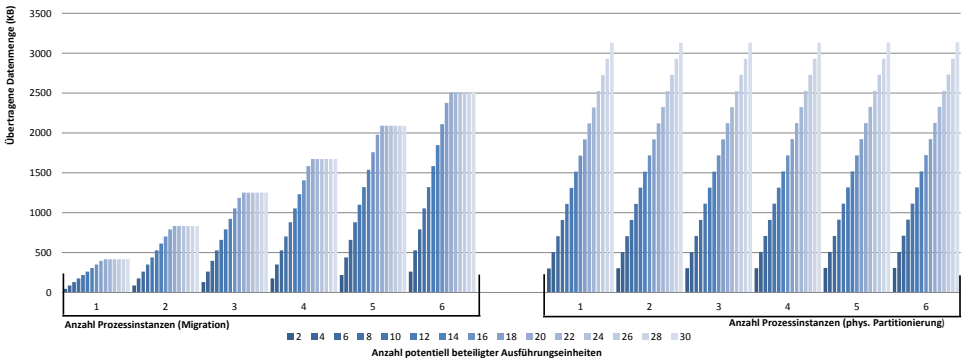


Abbildung 8.19: Vergleich von Migration von Prozessinstanzen und physischer Partitionierung von Prozessen: Gesamter Umfang zu übertragender Daten für Beispielprozess PM_4 in Abhängigkeit zur Verfügung stehender Ausführungseinheiten (mehrfache Ausführung, 30 Ausführungseinheiten)

Zusammenfassend lässt sich festhalten, dass für den Ansatz der physischen Partitionierung ein relativ großer Aufwand betrieben werden muss, um durch A-priori-Flexibilität einen ähnlichen Anpassungsspielraum zu erreichen, wie er durch den hier vorgeschlagenen Ansatz der dynamischen Verteilung durch die Migration von Prozessinstanzen bereitgestellt wird. Als wesentlicher Aspekt ist zu nennen, dass insbesondere bei einer zentraler Ausführung des Prozesses oder einer Verteilung mit wenig Verteilungsschritten eine unnötige Verzögerung durch die in den Prozess integrierten Aktivitäten zur spontanen Durchführung von Verteilungsenscheidungen zu erwarten ist. Dies widerspricht vor allem der Anforderung nach einer möglichst unbeeinflussten zentralen Ausführung außerhalb der Verteilung des Prozesses, welche durch den Ansatz der Migration erfüllt wird. Der Preis dieser höheren Flexibilität besteht jedoch in einem höheren Übertragungsvolumen ab einer gewissen Anzahl von verteilt ausgeführten Prozessinstanzen mit einer hohen Anzahl von Verteilungsschritten. Insbesondere in Ausführungsumgebungen mit einer starken Fluktuation von Ausführungseinheiten, einer zur Entwicklungszeit unvorhersehbaren Verteilungskonfiguration und einer geringen Menge an Verteilungsschritten ist die Verteilung durch Migration als vorteilhaft anzusehen, da ansonsten unnötig viele Partitionen vorverteilt werden müssten, um alle Ressourcen bei Bedarf ausschöpfen zu können. Da sich die Stärken beider Ansätze komplementär zueinander verhalten, sollte daher möglichst eine Koexistenz beider Möglichkeiten in Betracht gezogen werden, um abhängig von den genannten Parametern das geeignete Verteilungsmodell auswählen zu können.

8.5.8 Betrachtung ergänzender Lösungsansätze

Die als Ergänzung der hier vorgeschlagenen Konzepte und Realisierungsmöglichkeiten vorgestellten Ansätze zur regel- und ereignisbasierten Überwachung und zur proaktiven Anpassung der Prozessausführung durch Vorhersage von Kontextdaten wurden bei der bisherigen Aufwands- und Fle-

xibilitätsbetrachtung nicht berücksichtigt. Im Folgenden werden die beiden Ansätze hinsichtlich der in Abschnitt 8.1 definierten Bewertungskriterien und den sich aus ihrer Spezialisierung ergebenden zusätzlichen Aufwands- und Flexibilitätserscheinungen diskutiert.

Regel- und ereignisbasiertes Management

Die von einer etwaigen Verteilung des Prozesses unabhängige regel- und ereignisbasierten Überwachung (vgl. Abschnitte 6.3.5 bzw. 7.4) stellt eine Ergänzung des Prozessmanagementsystems als verwaltbare Ressource dar, welche entweder auf Seiten des Konsumenten einer entfernt ausgeführten Prozesspartition oder auf Seiten des prozessausführenden Systems installiert werden kann. Eine Verwendung der Komponente auf Seiten des Konsumenten einer entfernt ausgeführten Prozesspartition (*Online-Management*) führt bei einer passiven Überwachung (d. h. bei Bezug von ausschließlich nichtblockierenden Ereignissen) zu keinem zusätzlichen Aufwand auf Seiten des prozessausführenden Systems (vgl. Abschnitt 8.5.4). Werden Ereignisse jedoch als blockierend abonniert und/oder soll auch steuernd auf die entfernte Prozessausführung eingewirkt werden, so entsteht hierfür der unter Abschnitt 8.5.4 definierte Zeitaufwand T_{US} mit spezieller Berücksichtigung der Nachrichtenlaufzeit T_N, da die Prozessausführung angehalten werden muss, bis die beobachtende Partei den Eingriff beendet hat, die Prozessausführung wieder frei gibt oder die Zeitbeschränkung für das blockierende Ereignis abgelaufen ist. Eine Verwendung der Komponente auf Seiten der prozessausführenden Partei (*Offline-Management*) führt zu einer analogen Beeinflussung der überwachten Prozessinstanz, wobei die Nachrichtenlaufzeit T_N jedoch nicht berücksichtigt werden muss ($T_N = 0$). Da die zusätzliche Management-Komponente lediglich die Schnittstellen der verwaltbaren Ressource nutzt, ist eine spezielle Anpassung auf das jeweilige zugrunde liegende Prozessmanagementsystem nicht notwendig. Es entsteht daher zur Entwicklungszeit des Prozessmanagementsystems hierfür (in beiden Fällen) kein zusätzlicher Aufwand. Zu erwähnen ist jedoch, dass der Einsatz der regel- und ereignisbasierten Überwachung auf der Seite der prozessausführenden Partei mit Regeln und Ereignisstrukturen des Konsumenten der entfernten Prozessausführung einen zusätzlichen Abstimmungsaufwand zwischen den teilnehmenden Parteien erfordert, welcher im Vorfeld der Prozessausführung stattgefunden haben muss.

Hinsichtlich des Anpassungsspielraums lässt sich festhalten, dass bei einem Offline-Management nur der Umgang mit vorhersehbaren Situationen unterstützt werden kann, da der gesamte Anpassungsvorgang von der Erhebung der Informationen bis zur Durchführung der Anpassung für den Zeitraum der Offline-Ausführung fest determiniert ist. Da bei Vorliegen einer Kommunikationsverbindung zur prozessausführenden Partei oder bei einem Online-Management prinzipiell auch die zu überwachenden Regeln zur Laufzeit hinzugefügt, gelöscht oder modifiziert werden können, ist auch ein Umgang mit

unvorhersehbaren Ereignissen auf der Seite des Beobachters möglich. Für die Automatisierbarkeit dieser Anpassungen gelten die in Abschnitt 8.5.6 genannten Einschränkungen.

Strukturierte Kontextdatenprognose

Eine proaktive Anpassung der Prozessausführung durch die Prognose von Ausführungskontexten (vgl. Abschnitte 6.5 bzw. 7.7) richtet sich an Anwendungsfälle, welche einen speziellen Bedarf zur Verbesserung von Migrationsentscheidungen aufweisen. Dabei ist die Vorhersage von Kontextdaten, wie etwa zur Verfügbarkeit der für die Prozessausführung benötigten Ressourcen (vgl. z. B. Abschnitt 8.4.1), weitestgehend unabhängig von der eigentlichen Ausführung der Prozesse. Lediglich durch das Einbringen von Anwendungswissen auf Instanzebene kommt es zu einer (anwendungsabhängigen) Erhöhung des Umfangs der Migrationsdaten, da das Anwendungswissen bei jeder Verteilung des Prozesses weitergegeben werden muss. Die Dauer der Prognose ist dem in Abschnitt 8.5.5 definierten Zeitaufwand T_F für das Auffinden bzw. die Auswahl einer geeigneten Zielausführungseinheit zuzurechnen. Dabei ist jedoch zu berücksichtigen, dass eine Vorhersage von Kontexten unter Umständen nur dann erfolgt, wenn bereits im ersten Versuch keine direkt geeignete Zielausführungseinheit aufgefunden werden konnte. Kann durch die Prognose von Kontextdaten in diesem Fall eine (voraussichtlich) geeignete Ausführungseinheit bestimmt werden, so ist auf der anderen Seite auch eine Reduzierung von T_F möglich, da unter Umständen ein wiederholter Suchvorgang oder das Warten auf das Auffinden weiterer geeigneter Ausführungseinheiten verhindert werden kann. Ob T_F durch die Durchführung von Prognosen erhöht oder reduziert wird, ist demnach vom Eintreten der durch die Prognose vorhergesagten Ausführungskontexte abhängig.

Betrachtet man den Aufwand für die Entwicklung von Prognosemodellen, so kann festgestellt werden, dass deren Erstellung teilweise anwendungsabhängig und somit potentiell an die Modellierung der später auszuführenden Prozesse geknüpft ist. Es kommt zwar nicht zu einer notwendigen Anpassung von Prozessmodellen, jedoch muss im schlimmsten Fall zur Entwicklungszeit für jedes Prozessmodell ein eigenes Prognosemodell erstellt und an alle voraussichtlich beteiligten Ausführungseinheiten verteilt werden, damit diese mit der Erhebung von Trainingsdaten beginnen können, bevor zu einzelnen Prozessinstanzen Prognosen abgefragt werden können. Die Anwendung der strukturierten Kontextdatenprognose erfordert daher eine längere Vorbereitungszeit, wodurch der Anpassungsspielraum zur Laufzeit der Prozesse auf die zur Entwicklungszeit der Prognosemodelle vorhersehbaren Prognosefälle begrenzt ist.

Durch die im Rahmen dieser Arbeit vorgestellte Prognose von Dienstverfügbarkeiten wurde gezeigt, dass die strukturierte Kontextdatenprognose auch relativ anwendungsunabhängig zur Verbesserung von Migrations-

entscheidungen eingesetzt werden kann. Der für die Ausführungseinheiten zusätzlich zu erbringende Aufwand hinsichtlich Rechenzeit und Speicher wurde bereits in Abschnitt 8.4.1 diskutiert. Da jedoch auch hierbei die Installation und der Betrieb der notwendigen Middleware-Komponente (Prognosesystem), die Verteilung und ggf. Anpassung des Prognosemodells und das Sammeln von Trainingsdaten einen zusätzlichen Aufwand (außerhalb der hier betrachteten Kriterien) verursacht, ist über die Effizienz des Einsatzes der strukturierten Kontextdatenprognose für die proaktive Anpassung der Prozessausführung im Einzelfall zu entscheiden.

8.6 Überprüfung anhand des Anforderungskataloges

In Abschnitt 4.4 wurde eine Anforderungsanalyse für prozessorientierte Anwendungen durchgeführt, welche auf Basis aktueller Umgebungsbedingungen dynamisch und individuell verteilt ausgeführt werden sollen. In diesem Abschnitt wird zusammengefasst, inwieweit diese Anforderungen durch den in dieser Arbeit vorgeschlagenen integrierten Lösungsansatz erfüllt werden. Zunächst werden dazu die in Abschnitt 4.4.1 betrachteten allgemeinen Anforderungen an eine verteilte Ausführung von Prozessen überprüft. Im Anschluss werden die in Abschnitt 4.4.2 beschriebenen Anforderungen an die verteilte Ausführung von Prozessen in Hinblick auf eine dynamische Anpassung der Verteilung, Überwachung und Steuerung individueller Prozessinstanzen genauer betrachtet.

8.6.1 Allgemeine Anforderungen verteilt ausgeführter Prozesse

Für eine verteilte Ausführung von Prozessen wurde als erste Gruppe von Anforderungen zunächst ein Mindestmaß an Interoperabilität gefordert, um Anweisungen und Daten für die Prozessausführung zwischen verschiedenen Ausführungseinheiten austauschen, interpretieren und auf gleiche Weise verarbeiten zu können (Anforderungsgruppe $\mathbf{A_1}$). Durch den im Rahmen dieser Arbeit vorgeschlagenen Lösungsansatz für eine nicht-invasive Migration von Prozessen standardisierter Prozessbeschreibungssprachen kann die Annahme einer Möglichkeit zur gemeinsam zugrunde gelegten Syntax und Semantik für die Prozessausführung bestätigt werden. Insbesondere werden hierbei keine höheren Anforderungen an ein gemeinsames Verständnis der Teilnehmer über die Prozessbeschreibungssprache und den Anwendungsbereich der Prozessausführung gestellt als bei anderen Verteilungsansätzen (wie etwa der physischen Partitionierung von Prozessen), bei denen ausgewählte Prozesspartitionen für die verteilte Ausführung eines Prozesses ebenfalls auf den ausgewählten Zielausführungseinheiten installiert und ausgeführt werden müssen. Die in Abschnitt 7.5 vorgestellte Middleware stellt dazu geeignete Schnittstellen zur Verteilung des Prozesses durch Zuweisung von Prozesspartitionen zu Ausführungseinheiten, zur Übergabe des Kontrollflusses und zum Austausch von Prozessdaten nach dem abstrakten Migrationsmodell bereit.

Der Ansatz des Prozessmanagementsystems als verwaltbare Ressource bietet weiterhin die benötigten Funktionalitäten zur Bereitstellung und Sammlung von Daten über die Prozessausführung. Ein integrierter Ansatz hierfür wurde im Rahmen des verteilten *Process-Management-as-a-Service*-Szenarios vorgestellt (vgl. Abschnitt 8.4.2).

Die Autonomie der an der Prozessausführung beteiligten Parteien kann durch die Spezifikation von lokalen Richtlinien zur Entgegennahme von Prozessen anderer Ausführungseinheiten und zu deren Ausführung (z. B. durch Priorisierung) individuell berücksichtigt werden (Anforderungsgruppe $\mathbf{A_2}$). Die verteilte Ausführung von Prozessen durch die Migration von Prozessinstanzen ist dabei prinzipiell von einer bestimmten Organisationsstruktur unabhängig und kann sowohl zwischen hierarchisch organisierten als auch zwischen gleichberechtigt agierenden Teilnehmern angewendet werden. Die Auswahl der bereitgestellten Überwachungs- und Steuerungsfunktionalitäten im Rahmen des Prozessmanagementsystems als verwaltbare Ressource ist prinzipiell beliebig auf verschiedene (externe) Konsumenten anpassbar. Da alle für Verteilung, Überwachung und Steuerung relevanten Funktionalitäten als Dienste bereitgestellt werden, kann eine Beschränkung des Zugriffs sowie eine etwaige Verhandlung der Nutzung über bestehende Konzepte für Nutzungsvereinbarungen, Zugriffsbeschränkungen und Rechteverwaltung für dienstorientierte Architekturen im Allgemeinen erfolgen. Eine dezentrale Einsetzbarkeit der Lösungsansätze wurde durch die Anwendung im Rahmen der kontextbasierten Kooperation gezeigt (vgl. Abschnitt 8.4.1). Vertiefende Informationen zur Rechteverwaltung und zum Zugriffsschutz von Diensten auch in mobilen Umgebungen sind der Arbeit von KOTTKE [Kot10] zu entnehmen.

Der im Rahmen dieser Arbeit vorgestellte Lösungsansatz bietet die Möglichkeit zur Einbettung von Verteilungsstrategien als lokale Richtlinien oder als benutzerdefinierte Rahmenbedingungen von Seiten des Prozessmodellierers oder -initiators (Anforderungsgruppe $\mathbf{A_3}$). Die explizite Beschreibung von lokalen Richtlinien oder benutzerdefinierten (komplexeren) Verteilungsstrategien wird in dieser Arbeit nicht vertiefend behandelt. Es werden jedoch exemplarische Gruppen von Verteilungsstrategien (*Assignment Strategies*, vgl. Abschnitte 6.2.2 und 7.5.3) vorgestellt, welche für grundlegende Verteilungsentscheidungen bereits gut geeignet sind und beliebig erweitert werden können. Die Nutzung spezialisierter Verteilungsstrategien erfordert jedoch wiederum ein gemeinsames Verständnis der Syntax und Semantik von Anweisungen zur Berechnung der Migrationsentscheidung und das Vorliegen von Komponenten zu deren Durchführung. Ein vollständiges Beispiel für die Zuweisung von Prozesspartitionen auf Basis von nicht-funktionalen Kriterien kann einer vertiefenden Arbeit [HSZ11] entnommen werden. Ein Auffinden und eine Auswahl von Ausführungseinheiten als Ziel der Verteilung kann wiederum über das zugrunde liegende Dienstkonzept vorgenommen werden (z. B. über einen zentralen oder verteilten Verzeichnisdienst, vgl. auch Abschnitt 7.2).

Gruppe	Anforderung	Bewertung / Kommentar
A_1	Interoperabilität	
	$A_{1.1}$ Syntax und Semantik des auszuführenden Prozesses	✓
	$A_{1.2}$ Verteilung von Prozesspartitionen	✓
	$A_{1.3}$ Übergabe des Kontrollflusses	✓
	$A_{1.4}$ Austausch von Prozessdaten	✓
	$A_{1.5}$ Bereitstellung und Sammlung von Daten über die Prozessausführung	✓
A_2	Autonomie beteiligter Parteien	
	$A_{2.1}$ Dezentralität	✓
	$A_{2.2}$ Beschränkung des Zugriffs auf Ressourcen, Funktionalitäten und Informationen	✓
A_3	Verteilungsstrategie und -durchführung	
	$A_{3.1}$ Vorliegen und Spezifikation von Zielen und Rahmenbedingungen für die (verteilte) Prozessausführung	✓
	$A_{3.2}$ Vorliegen und Spezifikation von Zusammenhängen zwischen Prozesspartitionen und Ausführungseinheiten	✓
	$A_{3.3}$ Existenz und Bekanntheit bzw. Möglichkeit zum Auffinden von Ausführungseinheiten	✓
A_4	Korrektheit der Ausführung	
	$A_{4.1}$ Ausführung unter Einhaltung der spezifizierten Ausführungsreihenfolge und -bedingungen	✓(unter Einschränkung der Migration)
	$A_{4.2}$ Kontrollierter Zugriff auf gemeinsame Daten	✓(für Prozessvariablen)
	$A_{4.3}$ Berücksichtigung organisatorischer oder technischer Vorgaben	✓(unter Einschränkung der Migration)
A_5	Fehlererkennung und -behandlung	
	$A_{5.1}$ Lokale Fehlererkennung und -behandlung	✓
	$A_{5.2}$ Globale Fehlererkennung und -behandlung	✓(unter Einschränkung der Migration)
A_6	Überwachung und Nachvollziehbarkeit der Prozessausführung	
	$A_{6.1}$ Bereitstellung und Sammlung von Daten über die Prozessausführung	✓
	$A_{6.2}$ Zuordnung von Verantwortlichkeiten	✓
A_7	Persistenz von prozessrelevanten Daten	
	$A_{7.1}$ Prozessmodelle	✓(lokal)
	$A_{7.2}$ Prozessinstanzdaten	✓(lokal)
	$A_{7.3}$ Prozesshistorien	✓(lokal)
A_8	Vermeidung hohen Koordinationsaufwands	
	$A_{8.1}$ Minimierung des Nachrichtenaustausches	nur bzgl. der Anzahl von Kommunikationsvorgängen, jedoch tlw. größerer Umfang der Nachrichten
	$A_{8.2}$ Minimierung von Verzögerungen der (verteilten) Prozessausführung	✓
A_9	Sicherheit	
	$A_{9.1}$ Autorisierung und Authentisierung von Ausführungseinheiten	✓
	$A_{9.2}$ Verschlüsselung der Kommunikation zwischen Ausführungseinheiten	✓
	$A_{9.3}$ Vertraulichkeit und Integrität der Prozessbeschreibung	✓(unter Einschränkung der Migration)
	$A_{9.4}$ Delegation der Ausführung von Prozesspartitionen	möglich, vgl. KOTTKE (Kot10)
	$A_{9.5}$ Verantwortlichkeit für die Ausführung von Prozesspartitionen	✓

Tabelle 8.8: Bewertung der Lösung hinsichtlich der allgemeine Anforderungen verteilt ausgeführter Prozesse (vgl. Tabelle 4.2)

Eine wesentliche Gruppe von Anforderungen betrifft die Korrektheit der verteilten Ausführung (Anforderungsgruppe $\mathbf{A_4}$). Im Rahmen der Anwendung auf verschiedene Prozessbeschreibungssprachen wurde dabei identifiziert, dass die Migration von Prozessinstanzen bei Vorliegen bestimmter Kontrollflussstrukturen eingeschränkt werden muss, um die Ausführbarkeit des Prozesses nicht zu gefährden (vgl. Abschnitt 8.3). Dabei wurde gezeigt, dass diese Einschränkungen im Wesentlichen jedoch auch für eine physische Partitionie-

rung von Prozessen gegeben sind. Da die verteilte Ausführung von Prozessen durch Migration von Prozessinstanzen bei einer sequentiellen Ausführung des Prozesses ansonsten einer zentralen (nicht-verteilten) Ausführung entspricht, ist die jeweils lokale Ausführungseinheit für die Korrektheit der Ausführung im jeweiligen Prozessabschnitt verantwortlich (vgl. Abschnitt 6.2.4). Für den kontrollierten Zugriff auf gemeinsam genutzte Daten bei verteilt ausgeführten parallelen Prozessabschnitten wurde ein Konzept zur Erkennung von Datenabhängigkeiten, zur (optimistischen) Konfliktauflösung und zur Synchronisation von Prozessdaten vorgeschlagen (vgl. Abschnitt 6.2.5).

Die Möglichkeiten zur inhaltlichen Fehlererkennung und -behandlung sind ebenfalls zunächst abhängig von der Ausdrucksmächtigkeit der zugrunde liegenden Prozessbeschreibungssprache (Anforderungsgruppe $\mathbf{A_5}$). Lokale technische Fehler, wie etwa die fehlenden Verfügbarkeit einer Ressource, werden von der jeweils für die Ausführung verantwortlichen Ausführungseinheit wie bei einer zentralen Ausführung erkannt und behandelt. Hierbei stellt die dynamische Verteilung des Prozesses zugleich eine Möglichkeit zur Behebung des Fehlers dar, indem der Prozess durch eine andere Ausführungseinheit fortgesetzt werden kann. Globale Fehler können auftreten, wenn als Transaktion gekennzeichnete Aktivitäten durch verschiedene Ausführungseinheiten ausgeführt und zurückgesetzt werden müssen. Kompensationsaktivitäten können dabei auf Basis der zu spezifizierenden Rahmenbedingungen für die Ausführung wahlweise den Ausführungseinheiten zugewiesen werden, welche auch für die Ausführung der zu kompensierenden Aktivitäten verantwortlich gewesen sind, oder durch eine andere geeignete Ausführungseinheit durchgeführt werden. Da nur bei einer parallelen Ausführung mehr als eine Ausführungseinheit gleichzeitig an der Ausführung einer Prozessinstanz beteiligt sein kann, ist die verteilte Erkennung und Behandlung von Ausnahmesituationen über mehrere Ausführungseinheiten auch nur in diesem Fall relevant. Die sich hierdurch ergebenden Einschränkungen für die Verteilung des Prozesses sind jedoch auch für eine physische Partitionierung von Prozessen gegeben (vgl. Abschnitt 8.3).

Die Anforderungen nach Überwachung und Nachvollziehbarkeit der Prozessausführung (Anforderungsgruppe $\mathbf{A_6}$) werden in dem in dieser Arbeit vorgestellten Lösungsansatz unabhängig von der Art der Verteilung auf Basis des Prozessmanagementsystems als verwaltbare Ressource adressiert. Eine Bereitstellung von Daten über die Prozessausführung ist dabei generell über die angebotenen Schnittstellen möglich. Im Rahmen des Modells der verwaltbaren (Sub-)Ressourcen werden auch die Verantwortlichkeiten für die Ausführung von Aktivitäten berücksichtigt und können als Prozesshistorien auch gesammelt und z. B. im Anschluss an die Prozessausführung ausgetauscht werden. Ein integrierter Ansatz mit ergänzenden Möglichkeiten zur gegenseitigen Kontrolle wurde im Rahmen des verteilten *Process-Management-as-a-Service*-Szenarios vorgestellt (vgl. Abschnitt 8.4.2).

Die Verwaltung von Prozessmodellen, Prozessinstanzdaten und ggf. Prozesshistorien erfolgt lokal über die beteiligten Ausführungseinheiten, z. B.

in einer Datenbank (Anforderungsgruppe $\mathbf{A_7}$). Werden die Daten durch die Ausführungseinheiten nicht persistent gespeichert, so besteht ergänzend die Möglichkeit, die erforderlichen Daten durch den Migrationsmanager oder im Rahmen der verwaltbaren Ressource zu sichern. Der Aspekt der Persistenz von prozessrelevanten Daten wurde in dieser Arbeit jedoch nicht weiter vertieft.

Betrachtet man den durch die Verteilung verursachten Koordinationsaufwand (Anforderungsgruppe $\mathbf{A_8}$), so sind im Wesentlichen zwei Arten von Koordinationsnachrichten zu unterscheiden: Nachrichten zur Übergabe des Kontrollflusses und Nachrichten zur zwischenzeitlichen Synchronisation von Prozessdaten. Die Anzahl der Kommunikationsvorgänge zur Übergabe des Kontrollflusses ist – bei gleicher Verteilungskonfiguration des Prozesses – für die Migration von Prozessinstanzen nicht höher als für eine physische Partitionierung des Prozesses. Bei der Durchführung der Verteilung des Prozesses durch Migration muss jedoch stets der gesamte Prozess, bestehend aus der Prozessbeschreibung, den aktuellen Instanzdaten und ggf. weiteren Anweisungen zur Verteilung und/oder Überwachung, transferiert werden. Dies ist zwar auch für eine physische Partitionierung der Fall – hier muss dass Prozessmodell jedoch nur einmalig (zur Entwicklungszeit) auf die (potentiell) teilnehmenden Ausführungseinheiten verteilt werden, während zur Laufzeit nur noch Nachrichten zur Übergabe von Instanzdaten ausgetauscht werden müssen. Aus diesem Grund ist bei einer physischen Partitionierung der Prozessausführung auch die Verzögerung bei der Übergabe des Kontrollflusses geringer als bei der Migration von Prozessinstanzen, da diese zunächst ein vollständiges Deployment des Prozessmodells erfordert, bevor die Instanzdaten hierauf angewendet werden können. Bei der Synchronisation von Prozessdaten im Fall von parallel verteilt ausgeführten Prozesspfaden ist die Anzahl der Koordinationsnachrichten bei beiden Ansätzen abhängig von der Anzahl der erkannten Datenabhängigkeiten. Da bei einer dynamsichen Verteilung durch Replikation des Prozesses jedoch auch die zur Laufzeit entfallenden Datenabhängigkeiten erkannt werden können, können hier im Einzelfall unnötige Koordinationsnachrichten eingespart werden.

Die Anwendung von Sicherheitsmechanismen (Anforderungsgruppe $\mathbf{A_9}$) stellt für die Migration von Prozessinstanzen größere Herausforderungen dar. Die Autorisierung und Authentisierung von Ausführungsumgebungen sowie die Verschlüsselung der Kommunikation der Ausführungseinheiten untereinander sind weitestgehend unabhängig vom Verteilungsmodell und auch für mobile Umgebungen umsetzbar (vgl. [Win07, Kot10]). Die Vertraulichkeit der Prozessbeschreibung ist jedoch durch den Ansatz der physischen Partitionierung von Prozessmodellen deutlich besser umzusetzen, da diese hierbei durch die Zerteilung des Prozesses und die individuelle Zuweisung der Partitionen bereits inhärent gewährleistet wird. Bei einer Migration von Prozessinstanzen kommt es nicht zu dieser Zerteilung und es muss eine Verschlüsselung der vertraulichen Prozesspartitionen vorgenommen werden. Diese erfordert zum einen weiteren Aufwand für Ver- und Entschlüsselung, zum anderen erschwert sie das Deployment von Prozessmodellen, da hierbei

für die verschlüsselten Prozesspartitionen und -variablen unter Umständen Platzhalter-Elemente eingesetzt werden müssen (vgl. Abschnitt 7.5.2). Eine weiterführende Möglichkeit zur Delegation der Ausführung von (verschlüsselten) Prozesspartitionen wird von KOTTKE [Kot10] diskutiert. Die Verantwortlichkeit für die Ausführung einer bestimmten Prozesspartition kann über digital signierte Quittungen für die Übergabe des Kontrollflusses und für Protokolleinträge erfolgen. Auch hierfür ist ein erhöhter Kommunikationsaufwand hinzunehmen, welcher jedoch auch für die Nachvollziehbarkeit der Ausführung im Rahmen einer physischen Fragmentierung des Prozesses anfallen würde.

Tabelle (vgl. Tabelle 8.8) zeigt die in Abschnitt 4.4.1 beschriebenen Gruppen von Anforderungen mit ihren jeweiligen Teilanforderungen und -voraussetzungen in der Zusammenfassung.

8.6.2 Anforderungen dynamisch verteilt ausgeführter Prozesse

Die Anforderungen dynamisch verteilt ausgeführter Prozesse richten sich zunächst an den Umgang mit Unvorhersehbarkeit (Anforderungsgruppe $\mathbf{D_1}$). Die im Rahmen dieser Arbeit vorgeschlagene Migration von Prozessinstanzen unterstützt die dynamische Verteilung von Prozessen zur Laufzeit, wobei auch aktuell zentral ausgeführte Prozessinstanzen durch die Ergänzung von Migrationsmetadaten spontan verteilt fortgeführt werden können. Die Verteilung kann aber dennoch über die Definition von benutzerdefinierten Rahmenbedingungen oder bereits durch eine als Teil der fachlichen Prozessbeschreibung festgelegten Zuordnung zu Organisations- bzw. Ausführungseinheiten auch (teilweise oder gänzlich) zur Entwicklungszeit für alle von dem Prozessmodell abgeleiteten Prozessinstanzen oder für eine einzelne Prozessinstanz unmittelbar vor dem Beginn der Ausführung festgelegt werden. Es werden somit in Hinblick auf den Lebenszyklus prozessorientierter Anwendungen alle prinzipiell möglichen Zeitpunkte zur Festlegung der Verteilungskonfiguration unterstützt. Die Verteilung selbst erfolgt jedoch immer zur Laufzeit der einzelnen Prozessinstanz.

Ebenso kann durch die Migration von Prozessinstanzen ein maximales Verteilungsspektrum erreicht werden (Anforderungsgruppe $\mathbf{D_2}$). Dabei werden – unter den notwendigen Einschränkungen zum Erhalt der Korrektheit der Ausführung – prinzipiell sowohl die extremen Varianten der vollverteilten Prozessausführung bis auf die Ebene atomarer Aktivitäten als auch eine zentrale Ausführung des Prozesses durch eine einzige Ausführung unterstützt. Sowohl die Granularität als auch die Zielorte der Verteilung können jeweils zur Laufzeit festgelegt und unter Beachtung etwaiger benutzerdefinierter Rahmenbedingungen auch *fortwährend* angepasst werden. Über eine starke oder schwache Bindung von Ressourcen im Prozessmodell und die Definition von benutzerdefinierten Rahmenbedingungen können sowohl die bei der Ausführung von Aktivitäten aufzurufenden Ressourcen als auch die für den Aufruf verantwortlichen Ausführungseinheiten bei Bedarf zu verschiedenen Zeitpunkten zuge-

ordnet werden. Für jede Prozessinstanz kann dabei hinsichtlich der Granularität der Verteilung und der Zuordnung zu Ausführungseinheiten ein individuelles Verhalten erreicht werden (Anforderungsgruppe $\mathbf{D_3}$).

Die zusätzliche Bereitstellung der Prozessmanagementsysteme als verwaltbare Ressourcen erlaubt sowohl eine kontinuierliche Überwachung relevanter Informationen über die entfernte Prozessausführung durch das Abonnement von Ereignissen (*Push*-Modell) als auch durch die gezielte Abfrage einzelner Details im Bedarfsfall (*Pull*-Modell) (Anforderungsgruppe $\mathbf{D_4}$). Voraussetzungen hierfür sind die Möglichkeiten der prozessausführenden Partei, auf die benötigten Funktionen selbst zugreifen zu können und die Bereitschaft, diese dem (autorisierten) Beobachter bereitzustellen. Das gleiche gilt für die Bereitstellung von Steuerungsmechanismen wie etwa zum Abbruch des Prozesses (Anforderungsgruppe $\mathbf{D_6}$).

Die Vorhersage zukünftiger Entwicklungen ist insbesondere bei einer dynamisch verteilten Ausführung von unterschiedlichen prozessorientierten Anwendungen eine große Herausforderung, da in der Regel weder die benötigten Informationen noch die an der Ausführung beteiligten Ausführungseinheiten im Voraus bekannt sind. Mittels der strukturierten Kontextdatenprognose wurde ein allgemeines Modell zur Vorhersage von Dienstverfügbarkeiten als Unterstützung für Migrationsentscheidungen vorgeschlagen (vgl. Abschnitte 6.5 und 7.7). Eine Vorhersage und damit eine proaktive Anpassung der Verteilung ist jedoch nur eingeschränkt möglich, da ausreichend historische Daten zu dem nachgefragten Ausführungskontext durch die teilnehmenden Ausführungseinheiten gesammelt worden sein müssen. Die Vorhersage ist zudem nur durch eine relativ umfangreiche weitere Infrastruktur und die ergänzende Modellierung von Anwendungswissen möglich.

Weitere Anforderungen umfassen die Unterstützung von dezentralen Koordinationsformen (Anforderungsgruppe $\mathbf{D_7}$), die Unterstützung mobiler Infrastrukturen (Anforderungsgruppe $\mathbf{D_8}$) sowie – damit zusammenhängend – die Möglichkeit zur Anpassung an heterogene Infrastrukturen (Anforderungsgruppe $\mathbf{D_9}$). Eine dezentrale Koordination der verteilten Ausführung, insbesondere bei einer Anwendung in mobilen Umgebungen, wurde durch die Anwendung der in dieser Arbeit vorgeschlagenen Lösungsansätze zur Weiterentwicklung der kontextbasierten Kooperation gezeigt (vgl. Abschnitt 8.4.1). Die Erweiterung von Prozessmanagementsystemen für den mobilen Einsatz (vgl. *DEMAC* und *Sliver* in Abschnitt 8.2) zeigt, dass die vorgeschlagene Implementierung durch ihre modulare Architektur gut skalierbar ist und dadurch auch auf eingeschränkte Ressourcen (insbesondere hinsichtlich Rechenleistung und Speicherplatz) anpassbar ist. Das Prinzip der Migration von Prozessinstanzen hat sich dabei für die Offline-Ausführung von Prozesspartitionen als besonders vorteilhaft erwiesen, da hierbei immer alle benötigten Informationen bei der aktuell verantwortlichen (mobilen) Ausführungseinheit vorliegen (vgl. auch [CRW98, ZK07, Kun08]). Durch die Beschreibung abstrakter Benutzungsschnittstellen und die Spezifikation von Management-Regeln, welche jeweils durch das prozessausführende System verarbeitet werden können, wur-

den die Möglichkeiten zur Offline-Ausführung zudem weiter verbessert. Eine negative Beeinträchtigung stationärer System erfolgt durch die dynamische verteilte Ausführung von Prozessen durch mobile Prozessteilnehmer nicht, da jeder Prozessteilnehmer prinzipiell über eine eigene Ausführungseinheit verfügt. Normale Wartebeziehungen zwischen den Teilnehmern einer verteilten Prozessausführung (z. B. an Synchronisationspunkten) sind hierdurch aber – analog zu einer physischen Partitionierung von Prozessen – natürlich nicht auszuschließen.

Eine Anpassung an heterogene Infrastrukturen (Anforderungsgruppe $\mathbf{D_9}$) wird für den Aufruf von Ausführungseinheiten und durch Software-Anwendungen repräsentierte Ressourcen über eine dienstorientierte Architektur vorgenommen, welche auch die Verwendung unterschiedlicher und multipler Protokolle unterstützt (vgl. Abschnitt 7.2). Ein- und Ausgabedaten der funktionalen Dienste (insbesondere für die Migration von Prozessen) müssen hierbei in einem gemeinsamen bzw. standardisierten Format vorliegen. Ein Vorschlag für Schnittstellen und teilweise auch Ein- und Ausgabedaten wurde in Kapitel 7 vorgestellt. Eine Integration von menschlichen Prozessteilnehmern ist über die Beschreibung abstrakter Benutzungsschnittstellen möglich, welche jeweils an den aktuellen und individuellen Kontext des Benutzers und somit auch an unterschiedliche Endgeräte angepasst werden können (vgl. Abschnitt 8.4.4).

Eine Berücksichtigung benutzerdefinierter Vorgaben zur Beeinflussung der Verteilung des Prozesses (Anforderungsgruppe $\mathbf{D_{10}}$) kann durch eine Erzeugung und Anreicherung von Migrationsdaten bereits parallel zur Entwicklungszeit oder nach Abschluss der Modellierung durch den Prozessmodellierer vorgenommen werden. Alternativ oder komplementär können die benutzerdefinierten Vorgaben falls gewünscht durch den Prozessinitiator weiter angepasst oder sogar vollständig ersetzt werden. Benutzerdefinierte Vorgaben, welche sich auf noch nicht ausgeführte Prozesspartitionen beziehen, können prinzipiell auch zur Laufzeit durch Prozessteilnehmer oder Administratoren vorgenommen werden (vgl. Abschnitte 6.2 und 7.5.3). Zusätzlich besteht die Möglichkeit zur Spezifikation von lokalen Richtlinien, welche das gewünschte Verhalten der lokalen Ausführungseinheit hinsichtlich der Ausführung eigener und fremder Prozesse angeben kann. Eine Abstimmung der verschiedenen Vorgaben ist nicht Teil dieser Arbeit.

Anforderungsgruppe $\mathbf{D_{11}}$ umfasst verschiedene nicht-funktionale Aspekte. Zunächst war gefordert, dass alle auf der betrachteten Ausführungseinheit zentral ausgeführten Prozesse nicht durch die Verteilung anderer Prozesse dieser Ausführungseinheit oder durch die bloße Möglichkeit zur deren Verteilung negativ beeinträchtigt werden. Wie bereits in Abschnitt 8.5 detailliert erläutert wurde, ist eine Beeinträchtigung der zentral und verteilt ablaufenden Prozessen von der Art und Anzahl der durchzuführenden Migrationsentscheidungen auf Basis der benutzerdefinierten Rahmenbedingungen abhängig und kann in vielen Fällen vollständig vermieden werden. Der zur Anpassung (d. h. zur Verteilung) notwendige Koordinationsaufwand setzt sich aus dem Nach-

Gruppe	Anforderung	Bewertung / Kommentar
$\mathbf{D_1}$	Umgang mit Unvorhersehbarkeit	
	$D_{1.1}$ Unabhängigkeit von feststehenden Verteilungszeitpunkten	✓
	$D_{1.2}$ Spontane Verteilung von zentral ausgeführten Prozessen	✓
	$D_{1.3}$ Partitionierung zur Laufzeit	✓
$\mathbf{D_2}$	Maximales Verteilungsspektrum	
	$D_{2.1}$ Beliebige Granularität	✓
	$D_{2.2}$ Dynamische Bestimmung der Zielorte der Verteilung	✓
	$D_{2.3}$ Variable Bindungszeitpunkte	✓
$\mathbf{D_3}$	Hohe Individualität	
	$D_{3.1}$ Unterschiedliche Verteilung für einzelne Prozessinstanzen	✓
$\mathbf{D_4}$	Zeitnahe Bereitstellung und Erhebung von Informationen	
	$D_{4.1}$ Kontinuierliche Überwachung relevanter Informationen (*Push-Modell*)	✓
	$D_{4.2}$ Individuelle Abfrage einzelner Details im Bedarfsfall (*Pull-Modell*)	✓
$\mathbf{D_5}$	Proaktives Verhalten	
	$D_{5.1}$ Integration von Vorhersagen zukünftiger Entwicklungen	✓(eingeschränkt)
$\mathbf{D_6}$	Steuerung der Prozessausführung	
	$D_{6.1}$ Bereitstellung von Management-Funktionalitäten	✓
$\mathbf{D_7}$	Dezentralität	
	$D_{7.1}$ (Potentielle) dezentrale Koordination der verteilten Ausführung	✓
$\mathbf{D_8}$	Unterstützung mobiler Infrastrukturen	
	$D_{8.1}$ Eingeschränkte Ressourcen	✓
	$D_{8.2}$ Offline-Bearbeitung von Prozesspartitionen	✓
	$D_{8.3}$ Keine Beeinträchtigung stationärer Komponenten	✓
$\mathbf{D_9}$	Anpassung an heterogene Infrastrukturen	
	$D_{9.1}$ Aufruf von Ressourcen	✓
	$D_{9.2}$ Ein- und Ausgabedaten	✓
	$D_{9.3}$ Benutzungsschnittstellen	✓
$\mathbf{D_{10}}$	Berücksichtigung benutzerdefinierter Vorgaben	
	$D_{10.1}$ Vorgaben des Prozessmodellierers	✓
	$D_{10.2}$ Vorgaben des Prozessinitiators bzw. Dienstnutzers	✓
	$D_{10.3}$ Vorgaben von Prozessteilnehmern	✓
$\mathbf{D_{11}}$	Nicht-funktionale Aspekte	
	$D_{11.1}$ Keine Beeinflussung zentral ausgeführter Prozesse	✓(abhängig von benutzerdefinierten Vorgaben)
	$D_{11.2}$ Minimierung des zur Anpassung durchzuführende Koordinationsaufwands	naturgemäß höher als bei statischer Verteilung
	$D_{11.3}$ Keine negativen Auswirkungen auf Skalierbarkeit von Prozessen und Systemen	Abhängigkeit von der Anzahl an Aktivitäten und Prozessvariablen
$\mathbf{D_{12}}$	Trennung fachlicher und technischer Logik	
	$D_{12.1}$ Verteilungsinformationen	✓
	$D_{12.2}$ Maßnahmen zur Überwachung	✓
	$D_{12.3}$ Technische Anpassungen	✓
$\mathbf{D_{13}}$	Fortwährende Anpassbarkeit der Organisationsstruktur	
	$D_{13.1}$ Integration neuer und bisher unbekannter Dienste und Ausführungseinheiten	✓
	$D_{13.2}$ Reduktion bzw. Austreten bisher bekannter Dienste und Ausführungseinheiten	✓
$\mathbf{D_{14}}$	Erhalt der fachlichen Evolution von Prozessmodellen	
	$D_{14.1}$ Erweiterbarkeit von bestehenden Prozessmodellen	✓
$\mathbf{D_{15}}$	Erhalt der fachlichen Anpassbarkeit laufender Prozessinstanzen	
	$D_{15.1}$ Inhaltliche Anpassbarkeit auf Basis bestehender Konzepte	✓
$\mathbf{D_{16}}$	Automatisierbarkeit des gesamten Anpassungsvorgangs	
	$D_{16.1}$ Erhebung und Verarbeitung von Informationen	bei Vorhersehbarkeit
	$D_{16.2}$ Ableitung von Anpassungsbedarf	bei Vorhersehbarkeit
	$D_{16.3}$ Durchführung der Anpassung (Verteilung)	✓

Tabelle 8.9: Bewertung der Lösung hinsichtlich der Anforderungen dynamisch verteilt ausgeführter Prozesse (vgl. Tabelle 4.3)

richtenaustausch zum Auffinden geeigneter Ausführungseinheiten mittels eines (zentralen oder verteilten) Verzeichnisdienstes, der eigentlichen Migration des Prozesses und den unter Umständen notwendigen (Re-)Abonnements von Ereignissen für die Fortführung der Prozessausführung bzw. für deren Kündigung zusammen. Der zur Laufzeit durchzuführende Koordinationsaufwand ist dabei naturgemäß höher als bei einer statischen Verteilung des Prozesses (z. B. durch die physische Partitionierung des Prozessmodells), da hierbei die Prozessteilnehmer in der Regel feststehen und daher nicht zur Laufzeit aufgefunden werden müssen. In Hinblick auf die Skalierbarkeit von Prozessen und Ausführungssystemen kann festgestellt werden, dass der Umfang der Prozessbeschreibung sich entsprechend auf die Menge der bei jedem Verteilungsschritt zu übertragenden Daten auswirkt (vgl. Abschnitt 8.5.5). Selbiges gilt für die Instanzdaten des Prozesses (insbesondere für Prozessvariablen), welche bei jeder Migration – unabhängig von ihrer tatsächlichen Verwendung auf der aktuellen Zielausführungseinheit – inkludiert werden. Enthalten die Prozessvariablen große Datenmengen, so kann eine Verteilung des Prozesses durch Migration von Prozessinstanzen über mehrere Ausführungseinheiten zu einer relativ großen Belastung des Kommunikationsnetzwerks führen (vgl. Abschnitt 8.5). Durch die Verwendung von Verfahren zur Datenkompression lässt sich dieser Umfang jedoch (insbesondere bei XML-basierten Prozessbeschreibungen) teilweise deutlich reduzieren.

Ein großer Vorteil des hier vorgestellten Lösungsansatzes besteht in der Trennung fachlicher und technischer Anweisungen (Anforderungsgruppe $\mathbf{D_{12}}$). Dies wird einerseits durch die lose Kopplung von Migrationsdaten und Prozessbeschreibung erreicht, und andererseits durch die Bereitstellung von Überwachungs- und Steuerungsmaßnahmen ergänzt, welche während der entfernten Prozessausführung durch den Prozessinitiator zugegriffen werden können. Das Ergebnis ist, dass zu keinen Zeitpunkt innerhalb des Prozesslebenszyklus weder Verteilungsinformationen noch Maßnahmen zur Überwachung oder für die Ermöglichung technischer Anpassungen statisch mit den fachlichen Aktivitäten des Prozesses und ihrem Kontrollfluss integriert werden müssen. Die strikte Trennung erlaubt dabei nicht nur eine fortwährende Anpassbarkeit der Verteilung, Überwachung und Steuerung des Prozesses, sondern wirkt sich auch positiv auf die fortwährende Anpassbarkeit der Organisationsstruktur aus (Anforderungsgruppe $\mathbf{D_{13}}$). Da die Entscheidung über die Verteilung zur Laufzeit durchgeführt wird und keine feste Zuordnung von Prozesspartitionen zu Ausführungseinheiten notwendig ist, ist eine Integration neuer und bisher unbekannter Dienste und Ausführungseinheiten ebenso möglich wie eine Reduktion bzw. ein Austreten bisher bekannter Dienste und Ausführungseinheiten. Sofern der Abstraktionsgrad der Prozessbeschreibung und die benutzerdefinierten Rahmenbedingungen für die Verteilung des Prozesses dieses zulassen, ist während der gesamten Laufzeit ein dynamisches Rebinding von Diensten und Ausführungseinheiten möglich.

Sowohl die Trennung zwischen fachlichen und technischen Anweisungen als auch das Konzept der Prozessmigration wirken sich zudem positiv auf eine fachliche Evolution von Prozessmodellen aus (Anforderungsgruppe $\mathbf{D_{14}}$). Da das fachliche Prozessmodell in seiner ausführbaren Repräsentation nicht mit Verteilungs-, Überwachungs- und Steuerungsmaßnahmen angereichert werden muss, wird die Übersetzung von der fachlichen Modellierung in die technische Repräsentation sowie umgekehrt deren Weiterentwicklung auf fachlicher Ebene (im Rahmen des sogenannten *Roundtrips* [FG08]) bei verteilt auszuführenden Prozessen nicht unnötig erschwert. Zudem werden bei der Prozessmigration die Prozessmodelle zur Ausführung jeder Prozessinstanz neu transferiert. Eine fachliche Evolution im Prozessmodell setzt sich also sofort durch, sobald eine neue Prozessinstanz gestartet wird, während die sich bereits in der Ausführung befindlichen Prozessinstanzen jeweils nach dem alten Prozessmodell weiter ausgeführt werden. Es besteht daher nicht wie bei einer physischen Partitionierung von Prozessen die Notwendigkeit, bereits vorverteilte Prozessmodelle bzw. deren Fragmente aufzufinden und einzeln auszutauschen bzw. bei noch laufenden Prozessinstanzen eine Versionierung der Prozessmodelle vorzunehmen und bei der Ausführung zu berücksichtigen. Da bei einer solchen *verzögerten Ausbreitung* (vgl. Abschnitt 3.3.1) nur das Prozessmodell auf dem Startsystem angepasst werden muss, ist der Änderungsaufwand äquivalent zu einer Evolution des Prozessmodells bei einer nicht-verteilten Ausführung von Prozessen.

Auf die gleiche Weise wird eine fachliche Anpassbarkeit laufender Prozessinstanzen unterstützt (Anforderungsgruppe $\mathbf{D_{15}}$). Da jede Prozessinstanz ihre eigene Prozessbeschreibung mit sich führt, kann diese prinzipiell jederzeit in einem konsistenten Zustand angehalten werden und durch die Modifikation „ihres“ Prozessmodells individuell in ihrem fachlichen Ausführungsverhalten angepasst werden. Der vorgestellte Migrationsdienst zur verteilten Ausführung von Prozessen unterstützt diese Funktionalität bereits und kann in Kombination mit bestehenden Ansätzen zur inhaltlichen Anpassung von Prozessen für das Anhalten und Starten der Prozessinstanzen verwendet werden. Auf Basis der bereitgestellten Steuerungsfunktionalitäten der verwaltbaren Ressource ist ein solcher Eingriff sogar bei einer entfernten Prozessausführung denkbar. Die Funktionen zur Überwachung und Steuerung unterstützten dabei eine Identifikation und Anpassung der Prozesse. Die bei einer solchen dynamischen Anpassung zu berücksichtigenden Rahmenbedingungen für die Erhaltung der (fachlichen) Konsistenz des Prozesses bleiben durch die hier bereitgestellten (rein technischen) Möglichen unberührt. Zudem ist es notwendig, die Prozessinstanz für die Anpassung zunächst einmal aufzufinden. Die Identifikation der für die aktuelle Ausführung der Prozessinstanz verantwortlichen Ausführungseinheiten ist jedoch ebenso für andere Verteilungsmodelle (wie z. B. für die physische Partitionierung von Prozessen) notwendig.

Schließlich stellt die Möglichkeit zu einer vollständigen Automatisierbarkeit des gesamten Anpassungsvorgangs eine wichtige Anforderung dar, um Verzögerungen der Ausführung durch die Notwendigkeit manueller Handlun-

gen vermeiden zu können (Anforderungsgruppe $\mathbf{D_{16}}$). Die Verteilung des Prozesses kann auf Basis benutzerdefinierter Rahmenbedingungen oder durch die Spezifikation lokaler Richtlinien durchaus automatisiert durchgeführt werden. Für eine automatisierte Überwachung und Steuerung des Prozesses wurde zudem ein Managementdienst vorgestellt, welcher auf Basis von Regeln und der Verarbeitung komplexer Ereignisse automatisiert Aktionen ausführt (vgl. Abschnitte 6.3.5 und 7.4). Voraussetzung für eine zielgerichtete automatische Verteilung, Überwachung und Steuerung ist jedoch in jedem Fall, dass die für eine Verteilung wesentlichen Umstände und Rahmenbedingungen zumindest abstrakt im Voraus bekannt sind, damit die jeweils relevanten Informationen erhoben und verarbeitet werden können. Ist dies nicht der Fall oder können die benötigten Daten nicht (automatisiert) erhoben werden, so muss eine manuelle Anpassung erfolgen (vgl. z. B. das Anwendungsszenario zum Umgang mit großen Datenmengen in Abschnitt 8.4.5). Eine Anpassung ist also prinzipiell jederzeit möglich, kann jedoch bei vollständig unvorhersehbaren Ereignissen unter Umständen nicht immer auch automatisiert erfolgen.

Tabelle 8.9 fasst die in Abschnitt 4.4.2 identifizierten Anforderungen einer dynamisch verteilten Ausführung von Prozessen sowie die Bewertung des hier vorgeschlagenen Lösungsansatzes anhand dieser Anforderungen zusammen.

8.7 Zusammenfassung

In diesem Kapitel wurde gezeigt, dass die im Rahmen dieser Arbeit vorgeschlagenen Konzepte und deren prototypische Implementierung auf bestehende Prozessmanagementsysteme für stationäre und mobile Anwendungsbereiche angewendet werden können. Als Ergebnis ist es möglich, Prozessinstanzen bestehender Prozesse zur Laufzeit individuell zu überwachen und bei Feststellung von Anpassungsbedarf dynamisch zu verteilen oder anderweitig steuernd einzugreifen, ohne dass dabei Anpassungen an der fachlichen Prozessbeschreibung notwendig werden. Konkrete Beiträge dieser Arbeit ergeben sich aus

- der exemplarischen Anwendung des vorgeschlagenen abstrakten Migrationsmodells zur dynamischen Verteilung von XPDL- [ZHKK10], WS-BPEL- [ZKML10] und BPMN-Prozessen [BDH+12] und der dabei vorgenommenen Identifikation und Ableitung wesentlicher Einschränkungen hinsichtlich der Untrennbarkeit bestimmter Kontrollflusskonstrukte für eine Migration von Prozessinstanzen,
- der Anwendung der Konzepte zur zur Weiterentwicklung der *kontextbasierten Kooperation* in Form einer parallel verteilten Ausführung von Prozessinstanzen mit Datenabhängigkeiten [ZHKK10], der kontextbasierten Darstellung von Aktivitäten mit Benutzerinteraktion [ZVBK09] und der Prognose von Kontextdaten [ZML11],
- dem Vorschlag eines (kommerziellen) verteilten *Process-Management-as-a-Service*-Szenarios [ZL10] auf Basis einer Integration der hier vorgestellten Lösungsansätze.

Da die Anpassung der Überwachung und Verteilung des Prozesses bei relativ geringen Reaktionszeiten (vgl. Abschnitt 8.5.5) vollständig zur Laufzeit erfolgen kann, ist die Anpassungsfähigkeit von prozessorientierten Anwendungen in Hinblick auf ihre Überwachung und Verteilung vom Bereich einer mittel- bzw. langfristigen Anpassungsfähigkeit auch auf den Bereich der kurzfristigen Anpassungsfähigkeit ausgeweitet worden (vgl. Abschnitt 3.2). Da es durch die Verwendung ereignisbasierter Infrastrukturen möglich ist, Entscheidungen über die Verteilung von individuellen Prozessinstanzen weitestgehend unabhängig von deren Ausführung zu treffen, kann eine negative Beeinflussung der lokal ausgeführten Prozesse bzw. Prozesspartitionen bis zum etwaigen Zeitpunkt ihrer Verteilung weitestgehend vermieden werden. Zudem entsteht nur für die tatsächlich an der verteilten Prozessausführung teilnehmenden Ausführungseinheiten Aufwand für die Übertragung und das temporäre Speichern der Prozessbeschreibung und der Daten der aktuellen Prozessinstanz.

Unter Beachtung der identifizierten Untrennbarkeit bestimmter Kontrollflusskonstrukte ist durch die Migration von Prozessmodell- und Instanzdaten ein größtmöglicher Anpassungsspielraum möglich, welcher bei Bedarf durch die Spezifikation benutzerdefinierter Rahmenbedingungen oder lokaler Richtlinien der einzelnen Ausführungseinheiten eingeschränkt werden kann. Für das Erreichen eines maximalen Verteilungsspektrums ist dabei kein Aufwand für die Modellierung aller möglichen Handlungsalternativen notwendig, sondern es entsteht nur Aufwand für die etwaige Spezifikation von Einschränkungen dieser generellen Anpassungsfähigkeit. Der Preis für diese *A-priori-Flexibilität* besteht in einem über alle teilnehmenden Ausführungseinheiten verteilten Übertragungsaufwand, welcher abhängig von der Anzahl an verteilt ausgeführten Prozessinstanzen, der Verteilungsschritte pro Prozessinstanz sowie dem Umfang der Prozessbeschreibung und der Instanzdaten ist. Dieser Aufwand ist prinzipiell auch für andere Verteilungsansätze erforderlich und bei gleichem Anpassungsspielraum teilweise sogar deutlich höher, verhält sich jedoch bei dem hier vorgeschlagenen Ansatz in Hinblick auf die Anzahl der verteilt ausgeführten Prozessinstanzen stabil (vgl. Abschnitt 8.5.7). Während die Reaktionszeit, die Effizienz der Anpassung und der Anpassungsspielraum unabhängig von der Anzahl ausgeführter Prozessinstanzen durch die vorgeschlagenen Flexibilisierungsstrategien deutlich verbessert werden konnten, wird hinsichtlich des entstehenden Übertragungsaufwands gegenüber anderen Verteilungsansätzen insbesondere die Klasse derjenigen prozessorientierten Anwendungen unterstützt, für die einzelne Prozessinstanzen dynamisch verteilt ausgeführt werden müssen und für die im Voraus keine Verteilungskonfiguration bekannt ist. Das Ergebnis der Untersuchung entspricht also den im Rahmen dieser Arbeit an die dynamische Verteilung, Überwachung und Steuerung verteilt ausgeführter Prozesse gestellten Anforderungen.

9 Schlussbetrachtung

Ziel dieser Arbeit war es, durch die Untersuchung dynamisch verteilt ausgeführter Prozesse, die Identifikation relevanter Flexibilisierungsmöglichkeiten und die Entwicklung geeigneter Konzepte und technischer Umsetzungsmöglichkeiten auf eine Flexibilisierung verteilt ausgeführter prozessorientierter Anwendungen hinzuwirken. Als wesentlicher Untersuchungsgegenstand wurde ein solcher dynamisch verteilt ausgeführter Prozess auf Anwendungsebene als eine festgelegte Abfolge von Aktivitäten betrachtet, bei der zur Laufzeit die Verantwortlichkeit für die automatisierte Verwaltung des Prozesses, d. h. für die Bearbeitung des Kontrollflusses und den Aufruf von Ressourcen, an verschiedene Hard- und/oder Softwaresysteme übertragen werden kann. Dabei stand in dieser Arbeit insbesondere im Vordergrund, durch die Entkopplung von fachlichen Aktivitäten und technischen Anweisungen eine dynamische Anpassung der Verteilungskonfiguration, der Überwachung und der Steuerung von individuellen Prozessinstanzen zu ermöglichen und somit jederzeit eine Anpassung der Ausführung verteilter Prozesse an dynamische Kontexte zu erlauben, ohne dabei Änderungen an der ursprünglichen Prozessbeschreibung erforderlich zu machen.

In diesem Kapitel werden die Kernpunkte der vorliegenden Arbeit zusammengefasst, deren Ergebnisse reflektiert und von weiterführenden Fragestellungen abgegrenzt. Zunächst erfolgt dazu eine inhaltliche Zusammenfassung der wesentlichen Thesen, Erkenntnisse und Lösungsvorschläge, welche im Verlauf dieser Arbeit identifiziert und erarbeitet wurden (vgl. Abschnitt 9.1). Im Anschluss daran wird das Ergebnis dieser Arbeit in Form ihres Beitrages zum Stand der Forschung diskutiert (vgl. Abschnitt 9.2). Das Kapitel schließt mit einem Ausblick auf weiterführende Fragestellungen, welche aus dem Kontext dieser Arbeit hervorgehen, jedoch thematisch abzugrenzen und bei Bedarf durch anschließende Forschungsarbeiten zu untersuchen sind (vgl. Abschnitt 9.3).

9.1 Inhaltliche Zusammenfassung

Eine Untersuchung prozessorientierter Anwendungen im Allgemeinen hat gezeigt, dass die Abbildung von Anwendungsvorgängen als prozessorientierte Anwendungen im Rahmen des bestehenden Prozesslebenszyklus sowohl eine Evolution des fachlichen Anwendungsinhalts als auch eine von einzelnen Systemressourcen unabhängige technische Repräsentation zur Ausführung in sich verändernden Systemumgebungen erlaubt. Durch die zunehmende Notwendigkeit, prozessorientierte Anwendungen auch über mehrere Organisationen bzw. Organisationseinheiten verteilt auszuführen, durch eine immer

größere Mobilität von Prozessteilnehmern und Anwendungskomponenten und durch die verstärkten Anforderungen an eine Verringerung von Reaktionszeiten kann abgeleitet werden, dass eine hohe und vor allem kurzfristige Anpassungsfähigkeit auch für verteilt ausgeführte Prozesse wünschenswert ist. Zentrale Inhalte dieser Arbeit bestanden daher in der Untersuchung (dynamisch) verteilt auszuführender prozessorientierter Anwendungen, der Betrachtung ihrer Flexibilität und der Erarbeitung von weiteren Flexibilisierungsmöglichkeiten.

Im Rahmen der Untersuchung des Flexibilitätsbegriffs für soziotechnische Systeme wurde erarbeitet, dass Flexibilität die Eigenschaft eines Systems bezeichnet, ohne unverhältnismäßigen Aufwand fortwährend Anpassungen an veränderte Bedingungen zu erlauben. Für verteilt ausgeführte prozessorientierte Anwendungen manifestiert sich diese Eigenschaft vor allem in der *A-priori-Flexibilität*, zu jedem Zeitpunkt vor und während der Ausführung einer Prozessinstanz die Verteilungskonfiguration, d. h. die Granularität der Verteilung, die Partitionierung des Prozesses und die für die resultierenden Prozesspartitionen verantwortlichen Ausführungseinheiten (neu) festlegen zu können. Für die Feststellung von Anpassungsbedarf sowie für die Durchführung von konkreten Anpassungsmaßnahmen besteht zudem die Notwendigkeit, die für die Anpassung relevanten Informationen zu erheben und auf geeignete Funktionalitäten zur Steuerung der Prozessausführung zugreifen zu können. Eine Identifikation von Anforderungen anhand von typischen Anwendungsszenarien und deren Abgleich mit klassischen Verteilungsmodellen und aktuellen bestehenden Ansätze für die verteilte Ausführung von Prozessen hat dabei gezeigt, dass für dynamische Anpassungen verteilt ausgeführter Prozesse in Bezug auf die folgenden Aspekte noch Flexibilisierungspotential besteht:

- **Dynamische Verteilung:** Es wurde abgeleitet, dass sowohl eine spontane Initiierung einer individuellen Verteilung von einzelnen zentral ausgeführten Prozessinstanzen zu deren Laufzeit als auch eine Rekonfiguration der Verteilungsparameter bei bereits verteilt ausgeführten Prozessinstanzen ohne eine aufwendige (proaktive oder nachträgliche) Anpassung auf Prozessmodellebene bislang nicht angemessen unterstützt wird.

- **Dynamische Überwachung:** Unabhängig vom gewählten Verteilungsmodell wurde festgestellt, dass eine spontane Initiierung von Überwachungsmaßnahmen für einzelne Prozessinstanzen entfernt ausgeführter Prozesspartitionen sowie eine zur Laufzeit für einzelne Prozessinstanzen anpassbare Überwachung entfernt ausgeführter Prozesspartitionen über die Grenzen bestehender Ausführungseinheiten hinaus derzeit nicht ohne eine aufwendige (proaktive oder nachträgliche) Anpassung auf Prozessmodellebene unterstützt wird.

- **Dynamische Steuerung:** Unabhängig vom gewählten Verteilungsmodell wird bei festgestelltem Anpassungsbedarf die Durchführung von Steuerungsmaßnahmen (wie z. B. das temporäre Anhalten oder der Abbruch von Prozessinstanzen) über die Grenzen bestehender Ausführungseinheiten hinaus derzeit in der Regel nicht unterstützt, so dass auch viele der bisher erforderlichen nachträglichen Anpassungsmaßnahmen für eine dynamische Verteilung und Überwachung nicht möglich sind bzw. erschwert und verzögert werden.
- **Dynamische Benutzungsschnittstellen:** Während eine dynamische Bindung von Ressourcen auf Grundlage dienstorientierter Architekturen oder rollenbasierter Organisationsmodelle auch bei einer unvorhersehbaren verteilten Ausführung von Prozessen ermöglicht werden kann, können die für interaktiv auszuführende Aktivitäten eines Prozesses definierten Benutzungsschnittstellen aufgrund eines zu geringen Abstraktionsgrads in der Regel nicht angemessen an den erst zur Prozesslaufzeit festzustellenden Kontext des Benutzers und seine aktuelle Situation angepasst werden.
- **Proaktive Anpassungen:** Eine proaktive Anpassung der Prozessausführung ist in der Regel möglich, wenn die voraussichtlichen Eigenschaften zukünftiger Ausführungskontexte abgeleitet werden können. Eine Vorhersage von Kontextdaten für prozessorientierte Anwendungen ist jedoch durch die Vielfältigkeit potentieller Anwendungskontexte und den oft erst zur Laufzeit des Prozesses abzuleitenden Inhalt einer (Kontextdaten-)Prognose begrenzt.

Bei einer Betrachtung von bestehenden Möglichkeiten zur Flexibilisierung von prozessorientierten Anwendungen im Allgemeinen wurden dienstorientierte Architekturen, kontext-, regel-, und ereignisbasierte Systeme sowie Konzepte zur Selbstverwaltung und agentenorientierte Ansätze untersucht. Aus den dabei gesammelten Erkenntnissen wurde abgeleitet, dass die oben genannte A-priori-Flexibilität zur Ermöglichung von Anpassungen zur Laufzeit einzelner Prozessinstanzen insbesondere durch einen hohen Abstraktionsgrad und eine strikte Trennung von fachlichen Aktivitäten des Prozesses und technischen Ressourcen zur Verteilung, Überwachung und Steuerung des Prozesses erreicht werden kann. Basierend auf diesem Leitbild wurden die identifizierten Flexibilisierungsmöglichkeiten im einzelnen im Detail betrachtet und auf mögliche Lösungsansätze untersucht, welche im Folgenden jeweils zu den genannten Teilbereichen zusammengefasst werden.

Dynamische Verteilung

Für eine dynamische Verteilung wurde auf Basis einer Untersuchung bestehender Verteilungsmodelle das Konzept der *Migration von Prozessinstanzen* gewählt, um darauf aufbauend ein allgemeines Modell für die dynamische Verteilung von Prozessinstanzen durch eine logische Partitionierung des Prozes-

ses zu erarbeiten. Dazu wurde ein abstraktes Migrationsmodell vorgestellt, welches sich prinzipiell auf alle zu einem minimalen Prozessmetamodell konformen Prozesse anwenden lässt. Auf Grundlage des Migrationsmodells lassen sich Kontrollflusselemente des Prozessmodells, wie insbesondere Aktivitäten und Variablen, aus einer das Prozessmodell ergänzenden Beschreibung der Prozessinstanzdaten referenzieren und mit zusätzlichen Anweisungen zur Verteilung des Prozesses versehen. Durch eine gemeinsame Weitergabe des unveränderten Prozessmodells und den als Migrationsdaten zusammengefassten Beschreibungen der Prozessinstanz und der Verteilungsanweisungen kann der Zustand der Prozessinstanz prinzipiell auf jeder kompatiblen Ausführungseinheit zu jeder Zeit wiederhergestellt und unter den im Prozessmodell angegebenen Kontrollflussanweisungen unter Beachtung etwaiger benutzerdefinierter Rahmenbedingungen für die (weitere) Verteilung aus- bzw. fortgeführt werden.

Auf der Grundlage einer Auswahl der für prozessorientierte Anwendungen im Allgemeinen von VAN DER AALST identifizierten Kontrollflussstrukturen sowie deren Erweiterung durch aktuelle Forschungsarbeiten wurde der Lösungsansatz für sequentiell ausgeführte Prozesse, verteilt parallel ausgeführte Kontrollflusspfade und ereignisbasierte Kontrollflussstrukturen weiter angepasst und notwendige Einschränkungen für die dynamische Verteilung von Prozessen erarbeitet. Das Ergebnis wurde auf die drei standardisierten Prozessbeschreibungssprachen XPDL, WS-BPEL und BPMN angewendet und in Bezug auf konkrete Kontrollflussstrukturen überprüft.

Für die Verarbeitung der Migrationsdaten und der Durchführung von Verteilungsentscheidungen wurde eine komponentenbasierte Middleware vorgeschlagen, welche bestehende dienstorientierte Ansätze zum verteilten Prozessmanagement um einen Ansatz nach dem *Process-Management-as-a-Service*-Prinzip erweitert. Hierbei stellt jede potentiell an der verteilten Prozessausführung teilnehmende Ausführungseinheit einen Dienst bereit, welcher als Eingabeparameter ein ausführbares Prozessmodell in der von der Ausführungseinheit unterstützten Prozessbeschreibungssprache sowie die als Migrationsdaten spezifizierte Prozessinstanz und die ggf. dazugehörigen Verteilungsanweisungen entgegennimmt. Für ein dynamisches Auffinden, eine Auswahl von Ausführungseinheiten und eine etwaige Verhandlung von Dienstgütevereinbarungen können daher bestehende Konzepte zur Suche, Auswahl und Einbindung von elektronischen Diensten eingesetzt werden. Als Ergebnis ist eine weitestgehend nahtlose Integration der hier vorgestellten Lösung in bestehende dienstorientierte Architekturen möglich, welche für drei verschiedene Prozessmanagementsysteme prototypisch implementiert und an verschiedenen Anwendungsszenarien erprobt wurde [ZKML10, BDH+12, ZHKK10, ZL10].

Dynamische Überwachung und Steuerung

Für eine dynamische Überwachung und Steuerung von entfernt ausgeführten Prozessinstanzen wurde das Konzept der *verwaltbaren Ressource* aus dem Forschungsbereich des *Autonomic Computing* gewählt, um Funktionalitäten von Prozessmanagementsystemen im Allgemeinen über eine dienstbasierte Schnittstelle nach außen bereitstellen zu können. Auf Basis einer Literaturrecherche und einer Untersuchung grundlegender Eigenschaften von verschiedenen Prozessmanagementsystemen wurde dazu ein abstraktes Modell eines *Prozessmanagementsystems als verwaltbare Ressource* erarbeitet, welches sich hierarchisch in Subressourcen zu auf diesem System installierten Prozessmodellen, laufenden Prozessinstanzen und abgeschlossenen Prozessen in Form von Prozesshistorien gliedert. Jede (Sub-)Ressource besitzt dabei eine Auswahl an grundlegenden Eigenschaften, welche insbesondere für die zuvor genannte dynamische Verteilung von Prozessen relevant sind und durch weitere Eigenschaften für andere Anwendungszwecke ergänzt werden können. Die erforderlichen Informationen bzw. Steuerungsmöglichkeiten des Prozessmanagementsystems werden dabei durch eine systemspezifische Sensor-/Effektor-Schnittstelle zugegriffen und in Form von elektronischen Diensten über eine nach *Information*, *Ereignissen* und *Manipulation* differenzierende Schnittstelle verfügbar gemacht.

Eine Implementierung des Prozessmanagementsystems als verwaltbaren Ressource wurde durch die Anwendung des Rahmenwerks des *Web Services Distributed Management* (*WSDM*) vorgenommen und mit verschiedenem Umfang an mehreren bestehenden Prozessmanagementsystemen getestet, wobei für Prozessmanagementsysteme für mobile Geräte eine leichtgewichtige Variante erstellt wurde. Die lokale Sensor-/Effektor-Schnittstelle der verwaltbaren Ressource stellt dabei gleichzeitig die Grundlage für die Durchführung von Migrationsentscheidungen und die Installation bzw. Deinstallation von Prozessinstanzen für eine dynamische Verteilung dar. Die öffentliche Schnittstelle mit Diensten zur Überwachung und Steuerung von Prozessen kann darüber hinaus individuell komponiert und so für den Aufbau von komplexeren Managementanwendungen genutzt werden. Als Beispiel hierfür wurde ein Managementdienst vorgestellt, welcher auf Basis von *EPL-basierten Regeln* und *Complex Event Processing* automatische Reaktionen auf das Eintreten benutzerdefinierter Situationen in Hinblick auf die Ausführung entfernt ausgeführter Prozesspartitionen ermöglicht [Zap07, ZBH+10].

Dynamische Benutzungsschnittstellen

Werden Interaktionen mit menschlichen Prozessteilnehmern aus Sicht der im Prozess spezifizierten Aktivitäten nicht transparent durch den Aufruf einer Software-Anwendung mit Benutzungsschnittstelle durchgeführt, so müssen mit der Prozessbeschreibung zusätzliche Beschreibungen zur Darstellung einer Benutzungsschnittstelle mitgeführt werden, um die Ausgabe von Anweisungen bzw. die Eingabe von Daten durch den Benutzer explizit zu

ermöglichen. Da bei einer dynamischen Verteilung des Prozesses zur Entwicklungszeit dieser Benutzungsschnittstelle nicht bekannt ist, auf welchem Endgerät und in welcher Situation die Interaktion mit dem Benutzer stattfinden wird, wurde auf Basis des *ConcurTaskTree*(*CTT*)-Modells eine abstrakte Repräsentation für Benutzungsschnittstellen abgeleitet, welche zur Laufzeit in Abhängigkeit des aktuellen Kontextes in eine jeweils angemessene konkrete Darstellung überführt wird.

Für eine kontextbasierte Darstellung zur Laufzeit des Prozesses wurde ein konfigurierbarer generischer Interaktionsdienst konzipiert, welcher durch den Prozess analog zu Software-Anwendungen zur Ausführung von fachlichen Aktivitäten aufgerufen werden kann. Aufgrund der hohen Relevanz von Benutzerinteraktionen bietet das Konzept abstrakter Benutzungsschnittstellen insbesondere im Kontext heterogener mobiler Ausführungseinheiten Vorteile. Es wurde daher für verschiedene mobile Endgeräte (insb. Mobiltelefone) getestet und wird derzeit in zwei verschiedenen Anwendungsgebieten eingesetzt [ZBV09, ZVBK09].

Proaktive Anpassung

Liegt zum Zeitpunkt einer dynamischen Anpassung der verteilten Prozessausführung keine direkt für die Fortführung des Prozesses geeignete Ausführungseinheit vor, so kann die Verteilungsentscheidung bei sehr dynamischen Ausführungsumgebungen durch die Vorhersage zukünftiger Ausführungskontexte weiter verbessert werden. In Ergänzung zu einer dynamischen Verteilung des Prozesses durch die Migration von Prozessinstanzen wurde daher mit der *strukturierten Kontextdatenprognose* ein Rahmenwerk entwickelt, um Anwendungswissen in Form von Prognosemodellen zu beschreiben und in einem System aus Ausführungseinheiten zur Erhebung von Trainingsdaten und zur Vorhersage von Ausführungskontexten bei individuell anpassbarer Effizienz und Genauigkeit des Prognoseverfahrens nutzbar zu machen. Als konkretes Beispiel wurde ein Prognosemodell für die Vorhersage von Dienstverfügbarkeiten in lokalen Ad-hoc-Netzen für insbesondere mobile Ausführungseinheiten vorgeschlagen, prototypisch implementiert und in einem entsprechenden Anwendungsszenario eingesetzt [MZL10, ZML11].

Neben der Anwendung der vorgestellten Lösungsansätze auf eine Auswahl von Anwendungsgebieten und Fallbeispielen wurden auf Basis eines Modells von GOLDEN und POWELL für die Bewertung der Flexibilität in soziotechnischen Systemen [GP00] geeignete Kriterien für die Einschätzung des Aufwands, der Effektivität und des Nutzens der durch die Anwendung der hier vorgeschlagenen Lösungsansätze neu erreichbaren Anpassungsfähigkeit abgeleitet. Die Ergebnisse der resultierenden Flexibilitätsbetrachtung wurden zudem mit einem Ansatz zur statischen Verteilung von Prozessen ins

Verhältnis gesetzt. Die Flexibilitätsbetrachtung zeigt, dass die hier vorgestellten Lösungsansätze insbesondere bei Unvorhersehbarkeit der Verteilungskonfiguration in Ausführungsumgebungen mit häufig wechselnden Ausführungseinheiten besonders vorteilhaft sind. Die genauen Vor- und Nachteile sowie die sich daraus ergebene Einordnung dieses Lösungsansatzes in den Stand der Forschung werden als Ergebnis dieser Arbeit im folgenden Abschnitt zusammenfassend reflektiert.

9.2 Zusammenfassung und Diskussion der Ergebnisse

Als Pendant zu der bereits relativ umfassend erforschten Flexibilisierung und Anpassung der fachlichen Perspektive eines Prozesses trägt diese Arbeit zu einer Flexibilisierung und Anpassung der technischen Perspektive mit besonderem Fokus auf die verteilte Ausführung von prozessorientierten Anwendungen bei. Die hier vorgenommene Untersuchung dynamisch verteilt ausgeführter Prozesse hat dabei gezeigt, dass durch die Entkopplung von fachlichen Aktivitäten und technischen Anweisungen innerhalb des gesamten Prozesslebenszyklus nicht nur weiterhin eine hohe Flexibilität für fachliche Anpassungen resultiert, sondern auch in technischer Hinsicht die Verteilung, Überwachung und Steuerung von einzelnen Prozessinstanzen zu deren Laufzeit individuell angepasst werden können. Durch eine nicht-invasive Vorgehensweise und eine alternative Art des Einsatzes von dienstorientierten Architekturen zur Ausführung, Überwachung, Steuerung und Verteilung von Prozessen bleibt dabei die Verteilungstransparenz als inhärente Eigenschaft verteilter Systeme aus der Sicht der gesamten prozessorientierten Anwendung während aller Anpassungsmaßnahmen durchgängig erhalten.

Im Gegensatz zu den meisten anderen Arten des Umgangs mit verteilt ausgeführten Prozessen wurde durch diese Herangehensweise eine *A-priori-Flexibilität* geschaffen, durch welche bei der Ausführung jeder Prozessinstanz in Bezug auf technische Anpassungen prinzipiell ein unbegrenzter Anpassungsspielraum offen steht und dieser bei Bedarf von den an der Prozessausführung beteiligten Parteien sinnvoll eingeschränkt werden kann. Es konnte gezeigt werden, dass hiermit der Aufwand zur Anpassung der Prozesse in Bezug auf ihre Verteilung sehr gering ist und dass diese (dominiert von der Dauer für das Anhalten der Prozessausführung und das Durchführen der Verteilungsentscheidung) mit relativ geringen Reaktionszeiten durchgeführt werden kann. Dabei kann die Prozessausführung prinzipiell zu jedem beliebigen Zeitpunkt unter Beachtung weniger nicht trennbarer Kontrollflussstrukturen an beliebige kompatible Ausführungseinheiten verteilt werden. In Hinblick auf eine dynamische Verteilung von Prozessen sind insbesondere die folgenden Beiträge der beschriebenen Lösungsansätze zusammenzufassen:

- Einzelne Prozessinstanzen können dynamisch verteilt ausgeführt werden, ohne dass andere auf derselben Ausführungseinheit ablaufende Pro-

zessinstanzen desselben Prozessmodells oder Prozessinstanzen anderer Prozessmodelle hierdurch negativ beeinträchtigt werden.

- Eine Anpassung der Verteilungskonfiguration ist (für noch nicht ausgeführte Kontrollflusselemente) fortwährend während der gesamten Laufzeit des Prozesses möglich. Für die einzelne Anpassung der Verteilung ist es dabei unerheblich, wie viele Verteilungsschritte bereits zuvor durchgeführt wurden oder zu späterem Zeitpunkt noch durchgeführt werden.
- Der Prozess muss nicht auf der Ebene des Prozessmodells angepasst werden, um eine individuelle Verteilung einzelner Prozessinstanzen zu ermöglichen.
- Es ist eine dynamische Verteilung bestehender Prozesse auf Basis standardisierter Prozessbeschreibungssprachen (XPDL, WS-BPEL, BPMN) möglich. Auf Basis des vorgestellten abstrakten Migrationsmodells und unter Beachtung der in dieser Arbeit beschriebenen Einschränkungen ist eine Übertragbarkeit auf weitere Prozessbeschreibungssprachen anzunehmen.
- Die Benutzungsschnittstellen für Aktivitäten, welche die Interaktion mit einem Benutzer erfordern, können nach einer Verteilung an den aktuellen Kontext des Benutzers angepasst werden, ohne dass die Zielorte der Verteilung zur Entwicklungszeit der Benutzungsschnittstelle bekannt sein müssen.
- Es ist eingeschränkt möglich, die Verteilung des Prozesses proaktiv an erwartete Veränderungen des Ausführungskontextes anzupassen.

Der Aufwand für die Ermöglichung dieser zur Laufzeit vornehmbaren Anpassungen wurde durch die im Rahmen dieser Arbeit vorgeschlagenen Konzepte von der Ergänzung bzw. Transformation von Prozessmodellen in eine unterstützende Middleware als Ergänzung bestehender Prozessmanagementsysteme verschoben. Mit Ausnahme der Anwendung von Sicherheitsmechanismen zur Verschlüsselung von einzelnen Prozesspartitionen bleibt bei der dynamischen Verteilung des Prozesses sowie bei der Durchführung von Überwachungs- und Steuerungsmaßnahmen stets die *fortwährende Anpassbarkeit* der (technischen) Prozessausführung bestehen. Sowohl für die gekapselten Prozessmanagementsystemen als auch für die prozessorientierten Anwendungen und für die an deren Ausführung beteiligten Ressourcen erfolgt die verteilte Ausführung von Prozessen hierbei vollständig transparent. Durch die zusätzliche Möglichkeit zur Definition von Rahmenbedingungen kann die dynamische Verteilung von Prozessen durch den Initiator der verteilten Ausführung und/oder durch die lokalen Ausführungseinheiten manuell oder automatisch sinnvoll eingeschränkt werden. Die dynamische Verteilung des Prozesses in einem kommerziellen Szenario, welches von allen im Rahmen dieser Arbeit vorgeschlagenen Strategien Gebrauch macht, führt jedoch

auch zu einem nicht unerheblichen Overhead an Metadaten, welche zusätzlich zum Prozessmodell bei jedem Verteilungsschritt des Prozesses übertragen und durch die lokale Ausführungseinheit bzw. durch die unterstützende Middleware verwaltet werden müssen. Der Preis für die neu geschaffene Anpassungsfähigkeit zur Laufzeit besteht daher zusammengefasst in

- der einmalig notwendigen Erweiterung potentiell teilnehmender Ausführungseinheiten mit den hier vorgeschlagenen Middleware-Komponenten, wobei gezeigt wurde, dass die Erstellung individueller Adapterkomponenten mit relativ geringem Entwicklungsaufwand möglich ist,
- einem erhöhten Aufwand zur Beherrschung der neu geschaffenen Flexibilität, insbesondere durch die bedarfsgerechte Spezifikation zusätzlicher benutzerdefinierter Anforderungen und Rahmenbedingungen für die entfernte Ausführung eigener Prozesse und/oder lokaler Richtlinien der prozessausführenden Parteien,
- ab einer gewissen Anzahl von (über dieselben Ausführungseinheiten verteilten) Prozessinstanzen desselben Prozessmodells in einem gegenüber statischen Verteilungsansätzen insgesamt erhöhten Übertragungsaufwand für den Transfer von Prozessmodellen und Instanzdaten.

Eine Spezifikation von automatischen Reaktionen und die Anwendung von Sicherheitsmechanismen ist – wie gezeigt wurde – im Rahmen des dargestellten Lösungsansatzes durchaus möglich, führt jedoch in der Regel auch dazu, dass die Flexibilität für die Ausführung von Prozessen (in diesem Fall durchaus beabsichtigt) eingeschränkt wird. Im Gegensatz zu anderen Verteilungsansätzen kann als Nachteil insbesondere die Vertraulichkeit von Prozessen nur durch eine künstliche „Zensur“ nicht öffentlicher Prozesspartitionen durchgesetzt werden, für die weiterer Aufwand in Form von Ver- und Entschlüsselung sowie eine lokale Anpassung der fachlichen Prozessbeschreibung erforderlich sein kann.

Unabhängig vom gewählten Verteilungsmodell lassen sich durch die Abbildung des Prozessmanagementsystems als verwaltbare Ressource in Hinblick auf die Überwachung und Steuerung von entfernt ausgeführten Prozessen im Allgemeinen die folgenden Vorteile zusammenfassen:

- Einzelne Prozessinstanzen können individuell überwacht werden, ohne dass die überwachte Prozessinstanz selbst, andere auf derselben Ausführungseinheit ablaufende Prozessinstanzen desselben Prozessmodells oder Prozessinstanzen anderer Prozessmodelle hierdurch negativ beeinträchtigt werden.

- Die Überwachung und Steuerung von Prozessen kann zur Laufzeit jederzeit (unvorhergesehen) geändert werden, sofern die Initiierung der Anpassung durch das prozessausführende System erfolgt oder der Initiator der Anpassung und das prozessausführende System zur Zeit der Anpassung durch ein Netzwerk verbunden sind.

- Der Prozess muss nicht auf der Ebene des Prozessmodells angepasst werden, um eine individuelle Überwachung oder Steuerung einzelner Prozessinstanzen zu ermöglichen. Solange die definierten Rahmenbedingungen des abstrakten Modells der verwaltbaren Ressource eingehalten werden, ist es nicht wesentlich, in welcher Prozessbeschreibungssprache das betrachtete Prozessmodell verfasst ist.

- Prozessinstanzen können durch bereitgestellte Steuerungsfunktionalitäten auch zur Laufzeit angepasst werden, so dass eingeschränkt auch eine inhaltliche Anpassung von Prozessinstanzen unterstützt werden kann.

Voraussetzungen für die genannten Möglichkeiten zur Überwachung und Steuerung sind die Verfügbarkeit des Zugriffs auf die relevanten Informations- und Steuerungsfunktionalitäten für die prozessausführende Partei selbst und deren Bereitschaft, diese Funktionalitäten für den Konsumenten der entfernt ablaufenden Prozessausführung – ggf. unter weiteren einschränkenden Sicherheitsmaßnahmen – verfügbar zu machen. Die Verfügbarkeit von Informationen und Steuerungsfunktionalitäten bedingt auch die Unterstützung einer automatischen Feststellung von Anpassungsbedarf, die automatische Ableitung von Reaktionen sowie die Sammlung von Daten für eine Prognose von zukünftigen Ausführungskontexten für proaktive Anpassungen. Komplexere Management-Aktionen bzw. -Reaktionen, welche auch eine Automatisierung der Steuerung auf Basis aktuell erhobener Daten zulassen und dabei offline ausgeführt werden, erfordern zudem die Einigung auf weitere Standards zur Beschreibung von Management-Regeln. Ein proaktives Verhalten erfordert darüber hinaus die Einigung auf Möglichkeiten zur Spezifikation von Prognosewissen und zum Austausch von Datensätzen. Mit dem EPL-basierten Management-Dienst und dem Rahmenwerk zur strukturierten Kontextdatenprognose wurde im Rahmen dieser Arbeit jeweils eine mögliche Grundlage hierfür vorgestellt.

Die Kooperation zwischen verschiedenen Organisationen bzw. Organisationseinheiten macht im Fall von verteilt ausgeführten Prozessen zumindest die Einigung auf eine gemeinsame bzw. standardisierte Prozessbeschreibungssprache notwendig. Ein weitergehender Trend zur Standardisierung von Prozesstechnologien und zur Förderung der Interoperabilität von Prozessen und Prozessmanagementsystemen ist am Beispiel der Prozessbeschreibungssprachen bereits deutlich zu erkennen und kann in Zusammenhang mit den hier vorgestellten Lösungsansätzen auch in Bezug auf eine dynamische Verteilung, Überwachung und Steuerung von (zentral und/oder verteilt ausgeführten) Pro-

zessen weiter ausgebaut werden. Im Vergleich mit statischen Ansätzen zur Verteilung der Prozessausführung hat die Untersuchung der Ergebnisse dieser Arbeit zusammenfassend bestätigt, dass

- eine *physische Verteilung* durch eine invasive Partitionierung von Prozessen insbesondere geeignet ist, wenn vorhersehbar ist, dass viele Prozessinstanzen desselben, über längere Zeit unveränderten Prozessmodells auf im voraus bekannten Ausführungseinheiten verteilt und weitestgehend mit derselben Verteilungskonfiguration ausgeführt werden sollen, und/oder wenn für einzelne Prozesspartitionen eine hohe Vertraulichkeit gewährleistet werden muss,
- eine *logische Verteilung* durch die nicht-invasive Migration von Prozessinstanzen insbesondere für dynamische Umgebungen geeignet ist, in denen Ausführungseinheiten zur Laufzeit des Prozesses in die potentielle Ausführungsumgebung eintreten und/oder diese verlassen können, in denen nur einzelne Prozessinstanzen eines Prozessmodells verteilt ausgeführt werden sollen oder in denen nicht vorhersehbar ist, auf welche Weise und auf welchen Ausführungseinheiten Prozessinstanzen verteilt ausgeführt werden sollen.

Als Schlussfolgerung lässt sich festhalten, dass die Ziele dieser Arbeit, in Bezug auf verteilt ausgeführte Prozesse eine hohe Dynamik zu unterstützen, durch die vorgeschlagenen Lösungsansätze erreicht werden konnten. Darüber hinaus kann festgehalten werden, dass insbesondere durch eine Koexistenz von physischer Partitionierung von Prozessmodellen und logischer Verteilung durch Migration von Prozessinstanzen ein Großteil der Anforderungen an verteilt auszuführende Prozesse im Allgemeinen abgedeckt werden kann. Da bei der im Rahmen dieser Arbeit vorgeschlagenen Erweiterung von bestehenden Prozessmanagementsystemen die Möglichkeit zur Nutzung alternativer Verteilungsformen nicht beeinträchtigt, sondern nur ergänzt wird, ist eine alternative oder komplementäre Anwendung beider Verteilungsmodelle prinzipiell jederzeit möglich.

9.3 Ausblick

In dieser Arbeit wurden in erster Linie die Grundlagen für allgemein einsetzbare und anwendungsunabhängige Konzepte zur Flexibilisierung verteilter Prozessausführung beschrieben. Weiterführende Fragestellungen betreffen insbesondere die Unterstützung des Benutzers bei der Spezifikation von Rahmenbedingungen für die Verteilung des Prozesses und eine entsprechende anwendungsspezifische Entwicklung von Modulen zur Durchführung von Migrationsentscheidungen. Zudem können Möglichkeiten zur Vereinigung ereignisbasierter Flexibilisierungsstrategien mit der dynamischen Verteilung von Prozessinstanzen, die automatische Transformation von Benutzungsschnittstellen in das vorgeschlagene abstrakte Format und weiterführende Strate-

gien zur Reduzierung des Übertragungsaufwands für die dynamische Verteilung von Prozessen zu einer Weiterentwicklung der hier vorgeschlagenen Lösungsansätze beitragen.

Unterstützung zur zielgerichteten Einschränkung der Migration

Ein wesentlicher Ansatzpunkt zur Weiterentwicklung des hier vorgestellten Lösungsansatzes besteht in einer umfassenden Unterstützung für eine zielgerichtete Einschränkung der Migration von Prozessinstanzen. Hierbei besteht insbesondere Bedarf für eine automatisierte Erkennung von zusammenhängenden Kontrollflussstrukturen, welche durch eine Migration des Prozesses nicht getrennt voneinander auf verschiedenen Ausführungseinheiten ausgeführt werden dürfen. Wie dargestellt wurde, sind diese Einschränkungen jedoch abhängig von der verwendeten Prozessbeschreibungssprache, so dass eine individuelle Überprüfung und Entwicklung einer solchen Unterstützung gesondert für jede einzelne Beschreibungssprache erfolgen sollte.

Zudem ist in vielen Fällen auch semantisches Wissen über die Zusammenhänge von Kontrollflusselementen notwendig, so dass im Allgemeinen keine vollautomatisierte Einschränkung der Migration erfolgen kann, sondern der Benutzer nur in Form einer Empfehlung bei der Modellierung von Rahmenbedingungen für die Verteilung unterstützt werden kann. Eine Entwicklung von Werkzeugen, welche bei der Erzeugung von initialen Migrationsdaten sogleich auch einen Vorschlag für eine sinnvolle Einschränkung der Migration für ein vorliegendes Prozessmodell inkludieren, stellt für den Anwender jedoch in jedem Fall eine wertvolle Unterstützung dar und kann insbesondere bei einer ungeplanten Verteilung des Prozesses auch zu einer weiteren Reduzierung der Reaktionszeit führen.

Dasselbe gilt für eine Anwendung von Sicherheitsrichtlinien auf der Ebene der Prozessbeschreibung, wo die bisher vom Benutzer manuell zu entwickelnde Konfiguration möglichst automatisiert darauf überprüft werden sollte, ob eine Verschlüsselung von einzelnen Prozesspartitionen und Variablen in Zusammenhang mit den Anweisungen zur Migration vereinbar ist und durch wenigstens eine Kombination von autorisierten Ausführungseinheiten ausgeführt werden kann.

Entwicklung und Integration von Migrationsmodulen und -strategien

Im Rahmen dieser Arbeit wurde eine Beeinflussung der dynamischen Verteilung von Prozessinstanzen durch vier mögliche Akteure erlaubt: Zum einen können Migrationsstrategien für den verteilt auszuführenden Prozess durch den *Modellierer* des Prozessmodells, den *Initiator* der Prozessinstanz und während der Ausführung durch *Prozessteilnehmer* festgelegt und verändert werden. Zum anderen können für jede *Ausführungseinheit* lokale Richtlinien zum Umgang mit eigenen und fremden Prozessen spezifiziert werden, um die verteilte Prozessausführung von Seiten der Ausführungseinheiten zu steuern. Eine alternative oder komplementäre Anwendung von benutzerdefinierten

Rahmenbedingungen und/oder lokalen Richtlinien der Ausführungseinheiten ist anwendungsspezifisch und ist daher durch diese Arbeit nicht explizit festgelegt worden. Für den Fall, dass beide Konzepte zusammen eingesetzt werden sollen, ist jedoch eine Priorisierung oder eine individuelle Aushandlung von Rahmenbedingungen in Form von Dienstgütevereinbarungen denkbar. Benutzerdefinierte Rahmenbedingungen können bei Bedarf zudem durch Sicherheitsmechanismen wie digitale Signaturen vor unerlaubten Änderungen durch nachfolgende Akteure geschützt werden.

Konzeptionell und im Rahmen der hier vorgestellten Implementierung werden sowohl benutzerdefinierte Rahmenbedingungen als auch lokale Richtlinien der Ausführungseinheiten über Module zur Durchführung von Migrationsentscheidungen umgesetzt. Die Entwicklung und die Optimierung dieser Migrationsmodule besitzen für die dynamische Verteilung von Prozessen eine besondere Relevanz, da deren Gestaltung wesentlichen Einfluss auf die Beeinträchtigung der Ausführung von allen Prozessinstanzen hat, für die Migrationsstrategien definiert wurden, welche auf das jeweilige Migrationsmodul zurückgreifen. In dieser Arbeit wurden exemplarisch verschiedene (einfache) Verteilungsstrategien definiert und als Migrationsmodule umgesetzt. Das erweiterbare Konzept erlaubt jedoch auch die Formulierung komplexerer Migrationsentscheidungen und die Integration weiterer allgemeiner oder anwendungsspezifischer Migrationsmodule. Ein auf nicht-funktionalen Aspekten der Prozessausführung beruhendes Konzept wurde bereits im Rahmen einer weiterführenden Arbeit vorgestellt [HSZ11].

Verteilte Verarbeitung von komplexen Ereignissen

Ereignisbasierte Kontrollflussstrukturen und die Verarbeitung von während der Prozessausführung eintretenden (komplexen) Ereignissen stellen wichtige Konzepte zur Flexibilisierung der Prozessausführung in *fachlicher* Hinsicht dar. Während der Verarbeitung von komplexen Ereignissen ist jedoch zur Zeit (unabhängig vom gewählten Verteilungsmodell) keine dynamische Verteilung der Prozessausführung möglich, da in diesem Fall unter Umständen auch die Teilergebnisse (d. h. die Informationen über bereits eingetretene Teilereignisse) zwischen verschiedenen ereignisverarbeitenden Systemen übertragen werden müssten. Bei einer verteilten Prozessausführung können auf komplexen Ereignissen basierende Kontrollflussstrukturen daher zur Zeit in vielen Fällen nicht getrennt werden und schränken somit die Flexibilität der Verteilung (teilweise erheblich) ein. Dies gilt vor allem für Ereignisse, welche für umfangreichere Gültigkeitsbereiche definiert werden und die Prozessausführung in diesem Gültigkeitsbereich durch Routinen zur Ereignisbehandlung unterbrechen bzw. ergänzen, sobald das (komplexe) Ereignis eintritt. Um die Vereinbarkeit beider Flexibilisierungsstrategien zu verbessern sind weiterführende (interdisziplinäre) Arbeiten im Bereich des (verteilten) *Complex Event Processing* notwendig. Insbesondere ist hierfür ein Konzept zu erarbeiten, um eine auf einem ereignisverarbeitenden System begonnene Ereignisverarbeitung während

des potentiellen Eintretens von weiteren (Teil-)Ereignissen verlustfrei auf einem anderen ereignisverarbeitenden System fortsetzen zu können.

Automatische Transformation von Benutzungsschnittstellen

Die im Rahmen dieser Arbeit vorgeschlagene abstrakte Beschreibung von Benutzungsschnittstellen erfordert die Modellierung der Interaktionen in einem speziellen Format, welches zur Laufzeit durch den ausgewählten Interaktionsdienst auf Basis von aktuellen Kontextdaten als konkrete Benutzungsschnittstelle umgesetzt wird. Bestehende Prozessmodelle, deren interaktiv auszuführende Aktivitäten bereits über proprietär definierte Benutzungsschnittstellen verfügen, müssen daher (bei Bedarf) einmalig auf die abstrakte Repräsentation angepasst werden. Da für die konkreten Interaktionselemente im Allgemeinen eine eindeutige Abbildung in das abstrakte Format möglich ist, ist eine weitergehende Unterstützung in Form einer automatisierten Transformation von Benutzungsschnittstellen möglich. Aufgrund der Vielzahl an Beschreibungen für Interaktionen mit dem Benutzer ist jedoch für jede (proprietäre) Repräsentation ein eigenes Werkzeug zur Transformation notwendig.

Reduzierung des Übertragungsaufwands

Da bei jeder Migration des Prozesses für die verteilte Ausführung sowohl das Prozessmodell als auch die Instanzdaten des Prozesses mit ihren unter Umständen umfangreichen Variablenwerten übertragen werden, stellt sich die Frage nach einer möglichen Reduzierung dieses Übertragungsaufwands. Eine einfache Verbesserungsmöglichkeit besteht dabei in der Kompression der zur Zeit weitestgehend text- bzw. xml-basierten Prozessbeschreibungen und Migrationsdaten, wobei das Übertragungsvolumen bereits bei einfachen Kompressionsverfahren bis auf etwa ein Zehntel der Ursprungsgröße reduziert werden kann (vgl. [Wer07]). Anders verhält es sich bei der Integration von umfangreichen Variablenwerten (wie z. B. bei Multimediadateien), welche in der Regel nicht in größerem Umfang reduziert werden können.

Eine bereits bei der Betrachtung von bestehenden Ansätzen zur Migration von Prozessen diskutierte weiterführende Möglichkeit zur Reduzierung des Übertragungsaufwands besteht in der Verringerung der Prozessbeschreibung um die bereits ausgeführten Aktivitäten. Ein solches „Abschneiden“ des Prozessmodells ist jedoch nicht in allen Fällen möglich, da die Möglichkeit der Rücksetzung bzw. Kompensation von Transaktionen, die wiederholte Ausführung von Prozesspartitionen zur Wiederherstellung im Fehlerfall, die Iteration von Schleifen, die Existenz von impliziten Rücksprunganweisungen oder eine Wiederverwendung von Blockaktivitäten und Subprozessen im Kontrollfluss berücksichtigt werden muss. Es ist daher im Einzelfall eine aufwendige Überprüfung notwendig, um zu berechnen, welche Teile der Prozessbeschreibung für die Fortführung des Prozesses nicht mehr benötigt werden. Zudem ist eine schrittweise Verringerung der Prozessbeschreibung nicht ohne weiteres mit einer digitalen Signatur des verteilt auszuführenden Pro-

zesses zur Sicherstellung seiner Integrität vereinbar. Betrachtet man zudem den relativ geringen Umfang der zu übertragenden und ggf. bereits zusätzlich komprimierten Prozessbeschreibungen (vgl. Abschnitt 8.5.5), so ist festzustellen, dass eine Reduzierung einzelner Aktivitäten in Hinblick auf den dadurch verursachten Aufwand nicht effizient ist. Im Gegensatz zur Prozessbeschreibung kann jedoch eine Überprüfung der tatsächlichen Weiterverwendung von Instanzdaten, insbesondere von umfangreichen Variablenwerten wie den genannten Multimediadateien, vorteilhaft sein. Eine Möglichkeit zur Reduzierung des Übertragungsaufwands besteht dabei zum Beispiel in der lokalen Speicherung voraussichtlich nicht mehr benötigter Dateien und dem Einfügen einer Referenz, unter welcher die Datei bei Bedarf heruntergeladen werden kann.

Da der über alle Ausführungseinheiten insgesamt erforderliche Übertragungsaufwand als einer der wesentlichsten Nachteile des hier vorgestellten Ansatzes gegenüber den Ansätzen zur statischen Verteilung von Prozessen identifiziert wurde, kann insbesondere die Konzeption weiterer Strategien zur Reduzierung des Übertragungsaufwands dazu führen, dass das hier vorgestellte Verteilungsmodell gegenüber den Ansätzen zur statischen Verteilung weiter an Vorteilen gewinnt und auch für andere Anwendungszwecke mit geringeren Anforderungen an eine dynamische Verteilung effizient eingesetzt werden kann.

Eigene Veröffentlichungen

Folgende eigene Veröffentlichungen sind aus den Ergebnissen im Umfeld dieses Dissertationsprojektes hervorgegangen:

KRISTOF HAMANN, SONJA ZAPLATA, WINFRIED LAMERSDORF, Process Instance Migration: Flexible Execution of Distributed Business Processes. In *First International Workshop on European Software Services and Systems Research - Results and Challenges (S-Cube 2012)*, Seiten 21–22, IEEE Computer Society, 2012.

ANGINEH BARKHORDARIAN, FREDERIK DEMUTH, KRISTOF HAMANN, MINH HOANG, SONJA WEICHLER, SONJA ZAPLATA, Migratability of BPMN 2.0 Process Instances. In *Service-Oriented Computing - ICSOC 2011 Workshops*, Seiten 66–75, Springer, 2012.

SONJA ZAPLATA, MATTHIAS MEINERS, WINFRIED LAMERSDORF, Designing Future-Context-Aware Dynamic Applications with Structured Context Prediction. In *Software – Practice and Experience (SPE)*, 10.1002/spe.1126, Seiten 1-18, Wiley Online Library, 2011.

KRISTOF HAMANN, SEBASTIAN STEENBUCK, SONJA ZAPLATA, Towards NFC-Aware Process Execution for Dynamic Environments. In HORST HELLBRÜCK, NORBERT LUTTENBERGER, VOLKER TURAU: *Workshops der wissenschaftlichen Konferenz Kommunikation in verteilten Systemen 2011 (WowKiVS 2011)*, Seiten 1-12, Electronic Communications of the European Association of Software Science and Technology (EASST), Jg. 2011, Nr. 37, 2011.

SONJA ZAPLATA, DANIEL STRASSENBURG, BENJAMIN WUNDERLICH, DIRK BADE, KRISTOF HAMANN und WINFRIED LAMERSDORF, Ad-hoc Management Capabilities for Distributed Business Processes. In *3rd International Conference on Business Process and Services Computing (BPSC 2010)*, Seiten 139-152, Gesellschaft für Informatik, 2010.

SONJA ZAPLATA, KRISTOF HAMANN, KRISTIAN KOTTKE und WINFRIED LAMERSDORF, Flexible Execution of Distributed Business Processes based on Process Instance Migration. In *Journal of System Integration*, Jg. 2010, Nr. 3, Seiten 3-16, Czech Society for Systems Integration (CSSI), 2010.

SONJA ZAPLATA, KRISTIAN KOTTKE, MATTHIAS MEINERS und WINFRIED LAMERSDORF, Towards Runtime Migration of WS-BPEL Processes. In ASIT DAN, FREDERIC GITTLER, FAROUK TOUMANI (Hrsg.):*ICSOC/ServiceWave 2009: Fifth International Workshop on Engineering Service-Oriented Applications (WESOA 2009)*, Seiten 477-487, Springer, 2010.

MATTHIAS MEINERS, SONJA ZAPLATA und WINFRIED LAMERSDORF, Structured Context Prediction: A Generic Approach. In FRANK ELIASSEN, RÜDIGER KAPITZA (Hrsg.): *Proceedings of the 10th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS 2010)*, Seiten 84-97, Springer, 2010.

SONJA ZAPLATA und WINFRIED LAMERSDORF, Towards Mobile Process as a Service. In *25th ACM Symposium On Applied Computing (SAC 2010)*, Seiten 372-379, ACM, 2010.

SONJA ZAPLATA, VIKTOR DREILING und WINFRIED LAMERSDORF, Realizing Mobile Web Services for Dynamic Applications. In *AIS Transactions on Enterprise Systems*, Jg. 2009, Nr. 2, Seiten 3-12, GITO-Publishing, 2009.

SONJA ZAPLATA, CHRISTIAN P. KUNZE und WINFRIED LAMERSDORF, Context-based Cooperation in Mobile Business Environments: Managing the Distributed Execution of Mobile Processes. In *Business and Information Systems Engineering (BISE), Jg. 2009, Nr. 4, Seiten 301-314, Gabler, 2009.*

SONJA ZAPLATA, CHRISTIAN P. KUNZE *und* WINFRIED LAMERSDORF, *Kontextbasierte Kooperation für mobile Geschäftsanwendungen: Dezentrale Ausführung und Management von mobilen Prozessen. In* WIRTSCHAFTSINFORMATIK, Jg. 2009, Nr. 4, Seiten 347-362, Gabler, 2009.

SONJA ZAPLATA, VIKTOR DREILING und WINFRIED LAMERSDORF, Realizing Mobile Web Services for Dynamic Applications. In CLAUDE GODART, NORBERT GRONAU, SUSHIL SHARMA, GEROME CANALS (Hrsg.): *Proceedings of the 9th IFIP Conference on e-Business, e-Services, and e-Society (I3E 2009)*, Seiten 240-254, Springer, 2009.

SONJA ZAPLATA, DIRK BADE und ANTE VILENICA, Service-based Interactive Workflows for Mobile Environments. In HANS ROBERT HANSEN, DIMITRIS KARAGIANNIS, HANS-GEORG FILL (Hrsg.): *Business Services: Konzepte, Technologien, Anwendungen - 9. Internationale Tagung Wirtschaftsinformatik (WI 2009)*, Seiten 631-640, Österreichische Computer Gesellschaft, 2009.

SONJA ZAPLATA, ANTE VILENICA, DIRK BADE und CHRISTIAN P. KUNZE, Abstract User Interfaces for Mobile Processes. In KLAUS DAVID, GURT GEIHS (Hrsg.): *16. Fachtagung Kommunikation in Verteilten Systemen (KiVS 2009)*, Seiten 129-140, Springer, 2009.

CHRISTIAN P. KUNZE, SONJA ZAPLATA, MIRWAIS TURJALEI und WINFRIED LAMERSDORF, Enabling Context-based Cooperation: A Generic Context Model and Management System. In *Proceedings of the 11th International Conference on Business Information Systems (BIS 2008)*, Seiten 459-470, Springer, 2008.

SONJA ZAPLATA Collaborative Management of Distributed Business Processes - A Service-Based Approach In *Proceedings of the On the Move to Meaningful Internet Systems (OTM) 2007 Workshops*, Seiten 304-313, Springer, 2007.

SONJA ZAPLATA und CHRISTIAN P. KUNZE, *Prozessmanagement im Mobile Computing - Kooperative Ausführung von Geschäftsprozessen im Umfeld serviceorientierter Architekturen*. ISBN 978-3-8364-1010-6, VDM Verlag Dr. Müller, 2007.

MICHAEL VON RIEGEN und SONJA ZAPLATA, *Supervising Remote Task Execution in Collaborative Workflow Environments*, In: TORSTEN BRAUN, GEORG CARLE und BURKHARD STILLER (Hrsg.) *Konferenzband zur KiVS 2007: Kommunikation in Verteilten Systemen - Band für Industriebeiträge, Kurzbeiträge und Workshops*, Seiten 337-358, VDE Verlag, 2007.

CHRISTIAN P. KUNZE, SONJA ZAPLATA und WINFRIED LAMERSDORF, *Abstrakte Dienstklassen zur Realisierung mobiler Prozesse*, In: TORSTEN BRAUN, GEORG CARLE und BURKHARD STILLER (Hrsg.) *Konferenzband zur KiVS 2007: Kommunikation in Verteilten Systemen - Band für Industriebeiträge, Kurzbeiträge und Workshops*, Seiten 123-128, VDE Verlag, 2007.

CHRISTIAN P. KUNZE, SONJA ZAPLATA und WINFRIED LAMERSDORF, *Mobile Processes: Enhancing Co-operation in Distributed Mobile Environments*, In: GEORGE J. SUN (Hrsg.), *Journal of Computers*, ISSN 1796-203X, 2(1), Seite 1-11, Academy Publisher, 2007.

CHRISTIAN P. KUNZE, SONJA ZAPLATA und WINFRIED LAMERSDORF, *Mobile Process Description and Execution*, In: FRANK ELIASSEN und ALBERTO MONTRESOR (Hrsg.), *Proceedings of the International Conference on Distributed Applications and Interoperable Systems (DAIS)*, Seiten 32-47, Springer, 2006.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[AAA+95] ALONSO, GUSTAVO, DIVYAKANT AGRAWAL, AMR EL ABBADI, C. MOHAN, ROGER GÜNTHÖR *und* MOHAN KAMATH: Exotica/FMQM: A Persistent Message-based Architecture for Distributed Workflow Management. *In:* Proceedings of the IFIP Workgroup Conference on Information Systems and Development for Decentralized Organizations (ISDO 1995), *Seiten 1–18. Chapman & Hall, 1995.*

[AAD+07a] AGRAWAL, ASHISH, MIKE AMEND, MANOJ DAS, MARK FORD, CHRIS KELLER, MATTHIAS KLOPPMANN, DIETER KÖNIG, FRANK LEYMANN, RALF MÜLLER, GERHARD PFAU, KARSTEN PLÖSSER, RAVI RANGASWAMY, ALAN RICKAYZEN, MICHAEL ROWLEY, PATRICK SCHMIDT, IVANA TRICKOVIC, ALEX YIU *und* MATTHIAS ZELLER: Web Services Human Task (WS-HumanTask) Version 1.0. *Technischer Bericht, Active Endpoints Inc., Adobe Systems Inc., BEA Systems Inc., International Business Machines Corporation, Oracle Inc., and SAP AG, 2007.*

[AAD+07b] AGRAWAL, ASHISH, MIKE AMEND, MANOJ DAS, MARK FORD, CHRIS KELLER, MATTHIAS KLOPPMANN, DIETER KÖNIG, FRANK LEYMANN, RALF MÜLLER, GERHARD PFAU, KARSTEN PLÖSSER, RAVI RANGASWAMY, ALAN RICKAYZEN, MICHAEL ROWLEY, PATRICK SCHMIDT, IVANA TRICKOVIC, ALEX YIU *und* MATTHIAS ZELLER: WS-BPEL Extension for People (BPEL4People) Version 1.0. *Technischer Bericht, Active Endpoints Inc., Adobe Systems Inc., BEA Systems Inc., International Business Machines Corporation, Oracle Inc., and SAP AG, 2007.*

[Aag01] AAGEDAL, JAN ØYVIND: Quality of Service Support in Development of Distributed Systems. *Doktorarbeit, University of Oslo, 2001.*

[ABCR06] AULETTA, VINCENZO, CARLO BLUNDO, EMILIANO DE CRISTOFARO *und* GUERRIERO RAIMATO: A Lightweight Framework for Web Services Invocation over Bluetooth. *In:* Procee-

dings of the IEEE International Conference on Web Services (ICWS06), *Seiten 331–338. IEEE Computer Society, 2006.*

[ABH85] ACHILLES, MANFRED, JÜRGEN BENDISCH *und* BERND HARTKOPF*:* Einführung in die Zeitreihen-Analyse mit ARIMA-Modellen. *Gesellschaft für Mathematik und Datenverarbeitung mbH, Sankt Augustin, 1985.*

[ABW06] ARASU, ARVIND, SHIVNATH BABU *und* JENNIFER WIDOM*:* The CQL Continuous Query Language: Semantic Foundations and Query Execution. *The VLDB Journal, 15:121–142, 2006.*

[ACD+04] ANDRIEUX, ALAIN, KARL CZAJKOWSKI, ASIT DAN, KATE KEAHEY, HEIKO LUDWIG, TOSHIYUKI NAKATA, JIM PRUYNE, JOHN ROFRANO, STEVE TUECKE *und* MING XU*:* Web Services Agreement Specification (WS-Agreement) – Draft. *Technischer Bericht, Open Grid Forum (OGF), 2004.*

[ACD+07] ANDRIEUX, ALAIN, KARL CZAJKOWSKI, ASIT DAN, KATE KEAHEY, HEIKO LUDWIG, TOSHIYUKI NAKATA, JIM PRUYNE, JOHN ROFRANO, STEVE TUECKE *und* MING XU*:* Web Services Agreement Specification (WS-Agreement). *Technischer Bericht, Open Grid Forum (OGF), 2007.*

[ACKM04] ALONSO, GUSTAVO, FABIO CASATI, HARUMI KUNO *und* VIJAY MACHIRAJU*:* Web Services: Concepts, Architecture and Applications. *Springer, Berlin, Heidelberg, 2004.*

[ACM01] ATLURI, VIJAYALAKSHMI, SOON AE CHUN *und* PIETRO MAZZOLENI*:* A Chinese Wall Security Model for Decentralized Workflow Systems. *In:* Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS 2001), *Seiten 48–57. ACM, 2001.*

[ACMM07] ATLURI, VIJAYALAKSHMI, SOON AE CHUN, RAVI MUKKAMALA *und* PIETRO MAZZOLENI*:* A Decentralized Execution Model for Inter-Organizational Workflows. *Distributed and Parallel Databases, 22(1):55–83, 2007.*

[AESW09] AMMON, RAINER VON, CHRISTOPH EMMERSBERGER, FLORIAN SPRINGER *und* CHRISTIAN WOLFF*:* Event-Driven Business Process Management and its Practical Application Taking the Example of DHL. *In:* 1st International Workshop on Complex Event Processing for the Future Internet. *CEUR-WS.org, 2009.*

[AFG+07] ARDISSONO, LILIANA, ROBERTO FURNARI, ANNA GOY, GIOVANNA PETRONE *und* MARINO SEGNAN: Context-aware Workflow Management. *In:* Proceedings of the 7th International Conference on Web Engineering (ICWE 2007), *Seiten 47–52. Springer, 2007.*

[AGIS05] ADELSTEIN, FRANK, SANDEEP KS GUPTA, GOLDEN RICHARD III *und* LOREN SCHWIEBERT: Fundamentals of Mobile and Pervasive Computing). *McGraw-Hill Professional, New York, 2005.*

[AGK+95] ALONSO, GUSTAVO, ROGER GÜNTHÖR, MOHAN KAMATH, DIVYAKANT AGRAWAL, AMR EL ABBADI *und* C. MOHAN: Exotica/FMDC: Handling Disconnected Clients in a Workflow Management System. *In:* Proceedings of the Third International Conference on Cooperative Information Systems (CoopIS-95), *Seiten 99–110, 1995.*

[AGK+96] ALONSO, GUSTAVO, ROGER GÜNTHÖR, MOHAN KAMATH, DIVYAKANT AGRAWAL, AMR EL ABBADI *und* C. MOHAN: Exotica/FMDC: A Workflow Management System for Mobile and Disconnected Clients. *Distributed and Parallel Databases, 4(3):229–247, 1996.*

[AGP08] ARDISSONO, LILIANA, ANNA GOY *und* GIOVANNA PETRONE: A Framework for the Development of Distributed, Context-Aware Adaptive Hypermedia Applications. *In:* Adaptive Hypermedia and Adaptive Web-Based Systems, *Seiten 259–262. Springer, 2008.*

[AH04] ABRAMS, MARC *und* JAMES HELMS: User Interface Modelling Language (UIML), Version 3.1, Specification. *Technischer Bericht, Organization for the Advancement of Structured Information Standards (OASIS), 2004.*

[All05] ALLWEYER, THOMAS: Geschäftsprozessmanagement. Strategie, Entwurf, Implementierung, Controlling. *W3L Verlag, Herdecke, Bochum, 2005.*

[Alt08] ALTER, STEVEN: Defining Information Systems as Work Systems: Implications for the IS Field. *European Journal of Information Systems (EJIS), 17(5):448–469, 2008.*

[And71] ANDERSON, T. W.: The Statistical Analysis of Time Series. *John Wiley and Sons, New York, 1971.*

[And03] ANDREWS, TONY AND CURBERA, FRANCISCO AND DHOLAKIA, HITESH AND GOLAND, YARON AND KLEIN, JOHANNES AND LEYMANN,FRANK AND LIU, KEVIN AND ROLLER, DIETER AND SMITH, DOUG AND THATTE, SATISH AND TRICKOVIC, IVANA AND WEERAWARANA, SANJIVA*:* Business Process Execution Language for Web Services - Version 1.1. *Technischer Bericht, BEA Systems, International Business Machines Corporation, Microsoft Corporation, SAP AG, Siebel Systems, 2003.*

[And04] ANDRESEN, ANDREAS*:* Komponentenbasierte Software-Entwicklung mit MDA, UML 2 und XML. *Carl Hanser Verlag, München, 2. Auflage, 2004.*

[AP06] ARDAGNA, DANILO *und* BARBARA PERNICI*:* Global and Local QoS Guarantee in Web Service Selection. *In:* Business Process Management Workshops*, Seiten 32–46. Springer, 2006.*

[Apa07] APACHE SOFTWARE FOUNDATION*:* Apache Muse 2.2.0 - A java-based implementation of WSRF 1.2, WSN 1.3, and WSDM 1.1. *http://ws.apache.org/muse, 2007.*

[APK+95] AVISON, DAVID*,* P.L. POWELL*,* J. KEEN*,* J.H. KLEIN *und* S. WARD*:* Addressing the Need for Flexibility in Information Systems. *Journal of Management Systems, 7(2):43–60, 1995.*

[ATEvdA06] ADAMS, MICHAEL*,* TER*,* DAVID EDMOND *und* WIL VAN DER AALST*:* Worklets: A Service-Oriented Implementation of Dynamic Flexibility in Workflows. *In:* On the Move to Meaningful Internet Systems (OTM 2006): CoopIS, DOA, GADA, and ODBASE*, Seiten 291–308. Springer, 2006.*

[B+11] BAEYENS, TOM *und* OTHERS*:* Activiti BPM Platform*, 2011. http://www.activiti.org.*

[Bad07] BADE, DIRK*:* Kontextabhängige und eigenverantwortliche Migration von Software-Agenten in heterogenen Umgebungen. *Diplomarbeit, Universität Hamburg, Fachbereich Informatik, Verteilte Systeme und Informationssysteme, 2007.*

[Bar84] BARENDREGT, HENDRIK PIETER*:* The Lambda Calculus – Its Syntax and Semantics. *North-Holland, Amsterdam, 1984.*

[BBB+07] BERSTEL, BRUNO*,* PHILIPPE BONNARD*,* FRANÇOIS BRY*,* MICHAEL ECKERT *und* PAULA-LAVINIA PĂTRÂNJAN*:* Reactive Rules on the Web*, Seiten 183–239. Springer, 2007.*

[BBF+08] BARTEL, MARK, JOHN BOYER, BARB FOX, BRIAN LAMACCHIA *und* ED SIMON*:* XML Signature Syntax and Processing. *Technischer Bericht, Word Wide Web Consortium (W3C), 2008.*

[BBM+01] BALLINGER, KEITH, PETER BRITTENHAM, ASHOK MALHOTRA, WILLIAM A. NAGY *und* STEFAN PHARIES*:* Web Services Inspection Language (WS-Inspection), Version 1.0. *Technischer Bericht, IBM, Microsoft, 2001.*

[BCC+04] BOX, DON, ERIK CHRISTENSEN, FRANCISCO CURBERA, DONALD FERGUSON, JEFFREY FREY, MARC HADLEY, CHRIS KALER, DAVID LANGWORTHY, FRANK LEYMANN, BRAD LOVERING, STEVE LUCCO, STEVE MILLET, NIRMAL MUKHI, MARK NOTTINGHAM, DAVID ORCHARD, JOHN SHEWCHUK, EUGÈNE SINDAMBIWE, TONY STOREY, SANJIVA WEERAWARANA *und* STEVE WINKLER*:* Web Services Addressing (WS-Addressing). *Technischer Bericht, World Wide Web Consortium (W3C), 2004.*

[BCF01] BERTINO, ELISA, SILVANA CASTANO *und* ELENA FERRARI*:* Securing XML documents with Author-X. *IEEE Internet Computing, 5(3):21–31, 2001.*

[BCvH+03] BILLINGTON, JONATHAN, SØREN CHRISTENSEN, KEES VAN HEE, EKKART KINDLER, OLAF KUMMER, LAURE PETRUCCI, REINIER POST, CHRISTIAN STEHNO *und* MICHAEL WEBER*:* The Petri Net Markup Language: Concepts, Technology, and Tools. *In:* 24th International Conference on Applications and Theory of Petri Nets*, Seiten 1023–1024, 2003.*

[BD97] BAUER, THOMAS *und* PETER DADAM*:* A Distributed Execution Environment for Large-Scale Workflow Management Systems with Subnets and Server Migration. *In:* IFCIS Conference on Cooperative Information Systems (CoopIS 1997)*, Seiten 99–108, 1997.*

[BD99] BAUER, THOMAS *und* PETER DADAM*:* Verteilungsmodelle für Workflow-Management-Systeme: Klassifikation und Simulation. *Informatik - Forschung und Entwicklung, 14:203–217, 1999.*

[BD00] BAUER, THOMAS *und* PETER DADAM*:* Efficient Distributed Workflow Management Based on Variable Server Assignments. *In:* 12th International Conference on Advanced Information Systems Engineering (CAiSE 2000)*, Seiten 94–109. Springer, 2000.*

[BD10] BRUNS, RALF *und* JÜRGEN DUNKEL*:* Event-Driven Architecture: Softwarearchitektur für ereignisgesteuerte Geschäftsprozesse*. Springer, Berlin, 2010.*

[BDG07] BARROS, ALISTAIR*,* GERO DECKER *und* ALEXANDER GROSSKOPF*:* Complex Events in Business Processes*. In:* Proceedings of the 10th International Conference on Business Information Systems (BIS 2007)*, BIS'07, Seiten 29–40. Springer-Verlag, 2007.*

[BDH+12] BARKHORDARIAN, ANGINEH*,* FREDERIK DEMUTH*,* KRISTOF HAMANN*,* MINH HOANG*,* SONJA WEICHLER *und* SONJA ZAPLATA*:* Migratability of BPMN 2.0 Process Instances*. In:* Service-Oriented Computing - ICSOC 2011 Workshops*, Seiten 66–75. Springer, 2012.*

[Ben04] BENGEL, GÜNTHER*:* Grundkurs verteilte Systeme*. Vieweg, Wiesbaden, 3. Auflage, 2004.*

[BG80] BERNSTEIN, PHILIP A. *und* NATHAN GOODMAN*:* Timestamp-based Algorithms for Concurrency Control in Distributed Database Systems*. In:* Proceedings of the 6th International Conference on Very Large Data Bases (VLDB 1980)*, Seiten 285–300. VLDB Endowment, 1980.*

[BG83] BERNSTEIN, PHILIP A. *und* NATHAN GOODMAN*:* The Failure and Recovery Problem for Replicated Databases*. In:* Proceedings of the 2nd Annual ACM Symposium on Principles of Distributed Computing (PODC 1983)*, Seiten 114–122. ACM, 1983.*

[BG05] BARESI, LUCIANO *und* SAM GUINEA*:* Towards Dynamic Monitoring of WS-BPEL Processes*. In:* 3rd International Conference of Service-Oriented Computing (ICSOC 2005)*, Band 3826, Seiten 269–282. Springer, 2005.*

[BGG04] BARESI, LUCIANO*,* CARLO GHEZZI *und* SAM GUINEA*:* Smart Monitors for Composed Services*. In:* Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC 2004)*, Seiten 193–202. ACM Press, 2004.*

[BH05] BECKER, KATIE *und* ERIC HERNESS*:* Use Common Event Infrastructure for business-level logging to improve business processes*. Technischer Bericht, IBM, 2005.*

[Böh09] BÖHLING, BENJAMIN*:* Management mobiler Prozesse: Log-

ging, Monitoring und Recovery. *Diplomarbeit, Universität Hamburg, Fachbereich Informatik, Verteilte Systeme und Informationssysteme, 7 2009.*

[BHM+04] BOOTH, DAVID, HUGO HAAS, FRANCIS MCCABE, ERIC NEWCOMER, MICHAEL CHAMPION, CHRIS FERRIS *und* DAVID ORCHARD: Web Services Architecture. *Technischer Bericht, W3C, 2004.*

[Bis06] BISHOP, CHRISTOPHER M.: Pattern Recognition and Machine Learning. *Information Science and Statistics. Springer, Berlin, 2006.*

[BK96] BROWN, NAT *und* CHARLIE KINDEL: Distributed Component Object Model Protocol – DCOM/1.0. *Technischer Bericht, Microsoft, 1996.*

[BK06] BRABÄNDER, ERIC *und* JÖRG KLÜCKMANN: Geschäftsprozessmanagement als Grundlage für SOA. *OBJEKTspektrum, 2006(5):32–40, 2006.*

[BKI08] BEIERLE, CHRISTOPH *und* GABRIELE KERN-ISBERNER: Regelbasierte Systeme. *In:* Methoden wissensbasierter Systeme, *Seiten 72–97. Vieweg+Teubner, 2008.*

[BKK+04] BENDER, MATTHIAS, STEFFEN KRAUS, FLORIAN KUPSCH, GERMAN SHEGALOV, GERHARD WEIKUM, DIRK WERTH *und* CHRISTIAN ZIMMER: Peer-to-Peer-Technologie für unternehmensweites und organisationsübergreifendes Workflow-Management. *In:* INFORMATIK 2004 - Informatik verbindet, Band 2, Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), *Seiten 511–516. Gesellschaft für Informatik, 2004.*

[BKNT09] BAUN, CHRISTIAN, MARCEL KUNZE, JENS NIMIS *und* STEFAN TAI: Cloud Computing: Web-basierte dynamische IT-Services. *Informatik im Fokus. Springer, Berlin Heidelberg New York, 2009.*

[BMM04] BARESI, LUCIANO, ANDREA MAURINO *und* STEFANO MODAFFERI: Workflow Partitioning in Mobile Information Systems. *In:* Modellierung betrieblicher Informationssyteme (MOBIS 2004), *Seiten 93–106. Gesellschaft für Informatik, 2004.*

[BMM06] BARESI, LUCIANO, ANDREA MAURINO *und* STEFANO MODAF-

FERI: Towards Distributed BPEL Orchestrations. *Electronic Communications of the EASST, 3:1–14, 2006.*

[BMNR03] BERGER, STEFAN, SCOTT MCFADDIN, CHANDRA NARAYANASWAMI *und* MANDAYAM RAGHUNATH: Web Services on Mobile Devices – Implementation and Experience. *In:* IEEE Workshop on Mobile Computing Systems and Applications, *Seiten 100–109. IEEE Computer Society, 2003.*

[BMW06] BULLARD, VAUGHN, BRYAN MURRAY *und* KIRK WILSON: An Introduction to WSDM. *Technischer Bericht, Organization for the Advancement of Structured Information Standards (OASIS), 2006.*

[Bor06] BORCHARDT, ANDREAS: Koordinationsinstrumente in virtuellen Unternehmen: eine empirische Untersuchung anhand lose gekoppelter Systeme. *Doktorarbeit, Universität zu Kiel, 2006.*

[Bow01] BOWMAN, IVAN T.: Hybrid Shipping Architectures: A Survey. *Technischer Bericht, University of Waterloo, 2001.*

[BPJ+10] BRAUBACH, LARS, ALEXANDER POKAHR, KAI JANDER, WINFRIED LAMERSDORF *und* BIRGIT BURMEISTER: Go4Flex: Goal-oriented Process Modelling. *In:* Proceedings of the 4th International Symposium on Intelligent Distributed Computing (IDC-2010), *Seiten 77–87. Springer, 2010.*

[BR98] BOGASCHEWSKY, RONALD *und* ROLAND ROLLBERG: Prozeßorientiertes Management. *Springer, Berlin, Heidelberg, 1998.*

[Bra87] BRATMAN, MICHAEL E.: Intention, Plans, and Practical Reason. *Harvard University Press, Cambridge, MA, USA, 1987.*

[Bra03] BRAUN, PETER: The Migration Process of Mobile Agents Implementation, Classification, and Optimization. *Doktorarbeit, Friedrich-Schiller-Universität Jena, 2003.*

[BRD01] BAUER, THOMAS, MANFRED REICHERT *und* PETER DADAM: Adaptives und verteiltes Workflow-Management. *In:* 9. GI-Fachtagung Datenbanksysteme in Büro, Technik und Wissenschaft (BTW 2001), *Seiten 47–66. Springer, 2001.*

[BS98] BORGHOFF, UWE M. *und* JOHANN H. SCHLICHTER: Rechnergestützte Gruppenarbeit. *Springer, Berlin, Heidelberg, 2. Auflage, 1998.*

[BSBB06] BURMEISTER, BIRGIT, HANS-PETER STEIERT, THOMAS BAUER *und* HARTWIG BAUMGÄRTEL*:* Agile Processes Through Goal- and Context-Oriented Business Process Modeling. *In:* Business Process Management Workshops*, Seiten 217–228. Springer, 2006.*

[BSR+06] BERBNER, RAINER, MICHAEL SPAHN, NICOLAS REPP, OLIVER HECKMANN *und* RALF STEINMETZ*:* Heuristics for QoS-aware Web Service Composition. *In:* International Conference on Web Services (ICWS 2006)*, Seiten 72 –82. IEEE Computer Society, 2006.*

[Bus08] BUSINESS RULES GROUP (BRG)*:* Semantics of Business Vocabulary and Business Rules (SBVR), Version 1.0. *Technischer Bericht, Object Management Group (OMG), 2008.*

[BV06a] BULLARD, VAUGHN *und* WILLIAM VAMBENEPE*:* Web Services Distributed Management: Management Using Web Services (WSDM-MUWS) 1.1 Part 1. *Technischer Bericht, Organization for the Advancement of Structured Information Standards (OASIS), 2006.*

[BV06b] BULLARD, VAUGHN *und* WILLIAM VAMBENEPE*:* Web Services Distributed Management: Management Using Web Services (WSDM-MUWS) 1.1 Part 2. *Technischer Bericht, Organization for the Advancement of Structured Information Standards (OASIS), 2006.*

[BvR08] BREVES, CARLO *und* EBERHARD VON RADETZKY*:* Anwendungsmigration im Rahmen von Beratungsprojekten. *Zeitschrift für Unternehmensberatung, 2008(6), 2008.*

[Car89] CARLSSON, BO*:* Flexibility and the Theory of the Firm. *International Journal of Industrial Organization, 7(2):179–203, 1989.*

[CCC07] CHO, YONGYUN, JONGSUN CHOI *und* JAEYOUNG CHOI*:* A Context-Aware Workflow System for a Smart Home. *In:* International Conference on Convergence Information Technology (ICCIT 2007)*, Seiten 95–100, 2007.*

[CCL03] CHLAMTAC, IMRICH, MARCO CONTI *und* JENNIFER J.-N. LIU*:* Mobile Ad-hoc Networking: Imperatives and Challenges. *Ad Hoc Networks, 1:13–64, 2003.*

[CCSC07] CHOI, JONGSUN, YONGYUN CHO, KYOUNGHO SHIN *und* JAEY-

OUNG CHOI*:* A Context-aware Workflow System for Dynamic Service Adaptation*. In:* Proceedings of the 2007 International Conference on Computational Science and its Applications (ICCSA 2007)*, Seiten 335–345. Springer, 2007.*

[CCSD04] CASTELLANOS, MALU*,* FABIO CASATI*,* MING-CHIEN SHAN *und* UMESHWAR DAYAL*:* Visibility and Measurability for Trust Management in Cooperative Business Operations*. In:* Proceedings of the IEEE International Conference on E-Commerce Technology for Dynamic E-Business (CEC-EAST 2004)*, Seiten 92–99. IEEE Computer Society, 2004.*

[CCSD05] CASTELLANOS, MALU*,* FABIO CASATI*,* MING-CHIEN SHAN *und* UMESH DAYAL*:* iBOM: A Platform for Intelligent Business Operation Management*. In:* Proceedings of the 21st International Conference on Data Engineering (ICDE 2005)*, Seiten 1084–1095. IEEE Computer Society, 2005.*

[CDK05] COULOURIS, GEORGE*,* JEAN DOLLIMORE *und* TIM KINDBERG*:* Distributed Systems: Concepts and Design*. Addison-Wesley, Boston, 2005.*

[CDPEV05] CANFORA, GERARDO*,* MASSIMILIANO DI PENTA*,* RAFFAELE ESPOSITO *und* MARIA LUISA VILLANI*:* An Approach for QoS-aware Service Composition Based on Genetic Algorithms*. In:* Proceedings of the 2005 Conference on Genetic and Evolutionary Computation (GECCO 2005)*, Seiten 1069–1075. ACM, 2005.*

[CEL06] CHAARI, TARAK*,* BRAHIM ELLOUMI *und* FRÉDÉRIQUE LAFOREST*:* A Generic Description Language for the Automatic Generation of Pervasive Medical User Interfaces: The SEFAGI Project*. In:* IEEE Health Pervasive Systems Workshop*, Seiten 1–6. IEEE Computer Society, 2006.*

[CELS07] CHAARI, TARAK*,* DEJENE EJIGU*,* FRÉDÉRIQUE LAFOREST *und* VASILE-MARIAN SCUTURICI*:* A Comprehensive Approach to Model and Use Context for Adapting Applications in Pervasive Environments*. Journal of Systems and Software, 80(12):1973–1992, 2007.*

[CFF+06] CZAJKOWSKI, K.*,* D. FERGUSON*,* I. FOSTER*,* J. FREY AN S. GRAHAM*,* I. SEDUKHIN*,* S. TUECKE *und* W. VAMBENEPE*:* The WS-Resource Framework*. Technischer Bericht, Globus Alliance, IBM, HP, 2006.*

[Cha06] CHANDY, MANI K.*:* Event-Driven Applications: Costs, Benefits and Design Approaches. *Gartner Application Integration and Web Services Summit 2006, 2006.*

[Cha07] CHANG, WILLIAM Y.*:* Network-Centric Service Oriented Enterprise. *Springer, New York, 2007.*

[Che01] CHEN, GUANLING AND KOTZ, DAVID*:* A Survey of Context-Aware Mobile Computing Research. *Technischer Bericht, Department of Computer Science, Dartmouth College, 2001.*

[Che04] CHEN, HARRY*:* An Intelligent Broker Architecture for Pervasive Context-Aware Systems. *Doktorarbeit, University of Maryland, Baltimore County, 2004.*

[CHvRR04] CLEMENT, LUC*,* ANDREW HATELY*,* CLAUS VON RIEGEN *und* TONY ROGERS*:* UDDI Version 3.0.2 Specification. *Technischer Bericht, Organization for the Advancement of Structured Information Standards (OASIS), 2004.*

[Cic99] CICHOCKI, ANDRZEJ*:* Migrating Workflows and Their Transactional Properties. *Doktorarbeit, University of Houston, Texas, 1999.*

[CL05] CHAARI, TARAK *und* FRÉDÉRIQUE LAFOREST*:* SEFAGI: Simple Environment For Adaptable Graphical Interfaces – Generating User Interfaces for Different Kinds of Terminals. *In:* International Conference on Enterprise Information Systems (ICEIS 2005)*, Seiten 232–237, 2005.*

[CM04] CHARFI, ANIS *und* MIRA MEZINI*:* Hybrid Web Service Composition: Business Processes Meet Business Rules. *In:* Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC 2004)*, Seiten 30–38. ACM, 2004.*

[CP07] CHANDE, SURESH *und* LASSE PAJUNEN*:* ActiveForms: A Runtime for Mobile Application Forms. *In:* Proceedings of the International Conference on the Management of Mobile Business*, Seiten 9–16. IEEE Computer Society, 2007.*

[CPV97] CARZANIGA, ANTONIO*,* GIAN PIETRO PICCO *und* GIOVANNI VIGNA*:* Designing Distributed Applications with Mobile Code Paradigms. *In:* Proceedings of the 19th International Conference on Software Engineering (ICSE 1997)*, Seiten 22–32. ACM, 1997.*

[CR97] CICHOCKI, ANDRZEJ *und* MAREK RUSINKIEWICZ*:* Migrating Workflows. *In:* Advances in Workflow Management Systems and Interoperability*, Seiten 311–326. NATO, 1997.*

[CR04] CICHOCKI, ANDRZEJ *und* MAREK RUSINKIEWICZ*:* Providing Transactional Properties for Migrating Workflows. *Mobile Networks and Applications, 9:473–480, 2004.*

[CRW98] CICHOCKI, ANDRZEJ*,* MAREK RUSINKIEWICZ *und* DARRELL WOELK*:* Workflow and Process Automation: Concepts and Technology. *Kluwer Academic Publishers, Norwell, MA, USA, 1998.*

[CSCC07] CHO, YONGYUN*,* KYOUNGHO SHIN*,* JONGSUN CHOI *und* JAEYOUNG CHOI*:* A Context-Adaptive Workflow Language for Ubiquitous Computing Environments. *In:* Computational Science and Its Applications (ICCSA 2007)*, Seiten 829–838. Springer, 2007.*

[CYCX05] CAO, JIANNONG*,* JIN YANG*,* WAI TING CHAN *und* CHENGZHONG XU*:* Exception Handling in Distributed Workflow Systems Using Mobile Agents. *In:* Proceedings of the IEEE International Conference on e-Business Engineering (ICEBE 2005)*, Seiten 48–55. IEEE Computer Society, 2005.*

[Dam02] DAMISCH, PETER NICOLAI*:* Wertorientiertes Flexibilitätsmanagement durch den Realoptionsansatz. *Deutscher Universitäts-Verlag, Wiesbaden, 2002.*

[Dav84] DAVIDSON, SUSAN B.*:* Optimism and Consistency in Partitioned Distributed Database Systems. *ACM Transactions on Database Systems, 9:456–481, 1984.*

[DB92] DOURISH, PAUL *und* VICTORIA BELLOTTI*:* Awareness and Coordination in Shared Workspaces. *In:* Proceedings of the 1992 ACM Conference on Computer-supported Cooperative work (CSCW 1992)*, Seiten 107–114. ACM, 1992.*

[DEF+08] DUNKEL, JÜRGEN*,* ANDREAS EBERHART*,* STEFAN FISCHER*,* CARSTEN KLEINER *und* ARNE KOSCHEL*:* Systemarchitekturen für Verteilte Anwendungen*. Hanser, München, 2008.*

[Dey99] DEY, ANIND K. AND ABOWD, GREGORY D.*:* Towards a Better Understanding of Context and Context-Awareness*. Technischer*

Bericht, Georgia Institute of Technology, College of Computing, 1999.

[Dey01] DEY, ANIND K.*:* Understanding and Using Context. *Personal and Ubiquitous Computing Journal, 5(1):4–7, 2001.*

[DFAB04] DIX, ALAN*,* JANET E. FINLAY*,* GREGORY D. ABOWD *und* RUSSELL BEALE*:* Human-Computer Interaction. *Pearson Education Limited / Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 3. Auflage, 2004.*

[DGB07] DECKER, GERO*,* ALEXANDER GROSSKOPF *und* ALISTAIR BARROS*:* A Graphical Notation for Modeling Complex Events in Business Processes. *In:* Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference*, Seiten 27–36. IEEE Computer Society, 2007.*

[DGH92] DAGUM, P.*,* A. GALPER *und* E. HORVITZ*:* Dynamic Network Models for Forecasting. *In:* Proceedings of International Conference on Uncertainty in Artificial Intelligence (UAI 92)*, Seiten 41–48. Morgan Kaufmann, 1992.*

[DGH03] DUSTDAR, SCHAHRAM*,* HARALD GALL *und* MANFRED HAUSWIRTH*:* Software-Architekturen für verteilte Systeme. *Springer, Berlin, Heidelberg, 2003.*

[DHP+05] DIAO, YIXIN*,* JOSEPH L. HELLERSTEIN*,* SUJAY PAREKH*,* REAN GRIFFITH*,* GAIL E. KAISER *und* DAN PHUNG*:* A Control Theory Foundation for Self-managing Computing Systems. *In:* Proceedings of the 12th International Conference and Workshops on the Engineering of Computer-based Systems (ECBS 2005)*, Seiten 441–448. IEEE Computer Society, 2005.*

[Die00] DIETTERICH, THOMAS G.*:* Ensemble Methods in Machine Learning. *In:* Proceedings of Multiple Classifier Systems (MCS 2000)*, Seiten 1–15. Springer, 2000.*

[DKB08] DECKER, GERO*,* OLIVER KOPP *und* ALISTAIR BARROS*:* An Introduction to Service Choreographies. *it - Information Technology, 50(2):122–127, 2008.*

[DKLW07] DECKER, GERO*,* OLIVER KOPP*,* FRANK LEYMANN *und* MATTHIAS WESKE*:* BPEL4Chor: Extending BPEL for Modeling Choreographies. *In:* Proceedings of the International Conference

on Web Services (ICWS 2007)*, Seiten 296–303. IEEE Computer Society, 2007.*

[DKN+06] DEARLE, ALAN, GRAHAM N.C. KIRBY, STUART J. NORCROSS, ANGUS D. MACDONALD *und* GREG J. BIGWOOD*:* Towards Adaptable and Adaptive Policy-Free Middleware*. Technischer Bericht, University of St Andrews, 2006.*

[DP06] DANIEL, FLORIAN *und* BARBARA PERNICI*:* Insights into Web Service Orchestration and Choreography*. International Journal of E-Business Research, 2(1):58–77, 2006.*

[Dre08] DREILING, VIKTOR*:* Web Services für mobile Systeme*. Bachelor-Arbeit, Universität Hamburg, Fachbereich Informatik, Verteilte Systeme und Informationssysteme, 2008.*

[DS05] DUSTDAR, SCHAHRAM *und* WOLFGANG SCHREINER*:* A Survey on Web Services Composition*. International Journal of Web and Grid Services, 1:1–30, 2005.*

[DS08] DOMSCHKE, WOLFGANG *und* ARMIN SCHOLL*:* Grundlagen der Betriebswirtschaftslehre*. Springer, Berlin, Heidelberg, 4. Auflage, 2008.*

[DSB10] DING, YONG, HEDDA R. SCHMIDTKE *und* MICHAEL BEIGL*:* Beyond Context-awareness: Context Prediction in an Industrial Application*. In:* Proceedings of the 12th ACM International Conference Adjunct Papers on Ubiquitous Computing (Ubicomp 2010)*, Seiten 401–402. ACM, 2010.*

[Dub07] DUBRAY, JEAN-JACQUES*:* WS-BPEL 2.0 Metamodel*. http://www.ebpml.org/wsper/wsper/ws-bpel20.html, 2007.*

[EB09] ECKERT, MICHAEL *und* FRANÇOIS BRY*:* Aktuelles Schlagwort: Complex Event Processing (CEP)*. Informatik Spektrum, 32(2):163–167, 2009.*

[Ede88] EDELMANN, HELMUT*:* Flexibilität durch Prognose. Dargestellt am Beispiel eines adaptiven Regressionsmodells zur (kurzfristigen) Stromverbrauchsprognose und -analyse*. Doktorarbeit, Universität Dortmund, 1988.*

[EF03] EBERHART, ANDREAS *und* STEFAN FISCHER*:* Web-Services: Grundlagen und praktische Umsetzung mit J2EE und .NET*. Hanser-Verlag, München, 2003.*

[EFGK03] EUGSTER, PATRICK TH., PASCAL A. FELBER, RACHID GUERRAOUI *und* ANNE-MARIE KERMARREC*:* The Many Faces of Publish/Subscribe*. ACM Computing Surveys, 35:114–131, 2003.*

[EGLT76] ESWARAN, KAPALI. P., JIM. N. GRAY, RAYMOND. A. LORIE *und* IRVING. L. TRAIGER*:* The Notions of Consistency and Predicate Locks in a Database System*. Communications of the ACM, 19:624–633, 1976.*

[EMT06] ERRADI, ABDELKARIM, PIYUSH MAHESHWARI *und* VLADIMIR TOSIC*:* Policy-driven Middleware for Self-adaptation of Web Services Compositions*. In:* Proceedings of the ACM/IFIP/USENIX 2006 International Conference on Middleware*, Seiten 62–80. Springer, 2006.*

[ENS07] EICKER, STEFAN, ANNETT NAGEL *und* PETER M. SCHULER*:* ICB-Reserch Report No. 21: Flexibilität im Geschäftsprozessmanagement-Kreislauf*. Technischer Bericht, Universität Duisburg, Institut für Informatik und Wirtschaftsinformatik (ICB), 2007.*

[Erl05] ERL, THOMAS*:* Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.*

[Esp07] ESPERTECH*:* Esper Complex Event Processing: Performance-Related Information*. http://docs.codehaus.org/display/ESPER/Esper+performance, 2007.*

[Esp11] ESPERTECH*:* Esper Reference Documentation, Version 4.5.0*. Documentation, 2011.*

[Esp12] ESPERTECH*:* Event Stream Intelligence: Esper and NEsper*. http://esper.codehaus.org, 2012.*

[ETM07] ERRADI, ABDELKARIM, VLADIMIR TOSIC *und* PIYUSH MAHESHWARI*:* MASC - .NET-Based Middleware for Adaptive Composite Web Services*. In:* IEEE International Conference on Web Services (ICWS 2007)*, Seiten 727–734. IEEE Computer Society, 2007.*

[Etz05] ETZION, OPHER*:* Towards an Event-Driven Architecture: An Infrastructure for Event Processing*. In:* 1st International Con-

fernce on Rules and Rule Markup Languages for the Semantic Web (RuleML 2005)*, Seiten 1–7. Springer, 2005.*

[Eur09] EUROPEAN ASSOCIATION OF BPM*:* Guide to the Business Process Management Common Body of Knowledge*. Verlag Dr. Götz Schmidt, München, 2009.*

[Fer07] FERSCHA, ALOIS*:* Pervasive Computing - connected, aware, smart*. In:* MATTERN, FRIEDEMANN *(Herausgeber):* Die Informatisierung des Alltags*, Seiten 3–10. Springer, 2007.*

[FG08] FREUND, JAKOB *und* KLAUS GÖTZER*:* Vom Geschäftsprozess zum Workflow: Ein Leitfaden für die Praxis*. Hanser Fachbuchverlag, München, 2008.*

[FGM+99] FIELDING, ROY T.*,* JIM GETTYS*,* JEFFREY C. MOGUL*,* HENRIK FRYSTYK*,* LARRY MASINTER*,* PAUL LEACH *und* TIM BERNERS-LEE*:* Hypertext Transfer Protocol – HTTP/1.1*. Technischer Bericht, Internet Engineering Task Force (IETF), 1999.*

[FGS96] FARMER, WILLIAM M.*,* JOSHUA D. GUTTMAN *und* VIPIN SWARUP*:* Security for Mobile Agents: Issues and Requirements*. In:* Proceedings of the National Information Systems Security Conference*, Seiten 591–597, 1996.*

[FH93] FEILER, PETER H. *und* WATTS S. HUMPHREY*:* Software Process Development and Enactment: Concepts and Definitions*. In:* Proceeding of the 2nd International Conference on the Software Process*, Seiten 28–40, 1993.*

[Fin09] FINGAR, PETER*:* Dot Cloud: The 21st Century Business Platform Built on Cloud Computing*. Meghan-Kiffer Press, Tampa, USA, 2009.*

[FIP04] FIPA*:* FIPA Agent Management Specification*. Technischer Bericht, Foundation for Intelligent Physical Agents (FIPA), 2004.*

[FKT08] FÄHNRICH, KLAUS-PETER*,* STEFAN KÜHNE *und* MAIK THRÄNERT *(Herausgeber):* Model-Driven Integration Engineering. Modellierung, Validierung und Transformation zur Integration betrieblicher Anwendungssysteme*. Leipziger Informatik-Verbund (LIV), Leipzig, 2008.*

[FKZ08] FENSEL, DIETER*,* MICK KERRIGAN *und* MICHAL ZAREMBA *(Herausgeber):* Implementing Semantic Web Services: The SE-

SA Framework. *Springer, Berlin, Heidelberg, 2008.*

[Flo97] FLOYD, CHRISTIANE*:* Autooperationale Form und situiertes Handeln. *In:* Cognitio Humana - Dynamik des Wissens und der Werte*, Seiten 237–252, Berlin, 1997. Akademie Verlag.*

[FPV98] FUGGETTA, ALFONSO*,* GIAN PIETRO PICCO *und* GIOVANNI VIGNA*:* Understanding Code Mobility. *IEEE Transactions on Software Engineering, 24(5):342–361, 1998.*

[FRH10] FREUND, JAKOB*,* BERND RÜCKER *und* THOMAS HENNINGER*:* Praxishandbuch BPMN [inklusive BPMN 2.0]. *Hanser, München, 2010.*

[FUYP04] FAN, XIAOCONG*,* KARTHIKEYAN UMAPATHY*,* J. YEN *und* SANDEEP PURAO*:* Team-based Agents for Proactive Failure Handling in Dynamic Composition of Web Services. *In:* Proceedings of the IEEE International Conference on Web Services (ICWS 2004)*, Seiten 782–785. IEEE Computer Society, 2004.*

[Gad10] GADATSCH, ANDREAS*:* Grundkurs Geschäftsprozess-Management. *Vieweg Teubner, Wiesbaden, 6. Auflage, 2010.*

[GALH00] GREFEN, PAUL*,* KARL ABERER*,* HEIKO LUDWIG *und* YIGAL HOFFNER*:* CrossFlow: Cross-organizational workflow management in dynamic virtual enterprises. *International Journal of Computer Systems Science & Engineering, 15(5):277–290, 2000.*

[GCC+04] GRIGORI, DANIELA*,* FABIO CASATI*,* MALU CASTELLANOS*,* UMESHWAR DAYAL*,* MEHMET SAYAL *und* MING-CHIEN SHAN*:* Business Process Intelligence. *Computers in Industry, 53:321–343, 2004.*

[GCK+02] GRAY, ROBERT S.*,* GEORGE CYBENKO*,* DAVID KOTZ*,* RONALD A. PETERSON *und* DANIELA RUS*:* D'Agents: Applications and Performance of a Mobile-Agent System. *Software, Practice and Experience, 32:543–573, 2002.*

[GG99] GEHRING, HERMANN *und* ANDREAS GADATSCH*:* Eine Rahmenarchitektur für Workflow-Management-Systeme. *Technischer Bericht, Diskussionsbeiträge des Fachbereichs Wirtschaftswissenschaft, Fernuniversität Hagen, 1999.*

[GHHW01] GOODGER, BEN*,* IAN HICKSON*,* DAVID HYATT *und* CHRIS WATERSON*:* XML User Interface Language (XUL), Version 1.0,

Specification. *Technischer Bericht, Mozilla.org, 2001.*

[GHJJ10] GAMMA, ERICH, RICHARD HELM, RALPH JOHNSON *und* VLISSIDES JOHN: Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software. *Addison-Wesley, Bonn, 6. Auflage, 2010.*

[GHM06] GRAHAM, STEVE, DAVID HULL *und* BRYAN MURRAY: Web Services Base Notification 1.3 (WS-BaseNotification). *Technischer Bericht, Organization for the Advancement of Structured Information Standards (OASIS), 2006.*

[GMR+05] GREINER, U., R. MUELLER, E. RAHM, J. RAMSCH, B. HELLER *und* M. LOEFFLER: AdaptFlow: Protocol-based Medical Treatment Using Adaptive Workflows. *Methods of Information in Medicine, 44(1):80–88, 2005.*

[Gün09] GÜNTHER, RICHARD: Bezahlverfahren für die Nutzung elektronischer Dienste im Mobile Computing. *Bachelor-Arbeit, Universität Hamburg, Fachbereich Informatik, Verteilte Systeme und Informationssysteme, 2009.*

[Gol09] GOLTERMANN, LEIF: Ein kontextbasiertes Look-Ahead-Verfahren für die Migration von Prozessen im Mobile Computing. *Diplomarbeit, Universität Hamburg, Fachbereich Informatik, Verteilte Systeme und Informationssysteme, 2009.*

[GP00] GOLDEN, WILLIAM *und* PHILIP POWELL: Towards a Definition of Flexibility: In Search of the Holy Grail? *OMEGA - The International Journal of Management Science, 28(4):373–384, 2000.*

[GR92] GRAY, JIM *und* ANDREAS REUTER: Transaction Processing: Concepts and Techniques. *Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992.*

[GR07] GREENWOOD, DOMINIC *und* GIOVANNI RIMASSA: Autonomic Goal-Oriented Business Process Management. *In:* Proceedings of the 3rd International Conference on Autonomic and Autonomous Systems (ICAC 2007), *Seiten 43–49. IEEE Computer Society, 2007.*

[Gra07] GRAHAM, IAN: Business Rules Management and Service Oriented Architecture: A Pattern Language. *John Wiley & Sons, Chichester, 2007.*

[GRCB05] GUO, LI, DAVID ROBERTSON *und* YUN-HEH CHEN-BURGER*:* A Generic Multi-agent System Platform for Business Workflows using Web Services Composition*. In:* International Conference on Intelligent Agent Technology*, Seiten 301–307. IEEE Computer Society, 2005.*

[Gre06] GREFEN, PAUL W.P.J.*:* Towards Dynamic Interorganizational Business Process Management*. In:* Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Seiten 13–20. IEEE Computer Society, 2006.*

[Gry96] GRYCZAN, GUIDO*:* Prozessmuster zur Unterstützung kooperativer Tätigkeit*. DUV, Wiesbaden, 1996.*

[GS05] GEBAUER, JUDITH *und* FRANZ SCHOBER*:* Information System Flexibility and the Performance of Business Processes*. Technischer Bericht, University of Illinois, 2005.*

[GS06] GEBAUER, JUDITH *und* FRANZ SCHOBER*:* Information System Flexibility and the Cost Efficiency of Business Processes*. Journal of the Association for Information Systems, 7(3), 2006.*

[GT06] GRAHAM, STEVE *und* JEM TREADWELL*:* Web Services Resource Properties 1.2 (WS-ResourceProperties)*. Technischer Bericht, Organization for the Advancement of Structured Information Standards (OASIS), 2006.*

[Hac05] HACH, HENNING*:* Evaluation und Optimierung kommunaler E-Government Prozesse*. Doktorarbeit, Universität Flensburg, Internationales Institut für Management, 2005.*

[Hal01] HALLE, BARBARA VON*:* Business Rules Applied: Building Better Systems Using the Business Rules Approach*. John Wiley & Sons, Inc., New York, NY, USA, 2001.*

[Ham05] HAMMERSCHALL, ULRIKE*:* Verteilte Systeme und Anwendungen - Architekturkonzepte, Standards und Middleware-Technologien*. Pearson Studium, München, 2005.*

[Ham09] HAMANN, KRISTOF*:* Parallele Ausführung von Prozessen auf mobilen Geräten*. Diplomarbeit, Universität Hamburg, Department Informatik, Arbeitsbereich Verteilte Systeme und Informationssysteme, 2009.*

[Han05] HAN, JIAWEI: Data Mining: Concepts and Techniques. *Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.*

[Hüb03] HÜBNER, GERHARD: Stochastik - Eine anwendungsorientierte Einführung für Informatiker, Ingenieure und Mathematiker. *Friedr. Vieweg & Sohn, Braunschweig, 2003.*

[HBS+02] HAPNER, MARK, RICH BURRIDGE, RAHUL SHARMA, JOSEPH FIALLI *und* KATE STOUT: Java Message Service, Version 1.1. *Technischer Bericht, Sun Microsystems, 2002.*

[HCKC06] HAN, JOOHYUN, YONGYUN CHO, EUNHOE KIM *und* JAEYOUNG CHOI: A Ubiquitous Workflow Service Framework. *In:* Computational Science and its Applications (ICCSA 2006), *Seiten 30–39. Springer, 2006.*

[HDVL03] HELAL, SUMI, NITIN DESAI, VARUN VERMA *und* CHOONHWA LEE: Konark – A Service Discovery and Delivery Protocol for Ad-hoc Networks. *Wireless Communications and Networking, 3:2107–2113, 2003.*

[Hen08] HENNING, MICHI: The Rise and Fall of CORBA. *Communications of the ACM, 51(8):52–57, 2008.*

[HGR07] HACKMANN, GREGORY, CHRISTOPHER D. GILL *und* GRUIA-CATALIN ROMAN: Extending BPEL for Interoperable Pervasive Computing. *In:* Proceedings of IEEE International Conference on Pervasive Services 2007 (ICPS 2007), *Seiten 204–213. IEEE Computer Society, 2007.*

[HH00] HAY, DAVID *und* KERI ANDERSON HEALY: Defining Business Rules – What Are They Really? Final Report. *Technischer Bericht, Business Rules Group (BRG), 2000.*

[HHGR06] HACKMANN, GREGORY, MART HAITJEMA, CHRISTOPHER D. GILL *und* GRUIA-CATALIN ROMAN: Sliver: A BPEL Workflow Process Execution Engine for Mobile Devices. *In:* International Conference on Service-Oriented Computing (ICSOC 2006), *Seiten 503–508. Springer, 2006.*

[HHJ+99] HEINL, PETRA, STEFAN HORN, STEFAN JABLONSKI, JENS NEEB, KATRIN STEIN *und* MICHAEL TESCHKE: A Comprehensive Approach to Flexibility in Workflow Management Systems. *ACM SIGSOFT Software Engineering Notes, 24(2):79–88, 1999.*

[Hil95] HILARIO, MÉLANIE*:* An Overview Of Strategies For Neurosymbolic Integration*. In:* Connectionist-Symbolic Integration*, Seiten 13–35. Lawrence Erlbaum Assoc., 1995.*

[Hil07] HILDEBRAND, KNUT*:* IT-Integration und Migration: Konzepte und Vorgehensweisen*. Dpunkt, Heidelberg, 2007.*

[HLGA01] HOFFNER, Y.*,* H. LUDWIG*,* P.W.P.J. GREFEN *und* K. ABERER*:* CrossFlow: Integrating Workflow Management and Electronic Commerce*. ACM SIGecom Exchange, 2(1):1–10, 2001.*

[HLGG00] HOFFNER, YIGAL*,* HEIKO LUDWIG*,* CEKI GÜLCÜ *und* PAUL GREFEN*:* An Architecture for Cross-Organizational Business Processes*. In:* Proceedings of the 2nd International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS 2000)*, Seiten 125–134. IEEE Computer Society, 2000.*

[HM08] HUEBSCHER, MARKUS C. *und* JULIE A. MCCANN*:* A Survey of Autonomic ComputingDegrees, Models, and Applications*. ACM Computing Surveys, 40:7:1–7:28, 2008.*

[Hün08] HÜNDLING, JENS*:* Modellierung von Qualitätsmerkmalen für Services*. Doktorarbeit, Universität Potsdam, Mathematisch-Naturwissenschaftlichen Fakultät, 2008.*

[Hol95] HOLLINGSWORTH, DAVID*:* The Workflow Reference Model*. Technischer Bericht, Workflow Managament Coalition (WfMC), 1995.*

[Hol07] HOLBREICH, ALEXANDER*:* Transaktionsunterstützung für verteilt ausgeführte Mobile Prozesse*. Diplomarbeit, Universität Hamburg, Fachbereich Informatik, Verteilte Systeme und Informationssysteme, 2007.*

[HPA05] HEINIS, THOMAS*,* CESARE PAUTASSO *und* GUSTAVO ALONSO*:* Design and Evaluation of an Autonomic Workflow Engine*. In:* Proceedings of the 2nd International Conference on Autonomic Computing*, Seiten 27–38. IEEE Computer Society, 2005.*

[HPA06] HEINIS, THOMAS*,* CESARE PAUTASSO *und* GUSTAVO ALONSO*:* A Self-Configuring Service Composition Engine*. In:* Autonomic Computing: Concepts, Infrastructure, and Applications*, Seiten 237–251. CRC Press, 2006.*

[HSH+06] HACKMANN, GREGORY, ROHAN SEN, MART HAITJEMA, GRUIA-CATALIN ROMAN *und* CHRISTOPHER GILL*:* MobiWork: Mobile Workflow for MANETs. *Technischer Bericht, Washington University, 2006.*

[HSZ11] HAMANN, KRISTOF, SEBASTIAN STEENBUCK *und* SONJA ZAPLATA*:* Towards NFC-Aware Process Execution for Dynamic Environments. *In:* Workshops der wissenschaftlichen Konferenz Kommunikation in verteilten Systemen 2011 (WowKiVS 2011)*, Seiten 1–12. Electronic Communications of the EASST, 2011.*

[HT05] HOU, XIAOSONG *und* CHIK HOW TAN*:* A New Electronic Cash Model. *In:* Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC 2005)*, Seiten 374–379. IEEE Computer Society, 2005.*

[IBM04] IBM*:* Common Base Event. *Technischer Bericht, IBM, 2004.*

[IDS+12] IMAMURA, TAKESHI, BLAIR DILLAWAY, ED SIMON, KELVIN YIU *und* MAGNUS NYSTRÖM*:* XML Encryption Syntax and Processing Version 1.1. *Technischer Bericht, Word Wide Web Consortium (W3C), 2012.*

[Ill07] ILLIK, J. ANTON*:* Verteilte Systeme: Architekturen und Software-Technologien). *Expert Verlag, Renningen, 2007.*

[IT04] ITU-T*:* Generic Applications of ASN.1: Fast Web Services. *Technischer Bericht, International Telecommunication Union, 2004.*

[IWKK00] ILLMANN, TORSTEN, MICHAEL WEBER, FRANK KARGL *und* TILMANN KRÜGER*:* Migration of Mobile Agents in Java: Problems, Classification and Solutions. *In:* Proceedings of the International ICSC Symposium on Multi-Agents and Mobile Agents in Virtual Organizationa and E-Commerce (MAMA 2000. *Springer, 2000.*

[Jab97] JABLONSKI, STEFAN*:* Architektur von Workflow-Management-Systemen. *Informatik - Forschung und Entwicklung, 12(2):72–81, 1997.*

[JBF07] JURCA, RADU, WALTER BINDER *und* BOI FALTINGS*:* Reliable and Inexpensive QoS Monitoring in Service Markets. *ERCIM News, 2007(70), 2007.*

[JBo07] JBOSS INC.*:* JBoss jBPM jPDL 3.2 – jBPM jPDL User Guide. *http://docs.jboss.com/jbpm/v3/userguide/index.html, 2007.*

[Jem06] JEMIOLO, DAN*:* Web Services Resource Metadata 1.0 (WS-ResourceMetadataDescriptor). *Technischer Bericht, Organization for the Advancement of Structured Information Standards (OASIS), 2006.*

[JF05] JURCA, RADU *und* BOI FALTINGS*:* Reputation-based Service Level Agreements for Web Services. *In:* In Proceedings of the 3rd International Conference on Service Oriented Computing (ICSOC 2005)*, Seiten 396–409. Springer-Verlag, 2005.*

[JFB07] JURCA, RADU*,* BOI FALTINGS *und* WALTER BINDER*:* Reliable QoS Monitoring Based on Client Feedback. *In:* Proceedings of the 16th International Conference on World Wide Web (WWW 2007)*, Seiten 1003–1012. ACM, 2007.*

[JHH+00] JING, JIN*,* KAREN HUFF*,* BEN HURWITZ*,* HIMANSHU SINHA*,* BILL ROBINSON *und* MARK FEBLOWITZ*:* WHAM: Supporting Mobile Workforce and Applications in Workflow Environments. *In:* Proceedings of the 10th International Workshop on Research Issues in Data Engineering (RIDE 2000)*, Seiten 31–38. IEEE Computer Society, 2000.*

[JNDV99] JONKER, WILLEM*,* WIM NIJENHUIS*,* ZEF DAMEN *und* MARTIN VERWIJMEREN*:* Workflow Management Systems and Cross-Organisational Logistics. *In:* Cross-Organisational Workflow Management and Co-ordination*, Seiten 1–6, 1999.*

[Job10] JOBST, DANIEL*:* Service- und Ereignisorientierung im Contact-Center – Entwicklung eines Referenzmodells zur Prozessautomatisierung. *Gabler Verlag, Wiesbaden, 2010.*

[Joe00] JOERIS, GREGOR*:* Flexibles und adaptives Workflowmanagement für verteilte und dynamische Prozesse. *Doktorarbeit, Universität Bremen, Technologie-Zentrum Informatik, 2000.*

[Jör06] JÖRNS, CARSTEN*:* Business Process Implementation – mehr als nur IT-Realisierung. *In:* Agilität durch ARIS Geschäftsprozessmanagement*, Seiten 173–187. Springer, 2006.*

[JSH+01] JABLONSKI, STEFAN*,* RALF SCHAMBURGER*,* CHRISTIAN HAHN*,* STEFAN HORN*,* RAINER LAY*,* JENS NEEB *und* MICHAEL SCHLUNDT*:* A Comprehensive Investigation of Distribution

in the Context of Workflow Management. *In:* Proceedings of the 8th International Conference on Parallel and Distributed Systems (ICPADS 2001)*, Seiten 187–192. IEEE Computer Society, 2001.*

[Kal93] KALUZA, BERND*:* Betriebliche Flexibilität*, Seiten 1173–1184. Poeschel-Verlag, 1993.*

[KB04a] KALUZA, BERND *und* THORSTEN BLECKER*:* Flexibilität - State of the Art und Entwicklungstendenzen*, Seiten 1 – 25. Erich Schmidt Verlag, 2004.*

[KB04b] KARASTOYANOVA, DIMKA *und* ALEJANDRO BUCHMANN*:* Development Life Cycle of Web Service-based Business Processes: Enabling Dynamic Invocation of Web Services at Run Time. *In:* Proceedings of the 2nd International Workshop on Web Services: Modeling, Architecture and Infrastructure (WSMAI 2004)*, Seiten 9–22. ICEIS Press, 2004.*

[KC03] KEPHART, JEFFREY O. *und* DAVID M. CHESS*:* The Vision of Autonomic Computing. *Computer, 36:41–50, 2003.*

[KEP00] KNOLMAYER, GERHARD*,* RAINER ENDL *und* MARCEL PFAHRER*:* Modeling Processes and Workflows by Business Rules. *In:* Business Process Management, Models, Techniques, and Empirical Studies*, Seiten 16–29. Springer, 2000.*

[KG99] KRADOLFER, MARKUS *und* ANDREAS GEPPERT*:* Dynamic workflow schema evolution based on workflow type versioning and workflow migration. *In:* Proceedings of IFCIS International Conference on Cooperative Information Systems (CoopIS 1999)*, Seiten 104–114. IEEE Computer Society, 1999.*

[Kha08] KHALAF, RANIA*:* Supporting Business Process Fragmentation While Maintaining Operational Semantics: A BPEL Perspective. *Doktorarbeit, Universität Stuttgart, Institut für Architektur von Anwendungssystemen, 2008.*

[KHC+05] KARASTOYANOVA, DIMKA*,* ALEJANDRO HOUSPANOSSIAN*,* MARIANO CILIA*,* FRANK LEYMANN *und* ALEJANDRO BUCHMANN*:* Extending BPEL for Run Time Adaptability. *In:* IEEE International Conference on Enterprise Distributed Object Computing (EDOC 2005)*, Seiten 15–26. IEEE Computer Society, 2005.*

[Kin04] KINDLER, EKKARD*:* Using the Petri Net Markup Language for Exchanging Business Processes – Potential and Limitations. *In:* Proceedings of the 1st GI Workshop of XML Interchange Formats for Business Process Management (XML4BPM 2004)*, Seiten 43–60, 2004.*

[KKJ07] KLEINE, OLIVER, STEFFEN KINKEL *und* ANGELA JÄGER*:* Flexibilität durch Technologieeinsatz? Nutzung und Erfolgswirkung flexibilitätsfördernder Technologien. *Technischer Bericht, Fraunhofer Institute for Systems and Innovation Research (ISI), 2007.*

[KKL+05] KLOPPMANN, MATTHIAS, DIETER KÖNIG, FRANK LEYMANN, GERHARD PFAU, ALAN RICKAYZEN, CLAUS VON RIEGEN, PATRICK SCHMIDT *und* IVANA TRICKOVIC*:* WS-BPEL Extension for People (BPEL4People). *Technischer Bericht, IBM, SAP, 2005.*

[KL06] KHALAF, RANIA *und* FRANK LEYMANN*:* A Role-based Decomposition of Business Processes using BPEL. *In:* IEEE International Conference on Web Services (ICWS 2006)*, Seiten 770–780. IEEE Computer Soiciety, 2006.*

[Kla04] KLAMAR, SEBASTIAN*:* Adaptive Architekturen für verteilte Systeme. *Diplomarbeit, Technische Universität Dresden, 2004.*

[Kle01] KLENSIN, JOHN C.*:* Simple Mail Transfer Protocol. *Technischer Bericht, Internet Engineering Task Force (IETF), 2001.*

[Koc00] KOCK, NED*:* Benefits for virtual organizations from distributed groups. *Communications of the ACM, 43(11):107–112, 2000.*

[Kot10] KOTTKE, KRISTIAN*:* Zugriffsschutz für Dienste in verteilten mobilen Systemen. *Diplomarbeit, Universität Hamburg, Fachbereich Informatik, Verteilte Systeme und Informationssysteme, 2010.*

[KPF01] KANG, MYONG H., JOON S. PARK *und* JUDITH N. FROSCHER*:* Access Control Mechanisms for Inter-organizational Workflow. *In:* Proceedings of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT 2001)*, Seiten 66–74. ACM, 2001.*

[KR81] KUNG, H. T. *und* JOHN T. ROBINSON*:* On Optimistic Methods for Concurrency Control. *ACM Transactions on Database Sy-*

stems, 6(2):213–226, 1981.

[Krö10] KRÖSCHEL, IVONNE*:* On the Notion of Context for Business Process Use. *In:* 3rd International Conference on Business Process and Services Computing (BPSC 2010)*, Seiten 288–297. Gesellschaft für Informatik, 2010.*

[Kri99] KRIEBEL, STEFAN K. T.*:* A Combined Parametric and Nonparametric Approach To Time Series Analysis. *Akad. Verl.-Ges., Leipzig, 1999.*

[KS97] KANTZ, HOLGER *und* THOMAS SCHREIBER*:* Nonlinear time series analysis. *Cambridge University Press, Cambridge, MA, USA, 1997.*

[KS10] KRESS, MARKUS *und* DETLEF SEESE*:* Autonomous Optimization of Business Processes. *In:* Business Process Management Workshops: Revised Papers of the BPM 2009 International Workshops*, Band 43, Seiten 116–127. Springer, 2010.*

[KSK07] KIKUCHI, SHINJI*,* HISASHI SHIMAMURA *und* YOSHIHIRO KANNA*:* Monitoring Method of Cross-Sites' Processes Executed by Multiple WS-BPEL Processors. *In:* 9th International Conference on E-Commerce Technology and 4th International Conference on Enterprise Computing, E-Commerce and E-Services (CEC-EEE 2007)*, Seiten 55–64. IEEE Computer Society, 2007.*

[KT01a] KARAGIANNIS, DIMITRIS *und* RAINER TELESKO*:* Wissensmanagement: Konzepte der Künstlichen Intelligenz und des Softcomputing. *Oldenbourg, München, Wien, 2001.*

[KT01b] KARNIK, NEERAN M. *und* ANAND R. TRIPATHI*:* Security in the Ajanta Mobile Agent System. *Software – Practice and Experience, 31(4):301–329, 2001.*

[Kun05] KUNZE, CHRISTIAN P.*:* DEMAC: A Distributed Environment for Mobility-Aware Computing. *In:* Adjunct Proceedings of the Third International Conference on Pervasive Computing*, Seiten 115–121. Österreichische Computer Gesellschaft, 5 2005.*

[Kun08] KUNZE, CHRISTIAN P.*:* Kontextbasierte Kooperation: Unterstützung verteilter Prozesse im Mobile Computing. *Doktorarbeit, Universität Hamburg, Fachbereich Informatik, Verteilte Systeme und Informationssysteme, 2008.*

[KW05] KUPSCH, FLORIAN *und* DIRK WERTH*:* Integrating Business Processes with Peer-to-Peer Technology. *In:* 1st International Conference on Interoperability of Enterprise Software and Applications (INTEROP ESA 2005)*, Seiten 277–288. Springer, 2005.*

[KWA99a] KLINGEMANN, JUSTUS*,* JURGEN WASCH *und* KARL ABERER*:* Deriving Service Models in Cross-Organizational Workflows. *In:* Proceedings of the Ninth International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises (RIDE 1999)*, Seiten 100–107. IEEE Computer Society, 1999.*

[KWA99b] KLINGEMANN, JUSTUS*,* JÜRGEN WÄSCH *und* KARL ABERER*:* Adaptive Outsourcing in Cross-Organizational Workflows. *In:* Advanced Information Systems Engineering*, Seiten 417–421. Springer, 1999.*

[KWL09] KOPP, OLIVER*,* MATTHIAS WIELAND *und* FRANK LEYMANN*:* Towards Choreography Transactions. *In:* Proceedings of the 1st Central European Workshop on Services and their Composition (ZEUS 2009)*, Seiten 49–54. CEUR-WS.org, 2009.*

[KZL06] KUNZE, CHRISTIAN P.*,* SONJA ZAPLATA *und* WINFRIED LAMERSDORF*:* Mobile Process Description and Execution. *In:* Proceedings of the 6th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS 2006)*, Seiten 32 – 47. Springer, 2006.*

[KZL07a] KUNZE, CHRISTIAN P.*,* SONJA ZAPLATA *und* WINFRIED LAMERSDORF*:* Abstrakte Dienstklassen zur Realisierung mobiler Prozesse. *In:* Konferenzband zur KiVS 2007 für Industrie-, Kurz- und Workshopbeiträge*, Seiten 123 – 128. VDE Verlag, 2007.*

[KZL07b] KUNZE, CHRISTIAN P.*,* SONJA ZAPLATA *und* WINFRIED LAMERSDORF*:* Mobile Processes: Enhancing Cooperation in Distributed Mobile Environments. *Journal of Computers, 2(1):1–11, 2007.*

[KZTL08] KUNZE, CHRISTIAN P.*,* SONJA ZAPLATA*,* MIRWAIS TURJALEI *und* WINFRIED LAMERSDORF*:* Enabling Context-based Cooperation: A Generic Context Model and Management System. *In:* Business Information Systems (BIS 2008)*. Springer, 2008.*

[LCO+10] LERNER, BARBARA STAUDT, STEFAN CHRISTOV, LEON J. OSTERWEIL, REDA BENDRAOU, UDO KANNENGIESSER *und* ALEXANDER WISE*:* Exception Handling Patterns for Process Modeling. *IEEE Transactions on Software Engineering, 99(RapidPosts):162–183, 2010.*

[LDK04] LUDWIG, HEIKO, ASIT DAN *und* ROBERT KEARNEY*:* Cremona: An Architecture and Library for Creation and Monitoring of WS-Agreements. *In:* Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC 2004). *ACM Press, 2004.*

[Leu02] LEUTNER, DETLEV*:* Adaptivität und Adaptierbarkeit multimedialer Lehr- und Informationssysteme. *In:* Information und Lernen mit Multimedia und Internet*, Seiten 115–125. Beltz, 2002.*

[Ley97] LEYMANN, FRANK*:* Transaktionsunterstützung für Workflows. *Informatik - Forschung und Entwicklung, 12:82–90, 1997.*

[Ley03] LEYMANN, FRANK*:* Choreographie: Geschäftsprozesse mit Web-Services. *ObjektSpektrum, 2003(6):13–15, 2003.*

[LGJ+98] LEE, JINTAE, MICHAEL GRUNINGER, YAN JIN, THOMAS MALONE, AUSTIN TATE *und* GREGG YOST*:* The PIF Process Interchange Format and Framework Version 1.2. *Technischer Bericht, PIF Working Group, 1998.*

[LH99] LUDWIG, HEIKO *und* YIGAL HOFFNER*:* Contract-based Cross-Organisational Workflows - The CrossFlow Project. *In:* Cross-Organisational Workflow Management and Co-ordination*, Seiten 1–29. CEUR-WS.org, 1999.*

[Lie07] LIEBHART, DANIEL*:* SOA goes real: Service-orientierte Architekturen erfolgreich planen und einführen. *Hanser, München, 2007.*

[LLF+09] LIN, CUI, SHIYONG LU, XUBO FEI, ARTEM CHEBOTKO, DARSHAN PAI, ZHAOQIANG LAI, FARSHAD FOTOUHI *und* JING HUA*:* A Reference Architecture for Scientific Workflow Management Systems and the VIEW SOA Solution. *IEEE Transactions on Services Computing, 2:79–92, 2009.*

[LNP02] LOTSPIECH, JEFFREY, STEFAN NUSSER *und* FLORIAN PESTONI*:* Broadcast Encryption's Bright Future. *Computer, 35(8):57–*

63, 2002.

[LP06] LINHOFF-POPIEN, CLAUDIA*:* Ubiquitous Copmuting – Machbarkeit und Grenzen. *In:* Umhegt oder abhängig? Der Mensch in einer digitalen Umgebung*, Seiten 35 – 48. Springer, 2006.*

[LR00] LEYMANN, FRANK *und* DIETER ROLLER*:* Production Workflow - Concepts and Techniques*. PTR Prentice Hall, Upper Saddle River, NJ, USA, 2000.*

[LSPF07] LEE, KEVIN*,* RIZOS SAKELLARIOU*,* NORMAN W. PATON *und* ALVARO A. A. FERNANDES*:* Workflow Adaptation as an Autonomic Computing Problem*. In:* Proceedings of the 2nd Workshop on Workflows in Support of Large-scale Science (WORKS 2007)*, Seiten 29–34. ACM, 2007.*

[Luc02] LUCKHAM, DAVID*:* The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Professional, Boston, MA, USA, 3. Auflage, 2002.*

[LV06] LODGE, JULIET *und* R. VERMER*:* Case Study eErasmus eHigher Education (eEH)*. Technischer Bericht, 6th Framework Programme, Information Society Technologies, R4eGov, Deliverable WP3 D1-D4, 2006.*

[Mar10] MARTIN, WOLFGANG*:* SOA Check 2010 - Status Quo und Trends im Vergleich zum SOA Check 2007 bis 2009*. Zusammenfassung der Studie SOA Check 2010, Wolfgang Martin Team S.A.R.L. Martin, Annecy Fachgebiet Multimedia Kommunikation, Technische Universität Darmstadt SOA Competence Center im httc e.V., Darmstadt, 2010.*

[Mas07] MASAK, DIETER*:* SOA: Serviceorientierung in Business und Software*. Springer, Berlin, Heidelberg, 2007.*

[Mau09] MAUTE, CHRISTOPH*:* Zur Rolle und Nutzen von Key Performance Indicators*. GRIN Verlag, München, 2009.*

[May04a] MAYER, ANDREAS*:* Workflowsysteme als Basis für E-Government-Anwendungen. *In:* Das Reformkonzept E-Government: Potenziale – Ansätze – Erfahrungen*, Seiten 110–121. LIT Verlag, 2004.*

[May04b] MAYRHOFER, R.*:* An Architecture for Context Prediction*. Doktorarbeit, Johannes Kepler University Linz, 2004.*

[May05] MAYRHOFER, R.*:* Context Prediction based on Context Histories: Expected Benefits, Issues and Current State-of-the-Art*. In:* Proceedings of 1st International Workshop on Exploiting Context Histories in Smart Environments (ECHISE 2005)*, Seiten 31–36. Fraunhofer, 2005.*

[MBCP05] MODAFFERI, STEFANO*,* BOUALEM BENATALLAH*,* FABIO CASATI *und* BARBARA PERNICI*:* A Methodology for Designing and Managing Context-Aware Workflows*. In:* Mobile Information Systems II*, Seiten 91–106, Boston, USA, 2005. Springer.*

[MBM09] MENDES, MARCELO*,* PEDRO BIZARRO *und* PAULO MARQUES*:* A Performance Study of Event Processing Systems*. In:* Performance Evaluation and Benchmarking*, Seiten 221–236. Springer, 2009.*

[MCA+06] MECELLA, MASSIMO*,* TIZIANA CATARCI*,* MICHELE ANGELACCIO*,* BERTA BUTTARAZZI*,* ALENKA KREK*,* SCHAHRAM DUSTDAR *und* GUIDO VETERE*:* WORKPAD: an Adaptive Peer-to-Peer Software Infrastructure for Supporting Collaborative Work of Human Operators in Emergency/Disaster Scenarios*. In:* Proceedings of International Symposium on Collaborative Technologies and Systems (CTS 2006)*, Seiten 173–180. IEEE Computer Society, 2006.*

[MDP+00] MILOJÍČIĆ, DEJAN S.*,* FRED DOUGLIS*,* YVES PAINDAVEINE*,* RICHARD WHEELER *und* SONGNIAN ZHOU*:* Process migration*. ACM Computing Surveys, 32:241–299, 2000.*

[Mei09] MEINERS, MATTHIAS*:* Kontextdatenprognose auf mobilen Geräten*. Diplomarbeit, Universität Hamburg, Fachbereich Informatik, Verteilte Systeme und Informationssysteme, 2009.*

[Mel10] MELZER, INGO*:* Service-orientierte Architekturen mit Web Services: Konzepte – Standards – Praxis*. Spektrum-Verlag, Heidelberg, 4. Auflage, 2010.*

[Mer95] MERTENS, PETER*:* Integrierte Informationsverarbeitung 1. Administrations- und Dispositionssysteme in der Industrie*. Gabler, Wiesbaden, 10. Auflage, 1995.*

[MGR04] MÜLLER, ROBERT*,* ULRIKE GREINER *und* ERHARD RAHM*:*

AGENT WORK: A Workflow System Supporting Rule-based Workflow Adaptation. *Data and Knowledge Engineering, 51(2):223–256, 2004.*

[Müh04] MÜHLBAUER, BERND H.*:* Prozessorganisation im DRG-geführten Krankenhaus. *Wiley-VCH, Weinheim, 2004.*

[Mic95] MICROSOFT*:* The Component Object Model Specification. *Technischer Bericht, Microsoft, 1995.*

[Mil99] MILNER, ROBIN*:* Communicating and Mobile Systems: The π-Calculus. *Cambridge University Press, New York, NY, USA, 1999.*

[MK09] MODI, VIPUL *und* DEVON KEMP*:* Web Services Dynamic Discovery (WS-Discovery) Version 1.1. *Technischer Bericht, Organization for the Advancement of Structured Information Standards (OASIS), 2009.*

[Mül05] MÜLLER, JOACHIM*:* Workflow-based Integration: Grundlagen, Technologien, Management. *Springer-Verlag, Berlin, Heidelberg, 2005.*

[MLM+06] MACKENZIE, C. MATTHEW, KEN LASKEY, FRANCIS MCCABE, PETER F. BROWN *und* REBEKAH METZ*:* Reference Model for Service Oriented Architecture 1.0. *Technischer Bericht, Organization for the Advancement of Structured Information Standards (OASIS), 2006.*

[MLZ06] MENDLING, J., K. BISGAARD LASSEN *und* UWE ZDUN*:* Transformation Strategies between Block-Oriented and Graph-Oriented Process Modelling Languages. *International Journal of Business Process Integration and Management (IJBPIM), 3(2):297–312, 2006.*

[MM05a] MAURINO, ANDREA *und* STEFANO MODAFFERI*:* Partitioning Rules for Orchestrating Mobile Information Systems. *Personal and Ubiquitous Computing, 9:291–300, 2005.*

[MM05b] MAURINO, ANDREA *und* STEFANO MODAFFERI*:* Workflow Management in Mobile Environments. *In:* Ubiquitous Mobile Information and Collaboration Systems*, Seiten 83–95. Springer, 2005.*

[MM05c] MONTAGUT, FREDERIC *und* REFIK MOLVA*:* Enabling Pervasi-

ve Execution of Workflows. *In:* Collaborative Computing: Networking, Applications and Worksharing, *Seiten 10–20. IEEE Computer Society, 2005.*

[MM06] MONTAGUT, FREDERIC *und* REFIK MOLVA*:* Augmenting Web Services Composition with Transactional Requirements. *In:* IEEE International Conference on Web Services, *Seiten 91–98. IEEE Computer Society, 2006.*

[MM07] MARZOLLA, MORENO *und* RAFFAELA MIRANDOLA*:* Performance Prediction of Web Service Workflows. *In:* Software Architectures, Components, and Applications, *Band 4880, Seiten 127–144. Springer, 2007.*

[MM08] MONTAGUT, FREDERIC *und* REFIK MOLVA*:* Bridging Security and Fault Management within Distributed Workflow Management Systems. *IEEE Transactions on Service Computing, 1:33–48, 2008.*

[MNS+05] MANOEL, EDSON, MORTEN JUL NIELSEN, ABDI SALAHSHOUR, SAI SAMPATH K.V.L. *und* SANJEEV SUDARSHANAN*:* Problem Determination Using Self-Managing Autonomic Technology. *IBM Redbooks, 2005.*

[Mor07] MORRIS, JOHN*:* Practical Data Migration. *British Computer Society, Chippenham, 2007.*

[MPS02] MORI, GIULIO, FABIO PATERNO *und* CARMEN SANTORO*:* CTTE: Support for Developing and Analyzing Task Models for Interactive System Design. *IEEE Transactions on Software Engineering, 28:797–813, 2002.*

[MPS04] MORI, GIULIO, FABIO PATERNO *und* CARMEN SANTORO*:* Design and Development of Multidevice User Interfaces through Multiple Logical Descriptions. *IEEE Transactions on Software Engineering, 30:507–520, 2004.*

[MR09] MARINO, JIM *und* MICHAEL ROWLEY*:* Understanding SCA (Service Component Architecture). *Addison-Wesley Professional, Boston, MA, USA, 2009.*

[MRD08] MOSER, OLIVER, FLORIAN ROSENBERG *und* SCHAHRAM DUSTDAR*:* Non-intrusive Monitoring and Service Adaptation for WS-BPEL. *In:* Proceedings of the 17th International Confe-

rence on World Wide Web (WWW 2008)*, Seiten 815–824. ACM, 2008.*

[MT88] MALONE, THOMAS W. *und* MASSACHUSETTS INSTITUTE OF TECHNOLOGY*.:* What is Coordination Theory? *Technischer Bericht, Massachusetts Institute of Technology (MIT), Sloan School of Management, 1988.*

[Mur04] MURCH, RICHARD*:* Autonomic Computing*. IBM Press, 2004.*

[MWL08] MARTIN, DANIEL*,* DANIEL WUTKE *und* FRANK LEYMANN*:* A Novel Approach to Decentralized Workflow Enactment*. In:* Enterprise Distributed Object Computing*, Seiten 127–136. IEEE Computer Society, 2008.*

[MWW+98] MUTH, PETER*,* DIRK WODTKE*,* JEANINE WEISSENFELS*,* ANGELIKA KOTZ DITTRICH *und* GERHARD WEIKUM*:* From Centralized Workflow Specification to Distributed Workflow Execution*. Journal of Intelligent Information Systems, 10(2):159–184, 1998.*

[MZL10] MEINERS, MATTHIAS*,* SONJA ZAPLATA *und* WINFRIED LAMERSDORF*:* Structured Context Prediction: A Generic Approach*. In:* Proceedings of the 10th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS 2010)*, Seiten 84–97. Springer, 2010.*

[NBNK05] NARENDRA, N. C.*,* UMESH BELLUR*,* S. K. NANDY *und* K. KALAPRIYA*:* Functional and Architectural Adaptation in Pervasive Computing Environments*. In:* Proceedings of the 3rd International Workshop on Middleware for Pervasive and Ad-hoc Computing (MPAC 2005)*, Seiten 1–7. ACM, 2005.*

[NC04] NEIGER, DINA *und* LEONID CHURILOV*:* Goal-Oriented Business Process Modeling with EPCs and Value-Focused Thinking*. In:* Business Process Management*, Seiten 98–115. Springer, 2004.*

[NG06] NARENDRA, N. C. *und* SRINIVAS GUNDUGOLA*:* Automated Context-Aware Adaptation of Web Service Executions*. In:* Proceedings of the IEEE International Conference on Computer Systems and Applications (AICCSA 2006)*, Seiten 179–187. IEEE Computer Society, 2006.*

[NM02] NORIN, ROBERTA *und* MIKE MARIN*:* Workflow Process Defini-

tion Interface – XML Process Definition Language. *Technischer Bericht, Workflow Management Coalition (WfMC), 2002.*

[NMS05] NORIN, ROBERTA, MIKE MARIN *und* ROBERT SHAPIRO*:* Workflow Process Definition Interface - XML Process Definition Language Version 2.0. *Technischer Bericht, Workflow Management Coalition (WfMC), 2005.*

[NPS07] NISSEN, VOLKER, MATTHIAS PETSCH *und* HAGEN SCHORCHT*:* Service-orientierte Architekturen: Chancen und Herausforderungen bei der Flexibilisierung und Integration von Unternehmensprozessen. *Gabler-Verlag, Wiesbaden, 2007.*

[Nur08] NURCAN, SELMIN*:* A Survey on the Flexibility Requirements Related to Business Processes and Modeling Artifacts. *In:* Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)*, Seiten 378–388. IEEE Computer Society, 2008.*

[OAS04] OASIS*:* Asynchronous Service Access Protocol (ASAP) Version 1.0 (Working Draft G). *Technischer Bericht, Organization for the Advancement of Structured Information Standards (OASIS), 2004.*

[OAS05] OASIS*:* Asynchronous Service Access Protocol (ASAP) Version 1.0 (Proposed Committee Draft). *Technischer Bericht, Organization for the Advancement of Structured Information Standards (OASIS), 2005.*

[OAS07] OASIS*:* Web Services Business Process Execution Language Version 2.0. *Technischer Bericht, Organization for the Advancement of Structured Information Standards (OASIS), 2007.*

[ODvdA+09] OUYANG, CHUN, MARLON DUMAS, WIL VAN DER AALST, ARTHUR H. M. TER HOFSTEDE *und* JAN MENDLING*:* From business process models to process-oriented software systems. *ACM Transactions on Software Engineering and Methodology, 19(1):1–37, 2009.*

[OE03] O'SULLIVAN, JUSTIN *und* DAVID EDMOND*:* When and Where is a service? Investigating Temporal and Locative Service Properties. *In:* Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT 2003)*, Seiten 90–94. IEEE Computer Society, 2003.*

[OEH02] O'SULLIVAN, JUSTIN, DAVID EDMOND *und* ARTHUR TER HOFSTEDE*:* What's in a Service? Towards accurate description of non-functional service properties. *Distributed and Parallel Databases, 12(2-3), 2002.*

[Oh06] OH, SANGYOON*:* Web Service Architecture for Mobile Computing. *Doktorarbeit, Indiana University, Indianapolis, USA, 2006.*

[OMG08] OMG*:* Common Object Request Broker Architecture (CORBA) Specification, Version 3.1, Part I-III. *Technischer Bericht, Object Management Group (OMG), 2008.*

[OMG11] OMG*:* Business Process Model and Notation (BPMN), Version 2.0. *Technischer Bericht, Object Management Group (OMG), 2011.*

[Pan08] PAN, MICHAEL J.*:* Disconnected Operation in Scientific Workflow Management Systems. *Technischer Bericht, University of California, Los Angeles, 2008.*

[Pap03] PAPAZOGLOU, MICHAEL P.*:* Service-Oriented Computing: Concepts, Characteristics and Directions. *In:* Proceedings of the 4th International Conference on Web Information Systems (WISE 2003)*, Seiten 3–12. IEEE Computer Society, 2003.*

[Pap08] PAPAZOGLOU, MICHAEL P.*:* Web Services: Principles and Technology. *Pearson, Prentice Hall, Upper Saddle River, NJ, USA, 2008.*

[Par07] PARASHAR, MANISH*:* Autonomic Computing: Concepts, Infrastructure, and Applications. *Taylor & Francis, Inc., Bristol, PA, USA, 2007.*

[Pat00] PATERNO, FABIO*:* Model-based Design and Evaluation of Interactive Applications. *Springer-Verlag, Berlin, Heidelberg, 2000.*

[PC07] PAJUNEN, LASSE *und* SURESH CHANDE*:* Developing Workflow Engine for Mobile Devices. *In:* Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007)*, Seite 279. IEEE Computer Society, 2007.*

[PCP07] PASPALLIS, NEARCHOS, AVRAAM CHIMARIS *und* GEORGE A. PAPADOPOULOS*:* Experiences from Developing a Distributed Context Management System for Enabling Adaptivity. *In:*

Proceedings of the 7th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems DAIS 2007)*, Seiten 225–238. Springer, 2007.*

[Pel03] PELTZ, CHRIS*:* Web Services Orchestration and Choreography. *Computer, 36(10):46–52, 2003.*

[Pet05] PETZOLD, JAN*:* Zustandsprädiktoren zur Kontextvorhersage in ubiquitären Systemen. *Doktorarbeit, University of Augsburg, 2005.*

[Pib01] PIBERNIK, RICHARD*:* Flexibilitätsplanung in Wertschöpfungsnetzwerken. *Deutscher Universitäts-Verlag, Wiesbaden, 2001.*

[Pri03] PRIOR, CAROL*:* Workflow and Process Management. *Workflow Handbook 2003, 2003.*

[PRR08] PLOESSER, KARSTEN*,* JAN RECKER *und* MICHAEL ROSEMANN*:* Towards a Classification and Lifecycle of Business Process Change. *In:* 9th Workshop on Business Process Modeling, Development, and Support (BPMDS 2008)*. Tapir Academic Press, 2008.*

[PTDL07] PAPAZOGLOU, MICHAEL P.*,* PAOLO TRAVERSO*,* SCHAHRAM DUSTDAR *und* FRANK LEYMANN*:* Service-Oriented Computing: State of the Art and Research Challenges. *Computer, 40:38–45, 2007.*

[Puh06] PUHLMANN, FRANK*:* Why do we actually need the Pi-Calculus for Business Process Management. *In:* 9th International Conference on Business Information Systems (BIS 2006)*, Seiten 77–89. Gesellschaft für Informatik, 2006.*

[Puh07] PUHLMANN, FRANK*:* Soundness verification of business processes specified in the Pi-calculus. *In:* Proceedings of the 2007 OTM Confederated International Conference on On The Move to Meaningful Internet Systems (OTM 2007)*, Seiten 6–23, Berlin, Heidelberg, 2007. Springer.*

[PvdH05] PAPAZOGLOU, M. P. *und* W. J. VAN DEN HEUVEL*:* Web Services Management: A Survey. *IEEE Internet Computing, 9(6):58–64, 2005.*

[PvdH07] PAPAZOGLOU, MICHAEL P. *und* WILLEM-JAN VAN DEN HEU-

VEL: Service Oriented Architectures: Approaches, Technologies and Research Issues. *The International Journal on Very Large Data Bases (VLDB), 16(3):389–415, 2007.*

[PY02] PAPAZOGLOU, MIKE P. *und* JIAN YANG: Design Methodology for Web Services and Business Processes. *In:* Proceedings of the International Workshop on Technologies for E-Services*, Seiten 54–64. Springer, 2002.*

[RB07] REICHERT, MANFRED *und* THOMAS BAUER: Supporting ad-hoc changes in distributed workflow management systems. *In:* Proceedings of the 2007 OTM Confederated International Conference on On the Move to Meaningful Internet Systems: CoopIS, DOA, ODBASE, GADA, and IS - Volume Part I*, Seiten 150–168. Springer, 2007.*

[RD98] REICHERT, MANFRED *und* PETER DADAM: ADEPT flex - Supporting Dynamic Changes of Workflows Without Loosing Control. *Journal of Intelligent Information Systems, 10:93–129, 1998.*

[RD05] ROSENBERG, FLORIAN *und* SCHAHRAM DUSTDAR: Business Rules Integration in BPEL: A Service-Oriented Approach. *In:* IEEE International Conference on E-Commerce Technology (CEC 2005)*, Seiten 476–479. IEEE Computer Society, 2005.*

[Rei07] REIJERS, HAJO A.: Case Prediction in BPM Systems: A Research Challenge. *Journal of the Korean Institute of Industrial Engineers, 33:1–10, 2007.*

[Rei10] REISIG, WOLFANG: Petrinetze: Modellierungstechnik, Analysemethoden, Fallstudien. *Leitfäden der Informatik. Vieweg+Teubner, Wiesbaden, 2010.*

[RG95] RAO, ANAND S. *und* MICHAEL P. GEORGEFF: BDI Agents: From Theory to Practice. *In:* Proceedings of the 1st International Conference on Multiagent Systems (ICMAS 1995)*, Seiten 312–319. AAAI Press, 1995.*

[RHE04] RUSSELL, NICK, ARTHUR H. M. TER HOFSTEDE *und* DAVID EDMOND: Workflow Data Patterns. *Technischer Bericht, Queensland University of Technology, Eindhoven University of Technology, 2004.*

[RHE05] RUSSELL, NICK, ARTHUR H. M. TER HOFSTEDE *und* DAVID

EDMOND*:* Workflow Resource Patterns: Identification, Representation and Tool Support. *In:* Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE 2005)*, Seiten 216–232. Springer, 2005.*

[RHM06] RUSSELL, NICK*,* ARTHUR H. M. TER HOFSTEDE *und* NATALIYA MULYAR*:* Workflow ControlFlow Patterns: A Revised View. *Technischer Bericht, Queensland University of Technology, Eindhoven University of Technology, 2006.*

[Rie03] RIEDL, RENÉ*:* Begriffliche Grundlagen des Business Process Outsourcing. *Information Management and Consulting, 18(3):6–10, 2003.*

[Rie08] RIEF, ALEXANDER*:* Entwicklungsorientierte Steuerung strategischer Unternehmensnetzwerke. *Gabler Verlag, Wiesbaden, 2008.*

[RM06] RECKER, JAN *und* JAN MENDLING*:* On the Translation between BPMN and BPEL: Conceptual Mismatch between Process Modeling Languages. *In:* Proceedings of the 18th International Conference on Advanced Information Systems Engineering. Proceedings of Workshops and Doctoral Consortiums*, Seiten 521–532. Namur University Press, 2006.*

[Rom08] ROMMELSPACHER, JONAS*:* Ereignisgetriebene Architekturen. *Wirtschaftsinformatik, 50(4):314–317, 2008.*

[Ros03] ROSS, RONALD G.*:* Principles of the Business Rule Approach. *Addison-Wesley, Boston, MA, USA, 2003.*

[RR02] RAMANATHAN, RAM *und* JASON REDI*:* A Brief Overview of Adhoc Networks: Challenges and Directions. *IEEE Communications Magazine, 40(5):20–22, 2002.*

[RR06] ROSEMANN, MICHAEL *und* JAN RECKER*:* Context-aware Process Design: Exploring the Extrinsic Drivers for Process Flexibility. *In:* 18th International Conference on Advanced Information Systems Engineering. Proceedings of Workshops and Doctoral Consortium.*, Seiten 149–158. Namur University Press, 2006.*

[RRD03] REICHERT, MANFRED*,* STEFANIE RINDERLE *und* PETER DADAM*:* ADEPT Workflow Management System: Flexible Support for Enterprise-Wide Business Processes. *In:* Proceedings of the

1st International Conference on Business Process Management (BPM 2003)*, Seiten 371–379. Springer, 2003.*

[RRD04] RINDERLE, STEFANIE*,* MANFRED REICHERT *und* PETER DADAM*:* Correctness Criteria for Dynamic Changes in Workflow Systems: A Survey. *Data and Knowledge Engineering, 50(1):9–34, 2004.*

[RRFA06] ROSEMANN, MICHAEL*,* JAN RECKER*,* CHRISTIAN FLENDER *und* PETER ANSELL*:* Understanding Context-Awareness in Business Process Design. *In:* 17th Australasian Conference on Information Systems. *Australasian Association for Information Systems, 2006.*

[RS04] REICHERT, MANFRED *und* DIETMAR STOLL*:* Komposition, Choreographie und Orchestrierung von Web Services. *EMISA Forum, 24(2):21–32, 2004.*

[RSS06a] REGEV, GIL*,* PNINA SOFFER *und* RAINER SCHMIDT*:* Taxonomy of Flexibility in Business Processes. *In:* 7th Workshop on Business Process Modeling, Development, and Support: Requirements for Flexibility and the Ways to Achieve It (BPMDS 2006)*, Seiten 90–93. Presses Universitaires de Namur, 2006.*

[RSS06b] ROTH, HEINZ*,* JOSEF SCHIEFER *und* ALEXANDER SCHATTEN*:* Probing and Monitoring of WSBPEL Processes with Web Services. *In:* Proceedings of the 8th IEEE International Conference on E-Commerce Technology and 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC-EEE 2006)*, Seiten 30–39. IEEE Computer Society, 2006.*

[RT06] ROTH, ROMAN *und* STEFAN TILKOV*:* Ereignis-getriebene Architekturen: Ein Überblick. *OBJEKTspektrum, 13(2):26–27, 2006.*

[RtHEvdA05] RUSSELL, NICK*,* ARTHUR TER HOFSTEDE*,* DAVID EDMOND *und* WIL VAN DER AALST*:* Workflow Data Patterns: Identification, Representation and Tool Support. *In:* 24th International Conference on Conceptual Modeling (ER 2005)*, Seiten 353–368. Springer, 2005.*

[RvdAA06] RUSSELL, NICK*,* WIL VAN DER AALST *und* ARTHUR*:* Exception Handling Patterns in Process-Aware Information Systems. *Technischer Bericht, BPMcenter.org, 2006.*

[RvL11] RADEMAKERS, TIJS *und* RON VAN LIEMPD*:* Activiti in Action – Executable business processes in BPMN 2.0*. Manning Publications Co., Greenwich, 2011.*

[RWK+08] REICHLE, ROLAND*,* MICHAEL WAGNER*,* MOHAMMAD ULLAH KHAN*,* KURT GEIHS*,* MASSIMO VALLA*,* CRISTINA FRA*,* NEARCHOS PASPALLIS *und* GEORGE A. PAPADOPOULOS*:* A Context Query Language for Pervasive Computing Environments*. In:* Proceedings of the 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications*, Seiten 434–440. IEEE Computer Society, 2008.*

[Sat96] SATYANARAYANAN, MAHADEV*:* Fundamental Challenges in Mobile Computing*. In:* Proceedings of the 15 ACM Symposium on Principles of Distributed Computing *. ACM, 1996.*

[SCB06] SCHMITZ, P.*,* T.V. CANGH *und* A. BOUJRAF*:* 6th Framework Programme, Information Society Technologies, R4eGov, Deliverable WP3-D2, Case Studie Eurojust/Europol Collaboration*, 2006.*

[Sch00] SCHÄTZLE, ROLAND*:* Workflow-Management - ein ereignisbasierter Ansatz*. Doktorarbeit, Universität Karlsruhe (TH), Fakultät für Wirtschaftswissenschaften, 2000.*

[Sch01] SCHAMBURGER, RALF*:* Integrierte Betrachtung von Anwendungen und Systemen zur verteilten Workflow-Bearbeitung*. Doktorarbeit, Friedrich-Alexander-Universität Erlangen-Nürnberg, Institut für Informatik, 2001.*

[Sch04a] SCHMIDT, RAINER*:* Enactment of Inter-Organizational Workflows Using Aspect-Element-Oriented Web Services*. In:* Proceedings of the 15th International Workshop on Database and Expert Systems Applications*, Seiten 254–258. IEEE Computer Society, 2004.*

[Sch04b] SCHULER, CHRISTOPH*:* Verteiltes Peer-to-Peer-Prozessmanagement: Die Realisierung einer Hyperdatenbank*. Doktorarbeit, Eidgenössische Technische Hochschule Zürich, 2004.*

[Sch05] SCHMIDT, RAINER*:* Flexible Support of Inter-Organizational Business Processes Using Web Services*. In:* 6th Workshop on Business Process Modeling, Development, and Support

(BPMDS'05)*, Seiten 1–8. Systemic Modeling Laboratory LAMS, 2005.*

[Sch07a] SCHMEH, KLAUS*:* Kryptografie: Verfahren, Protokolle, Infrastrukturen*. Dpunkt-Verlag, Heidelberg, 2007.*

[Sch07b] SCHMELZLE, OLEG*:* Transformation von annotierten Geschäftsprozessen nach BPEL*. Diplomarbeit, Leibniz Universität Hannover, Fakultät für Elektrotechnik und Informatik, 2007.*

[SD03] SHI, D. *und* R. L. DANIELS*:* A Survey of Manufacturing Flexibility: Implications for E-Business Flexibility*. IBM Systems Journal, 42(3):414–427, 2003.*

[Ses10] SESHAN, PARAMESWARAN*:* Process-Centric Architecture For Enterprise Software Systems*. Auerbach Publications, Boca Raton, 2010.*

[SG06] SCHACHER, MARKUS *und* PATRICK GRÄSSLE*:* Agile Unternehmen durch Business Rules: Der Business Rules Ansatz*. Springer, Berlin, Heidelberg, New York, 2006.*

[SGS00] SPIRTES, PETER*,* CLARK GLYMOUR *und* RICHARD SCHEINES*:* Causation, Prediction, and Search*. MIT Press, Cambridge, MA, USA, 2000.*

[SGT+04] SCHLENOFF, CRAIG*,* MICHAEL GRUNINGER*,* FLORENCE TISSOT*,* JOHN VALOIS*,* JOSH LUBELL *und* JINTAE LEE*:* The Process Specification Language (PSL) Overview and Version 1.0 Specification*. Technischer Bericht, International Organization for Standardization (ISO), 2004.*

[Sha08] SHAPIRO, ROBERT M.*:* XPDL 2.1 - Integrating Process Interchange and BPMN*. Technischer Bericht, Workflow Management Coalition (WfMC), 2008.*

[SHD07] SIGG, STEPHAN*,* SANDRA HASELOFF *und* KLAUS DAVID*:* Prediction of Context Time Series*. In:* IKONEN, J.*,* M. JUUTILAINEN *und* J. PORRAS *(Herausgeber):* Proceedings of the 5th Workshop on Applications of Wireless Communications (WAWC'07)*, 2007.*

[SHH+07] SEN, ROHAN*,* GREGORY HACKMANN*,* MART HAITJEMA*,* GRUIA-CATALIN ROMAN *und* CHRISTOPHER GILL*:* Coordina-

ting Workflow Allocation and Execution in Mobile Environments. *In:* Coordination Models and Languages*, Seiten 249–267. Springer, 2007.*

[Sig08] SIGG, STEPHAN*:* Development of a Novel Context Prediction Algorithm and Analysis of Context Prediction Schemes. *Doktorarbeit, University of Kassel, 2008.*

[SKH03] SCHEER, AUGUST-WILHELM*,* HELMUT KRUPPKE *und* RALF HEIB*:* E-government: Prozessoptimierung in der öffentlichen Verwaltung. *Springer, Berlin, Heidelberg, 2003.*

[SKP09] SYMONDS, JUDITH *und* MEHDI KHOSROW-POUR*:* Ubiquitous and Pervasive Computing: Concepts, Methodologies, Tools, and Applications. *Information Science Reference, IGI Publishing, Hershey, PA, USA, 2009.*

[SLI08] SMANCHAT, SUCHA*,* SEA LING *und* MARIA INDRAWAN*:* A Survey on Context-aware Workflow Adaptations. *In:* Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia (MoMM 2008)*, Seiten 414–417. ACM, 2008.*

[SMOW02] SAHAI, AKHIL*,* VIJAY MACHIRAJU*,* JINSONG OUYANG *und* KLAUS WURSTER*:* Message Tracking in SOAP-based Web Services. *In:* Network Operations and Management Symposium (NOMS 2002)*, Seiten 33–47, 2002.*

[SMR+08] SCHONENBERG, HELEN*,* RONNY MANS*,* NICK RUSSELL*,* NATALIYA MULYAR *und* WIL VAN DER AALST*:* Process Flexibility: A Survey of Contemporary Approaches. *In:* Proceedings of the 4th International Workshop CIAO! and 4th International Workshop EOMAS*, Seiten 16–30. Springer, 2008.*

[SN07] SAIDANI, OUMAIMA *und* SELMIN NURCAN*:* Towards Context Aware Business Process Modelling. *In:* Proceedings of the 8th Workshop on Business Process Modelling, Development and Support (BPMDS 2007)*. Tapir Academic Press, 2007.*

[SO99] SADIQ, SHAZIA *und* MARIA ORLOWSKA*:* Architectural Considerations in Systems Supporting Dynamic Workflow Modification. *In:* Proceedings of the Workshop on Software Architectures for Business Process Management at the 11th Conference on Advanced Information Systems Engineering (CaiSE 1999)*. Springer, 1999.*

[SO04] SCHULZ, KARSTEN A. *und* MARIA E. ORLOWSKA*:* Facilitating Cross-organisational Workflows with a Workflow View Approach*. Data and Knowledge Engineering, 51(1):109–147, 2004.*

[SOB+03] SONDHEIMER, NORMAN K.*,* LEON P. OSTERWEIL*,* MATTHEW P. BILLMERS*,* JOEL T. SIEH *und* BRUCE B. SOUTHARD*:* e-Government through Process Modeling: A Requirements Field Study*. In:* International Conference on e-Society 2003*. IADIS, 2003.*

[Sof05] SOFFER, PNINA*:* On the Notion of Flexibility in Business Processes*. In:* Proceedings of the CAiSE 2005 Workshops*, Seiten 35–42. FEUP, 2005.*

[Sos11] SOSINSKY, BARRIE*:* Cloud Computing Bible*. Wiley Publishing, Indianapolis, 2011.*

[SPG04] SWENSON, KEITH D.*,* SAMEER PRADHAN *und* MIKE D. GILGER*:* Wf-XML 2.0: XML Based Protocol for Runtime Integration of Process Engines*. Technischer Bericht, Workflow Management Coalition (WfMC), 2004.*

[SRF06] SEN, ROHAN*,* GRUIA-CATALIN ROMAN *und* ANDREW FRANK*:* CiAN: A Language and Middleware for Collaboration in Ad hoc Networks*. Technischer Bericht, Washington University, 2006.*

[SRG08] SEN, ROHAN*,* GRUIA-CATALIN ROMAN *und* CHRISTOPHER D. GILL*:* CiAN: A Workflow Engine for MANETs*. In:* COORDINATION 2008*, Seiten 280–295. Springer, 2008.*

[SSS+04] SCHEK, HANS-JÖRG*,* HEIKO SCHULDT*,* CHRISTOPH SCHULER*,* CAN TÜRKER *und* ROGER WEBER*:* Hyperdatenbanken zur Verwaltung von Informationsräumen*. it - Information Technology, 46(2):67–75, 2004.*

[Öst95] ÖSTERLE, HUBERT*:* Business Engineering: Prozess- und Systementwicklung. Band 1: Entwurfstechniken*. Springer, Berlin, Heidelberg, 1995.*

[Sta06] STAUD, JOSEF L.*:* Geschäftsprozessanalyse: Ereignisgesteuerte Prozessketten und objektorientierte Geschäftsprozessmodellierung für Betriebswirtschaftliche Standardsoftware*. Springer, Berlin, Heidelberg, 2006.*

[Ste07] STELAND, ANSGAR*:* Basiswissen Statistik - Kompaktkurs für

Anwender aus Wirtschaft, Informatik und Technik. *Springer, Berlin, Heidelberg, 2007.*

[Str08] STROHMEIER, STEFAN*:* Business Process Management-Systeme. *In:* Informationssysteme im Personalmanagement*, Seiten 305–315. Vieweg+Teubner, 2008.*

[Str09] STRASSENBURG, DANIEL*:* Automatisches Management dienstbasierter verteilter Geschäftsprozesse. *Diplomarbeit, Universität Hamburg, Fachbereich Informatik, Verteilte Systeme und Informationssysteme, 2009.*

[Swe99] SWENSON, KEITH*:* Simple Workflow Access Protocol (SWAP). *Technischer Bericht, SWAP Working Group, 1999.*

[SWG+07] SNOWDON, ROBERT A.*,* BRIAN WARBOYS*,* R. MARK GREENWOOD*,* CHRISTOPHER P. HOLLAND*,* P. J. KAWALEK *und* DUNCAN R. SHAW*:* On the Architecture and Form of Flexible Process Support. *Software Process: Improvement and Practice, 12(1):21–34, 2007.*

[SWSS03] SCHULER, CHRISTOPH*,* ROGER WEBER*,* HEIKO SCHULDT *und* HANS-J. SCHEK*:* Peer-to-Peer Process Execution with OSIRIS. *In:* Proceedings of the 1st International Conference on Service-Oriented Computing (ICSOC 2003)*, Seiten 483–498. Springer, 2003.*

[SWSS04] SCHULER, CHRISTOPH*,* ROGER WEBER*,* HEIKO SCHULDT *und* HANS-JÖRG SCHEK*:* Scalable Peer-to-Peer Process Management - The OSIRIS Approach. *In:* IEEE International Conference on Web Services (ICWS 2004)*, Seiten 26–34. IEEE Computer Society, 2004.*

[SWT05] SIMON, RAINER*,* FLORIAN WEGSCHEIDER *und* KONRAD TOLAR*:* Tool-supported Single Authoring for Device Independence and Multimodality. *In:* Proceedings of the 7th Conference on Human-Computer Interaction with Mobile Devices and Services (Mobile HCI 2005)*, Seiten 91–98. ACM, 2005.*

[SY06] STROHMAIER, MARKUS *und* ERIC YU*:* Towards autonomic workflow management systems. *In:* Proceedings of the 2006 conference of the Center for Advanced Studies on Collaborative Research (CASCON 2006)*. ACM, 2006.*

[Szy02] SZYPERSKI, CLEMENS*:* Component Software: Beyond Object-

Oriented Programming. *Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.*

[Tan03] TANENBAUM, ANDREW S.*:* Moderne Betriebssysteme. *Pearson Studium, München, 3. Auflage, 2003.*

[TBB03] TURNER, MARK, DAVID BUDGEN *und* PEARL BRERETON*:* Turning Software into a Service. *Computer, 36(10):38–44, 2003.*

[Tha01] THATTE, SATISH*:* XLANG: Web Services for Business Process Design. *Technischer Bericht, Microsoft Corporation, 2001.*

[Thu04] THURNER, VERONIKA*:* Formal fundierte Modellierung von Geschäftsprozessen. *Doktorarbeit, Technische Universität Müchen, Fakultät für Informatik, 2004.*

[Tom05] TOMA, IOAN*:* Non-functional Properties in Web Services. *Technischer Bericht, WSML Working Group, 2005.*

[TS08] TANENBAUM, ANDREW S. *und* MAARTEN VAN STEEN*:* Verteilte Systeme - Prinzipien und Paradigmen. *Pearson Studium, München, 2. Auflage, 2008.*

[Tur06] TURJALEI, MIRWAIS*:* Integration von Context-Awareness in eine Middleware für mobile Systeme. *Diplomarbeit, Universität Hamburg, Fachbereich Informatik, Verteilte Systeme und Informationssysteme, 2006.*

[TVN+04] TIAN, M., T. VOIGT, T. NAUMOWICZ, H. RITTER *und* J. SCHILLER*:* Performance Considerations for Mobile Web Services. *Elsevier Computer Communications Journal, 27:1097–1105, 2004.*

[TvS03] TANENBAUM, ANDREW S. *und* MAARTEN VAN STEEN*:* Verteilte Systeme - Grundlagen und Paradigmen. *Pearson Studium, München, 1. Auflage, 2003.*

[vA09] AMMON, RAINER VON*:* Event-Driven Business Process Management. *In:* Encyclopedia of Database Systems, *Seiten 1068–1071. Springer, 2009.*

[vAEE+09] AMMON, RAINER VON, CHRISTOPH EMMERSBERGER, THOMAS ERTLMAIER, OPHER ETZION, THOMAS PAULUS *und* FLORIAN SPRINGER*:* Existing and Future Standards for Event-Driven Business Process Management. *In:* 3rd ACM International

Conference on Distributed Event-Based Systems (DEBS 2009). *ACM, 2009.*

[vAEE+10] AMMON, RAINER VON, THOMAS ERTLMAIER, OPHER ETZION, ALEXANDER KOFMAN *und* THOMAS PAULUS*:* Integrating Complex Events for Collaborating and Dynamically Changing Business Processes. *In:* Service-Oriented Computing. ICSOC/-ServiceWave 2009 Workshops*, Seiten 370–384, 2010.*

[vAEG+08] AMMON, RAINER VON, CHRISTOPH EMMERSBERGER, TORSTEN GREINER, ADRIAN PASCHKE, FLORIAN SPRINGER *und* CHRISTIAN WOLFF*:* Event-Driven Business Process Management. *In:* Second International Conference on Distributed Event-Based Systems (DEBS 2008). *ACM, 2008.*

[VB96] VOSSEN, GOTTFRIED *und* JÖRG BECKER*:* Geschäftsprozeßmodellierung und Workflow- Management. Modelle, Methoden, Werkzeuge. *Thomson Publishing, Bonn, 1996.*

[vdA99] AALST, WIL VAN DER*:* Process-oriented Architectures for Electronic Commerce and Interorganizational Workflow. *Information Systems, 24(9):639–671, 1999.*

[vdA00] AALST, WIL VAN DER*:* Loosely Coupled Interorganizational Workflows: Modeling and Analyzing Workflows Crossing Organizational Boundaries. *Information Management, 37(2), 2000.*

[vdABHK00] AALST, WIL VAN DER, ALISTAIR P. BARROS, ARTHUR H. M. TER HOFSTEDE *und* BARTEK KIEPUSZEWSKI*:* Advanced Workflow Patterns. *In:* Proceedings of the 7th International Conference on Cooperative Information Systems (CooplS 2002)*, Seiten 18–29. Springer, 2000.*

[vdADH03] AALST, WIL VAN DER, MARLON DUMAS *und* ARTHUR H. M. TER HOFSTEDE*:* Web Service Composition Languages: Old Wine in New Bottles? *In:* Proceedings of the 29th Conference on EUROMICRO*, Seiten 298–305. IEEE Computer Society, 2003.*

[vdASS11] AALST, WIL VAN DER, M. H. SCHONENBERG *und* M. SONG*:* Time Prediction based on Process Mining. *Information Systems, 36:450–475, 2011.*

[vdATHKB03] AALST, WIL VAN DER, A. H. M. TER HOFSTEDE, B. KIEPUSZEWSKI *und* A. P. BARROS*:* Workflow Patterns. *Distributed and*

Parallel Databases, 14(1):5–51, 2003.

[vdAvH02] AALST, WIL VAN DER *und* KEES VAN HEE*:* Workflow Management: Models, Methods, and Systems*. MIT Press, Cambridge, MA, USA, 2002.*

[Ver02] VERSTEEGEN, GERHARD*:* Software-Management: Beherrschung des Lifecycles*. Springer, Berlin, 2002.*

[VGBA99] VONK, JOCHEM*,* PAUL W. P. J. GREFEN*,* ERIK BOERTJES *und* PETER M. G. APERS*:* Distributed Global Transaction Support for Workflow Management Applications*. In:* Proceedings of the 10th International Conference on Database and Expert Systems Applications (DEXA 1999)*, Seiten 942–951. Springer-Verlag, 1999.*

[VGN06] VAMBENEPE, WILLIAM*,* STEVE GRAHAM *und* PETER NIBLETT*:* Web Services Topics 1.3 (WS-Topics)*. Technischer Bericht, Organization for the Advancement of Structured Information Standards (OASIS), 2006.*

[VH03] VOGEL-HEUSER, BIRGIT*:* Systems Software Engineering*. Oldenbourg Wissenschaftsverlag, München, 2003.*

[Vil08] VILENICA, ANTE*:* Integration von Interaktionskomponenten in Prozesse für mobile Umgebungen*. Diplomarbeit, Universität Hamburg, Fachbereich Informatik, Verteilte Systeme und Informationssysteme, 2008.*

[vLLM+08] LESSEN, TAMMO VAN*,* FRANK LEYMANN*,* RALPH MIETZNER*,* JORG NITZSCHE *und* DANIEL SCHLEICHER*:* A Management Framework for WS-BPEL*. In:* European Conference on Web Services (ICWS 2008)*, Seiten 187–196. IEEE Computer Society, 2008.*

[VMSS07] VOIGT, K.I.*,* H. MONSEES*,* S. SCHORR *und* M. SAATMANN*:* Flexibilität und Adaptivität - Verständnis und Ausprägung*. In:* Neue Wege in der Automobillogistik*, Seiten 39–59. Springer, 2007.*

[vRZ07] RIEGEN, MICHAEL VON *und* SONJA ZAPLATA*:* Supervising Remote Task Execution in Collaborative Workflow Environments*. In:* Konferenzband zur KiVS 2007 für Industrie-, Kurz- und Workshopbeiträge*, Seiten 337–358. VDE Verlag, 2007.*

[VS05] VOIGT *und* SAATMANN*:* Begriffsbestimmung Flexibilität und Adaptivität. *Technischer Bericht, Universität Erlangen-Nürnberg, 2005.*

[W3C04] W3C*:* XML Schema Part 0-2. *Technischer Bericht, World Wide Web Consortium (W3C), 2004.*

[W3C07a] W3C*:* SOAP Version 1.2 Part 0-2. *Technischer Bericht, World Wide Web Consortium (W3C), 2007.*

[W3C07b] W3C*:* Web Services Description Language (WSDL) Version 2.0 Part 0-2. *Technischer Bericht, World Wide Web Consortium (W3C), 2007.*

[WDR06] WU, EUGENE, YANLEI DIAO *und* SHARIQ RIZVI*:* High-performance Complex Event Processing over Streams. *In:* Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data (SIGMOD 2006)*, Seiten 407–418. ACM, 2006.*

[Wei93] WEISER, MARK*:* Ubiquitous Computing. *IEEE Computer Hot Topics, 1993.*

[Wer07] WERNER, CHRISTIAN*:* Optimierte Protokolle für Web Services mit begrenzten Datenraten. *Doktorarbeit, Universität zu Lübeck, Institut für Telematik, 2007.*

[Wes07] WESKE, MATHIAS*:* Business Process Management: Concepts, Languages, Architectures. *Springer, Berlin, Heidelberg, 2007.*

[WfM98a] WFMC*:* Workflow Management Coalition Audit Data Specification. *Technischer Bericht, Workflow Management Coalition (WfMC), 1998.*

[WfM98b] WFMC*:* Workflow Security Considerations. *Technischer Bericht, Workflow Managemenr Coalition (WfMC), 1998.*

[WfM99a] WFMC*:* Workflow Management Coalition Terminology and Glossary. *Technischer Bericht, Workflow Management Coalition (WfMC), 1999.*

[WfM99b] WFMC*:* Workflow Management Coalition Workflow Standard - Interoperability Abstract Specification. *Technischer Bericht, Workflow Management Coalition (WfMC), 1999.*

[WfM08] WFMC*:* Process Definition Interface – XML Process Definition Language, Version 2.1a. *Technischer Bericht, Workflow Management Coalition (WfMC), 2008.*

[WfM11] WFMC*:* XPDL Implementations. *http://www.wfmc.org/xpdl-implementations.html, 2011.*

[WHKS98] WESKE, MATHIAS*,* JENS HÜNDLING*,* DOMINIK KUROPKA *und* HILMAR SCHUSCHEL*:* Objektorientierter Entwurf eines flexiblen Workflow-Management-Systems. *Informatik - Forschung und Entwicklung, 13:179–195, 1998.*

[Win07] WINNICKI, ALICE*:* Erweiterung einer Middleware für das Mobile Computing um nicht-funktionale Sicherheits- und Vertrauensaspekte. *Diplomarbeit, Universität Hamburg, Fachbereich Informatik, Verteilte Systeme und Informationssysteme, 2007.*

[WJ95] WOOLDRIDGE, MICHAEL *und* NICHOLAS R. JENNINGS*:* Intelligent Agents: Theory and Practice. *Knowledge Engineering Review, 10(2):115–152, 1995.*

[WJ06] WEISS, GERHARD *und* RALF JAKOB*:* Agentenorientierte Softwareentwicklung: Methoden und Tools. *Springer, Berlin, Heidelberg, New York, 2006.*

[WK03] WEBER, MICHAEL *und* EKKART KINDLER*:* The Petri Net Markup Language. *In:* Petri Net Technology for Communication-Based Systems*, Seiten 124–144. Springer-Verlag, 2003.*

[WKK+10] WETZSTEIN, BRANIMIR*,* DIMKA KARASTOYANOVA*,* OLIVER KOPP*,* FRANK LEYMANN *und* DANIEL ZWINK*:* Cross-Organizational Process Monitoring based on Service Choreographies. *In:* Proceedings of the 25th Annual ACM Symposium on Applied Computing (SAC 2010)*, Seiten 2485–2490. ACM, 2010.*

[WKN08] WIELAND, MATTHIAS*,* PETER KACZMARCZYK *und* DANIELA NICKLAS*:* Context Integration for Smart Workflows. *In:* Proceedings of the 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PERCOM 2008)*, Seiten 239–242. IEEE Computer Society, 2008.*

[WKNL07] WIELAND, MATTHIAS*,* OLIVER KOPP*,* DANIELA NICKLAS *und* FRANK LEYMANN*:* Towards Context-Aware Workflows. *In:* Proceedings of the 19th International Conference on Advanced In-

formation Systems Engineering: Workshops and Doctoral Consortium (CAiSE 2007)*, Seiten 577–591. Tapir Academic Press, 2007.*

[WML09a] WUTKE, DANIEL*,* DANIEL MARTIN *und* FRANK LEYMANN*:* A Method for Partitioning BPEL Processes for Decentralized Execution*. In:* Proceedings of the 1st Central-European Workshop on Services and their Composition (ZEUS 2009)*, Seiten 109–114. CEUR-WS.org, 2009.*

[WML09b] WUTKE, DANIEL*,* DANIEL MARTIN *und* FRANK LEYMANN*:* Tuplespace-based Infrastructure for Decentralized Enactment of BPEL Processes*. In:* Internationale Tagung Wirtschaftsinformatik (WI 2009)*, Seiten 1–10. OCG, 2009.*

[WNL08] WIELAND, MATTHIAS*,* DANIELA NICKLAS *und* FRANK LEYMANN*:* Managing Technical Processes Using Smart Workflows*. In:* Towards a Service-Based Internet*, Seiten 287–298. Springer, 2008.*

[Woo09] WOOLDRIDGE, MICHAEL J.*:* An Introduction to MultiAgent Systems*. John Wiley & Sons, Chichester, 2009.*

[WPH07] WOLTER, C.*,* H. PLATE *und* C. HEBERT*:* Collaborative Workflow Management for eGovernment*. In:* 18th International Conference on Database and Expert Systems Applications (DEXA 2007)*, Seiten 845–849. IEEE Computer Society, 2007.*

[WS04] WAGNER, JOHANN *und* KURT SCHWARZENBACHER*:* Föderative Unternehmensprozesse: Technologien, Standards und Perspektiven für vernetzte Systeme*. Publicis Corporate Publishing, Erlangen, 2004.*

[WS06] WILSON, KIRK *und* IGOR SEDUKHIN*:* Web Services Distributed Management: Management of Web Services (WSDM-MOWS) 1.1*. Technischer Bericht, Organization for the Advancement of Structured Information Standards (OASIS), 2006.*

[Wun09] WUNDERLICH, BENJAMIN*:* Entwicklung einer service-basierten Management-Schnittstelle für verteilte Geschäftsprozesse*. Diplomarbeit, Universität Hamburg, Fachbereich Informatik, Verteilte Systeme und Informationssysteme, 2009.*

[Wut10] WUTKE, DANIEL*:* Eine Infrastruktur für die dezentrale

Ausführung von BPEL-Prozessen. *Doktorarbeit, Universität Stuttgart, Institut für Architektur von Anwendungssystemen, 2010.*

[WW97] WODTKE, DIRK *und* GERHARD WEIKUM*:* A Formal Foundation for Distributed Workflow Execution Based on State Charts. *In:* International Conference on Database Theory (ICDT 1997)*, Seiten 230–246. Springer, 1997.*

[WW08] WALTER, PHILIPP *und* DIRK WERTH*:* Eine Peer-to-Peer Infrastruktur zur Konstruktion kollaborativer Geschäftsprozesse. *In:* Multikonferenz Wirtschaftsinformatik*, Seiten 61–62. GITO-Verlag, 2008.*

[WWWD96] WODTKE, DIRK*,* JEANINE WEISSENFELS*,* GERHARD WEIKUM *und* ANGELIKA KOTZ DITTRICH*:* The Mentor Project: Steps Toward Enterprise-Wide Workflow Management. *In:* Proceedings of the 12th International Conference on Data Engineering (ICDE 1996)*, Seiten 556–565. IEEE Computer Society, 1996.*

[YV02] YU, HAIFENG *und* AMIN VAHDAT*:* Design and evaluation of a conit-based continuous consistency model for replicated services. *ACM Transactions on Computer Systems (TOCS), 20:239–282, 2002.*

[YY07] YU, WEIHAI *und* JIE YANG*:* Continuation-passing Enactment of Distributed Recoverable Workflows. *In:* Proceedings of the 2007 ACM Symposium on Applied computing (SAC 2007)*, Seiten 475–481. ACM, 2007.*

[Zap07] ZAPLATA, SONJA*:* Collaborative Management of Distributed Business Processes - A Service-Based Approach. *In:* On the Move to Meaningful Internet Systems (OTM 2007): OTM Workshops*, Seiten 304–313. Springer, 2007.*

[ZBH+10] ZAPLATA, SONJA*,* DIRK BADE*,* KRISTOF HAMANN*,* WINFRIED LAMERSDORF*,* DANIEL STRASSENBURG *und* BENJAMIN WUNDERLICH*:* Ad-hoc Management Capabilities for Distributed Business Processes. *In:* 3rd International Conference on Business Process and Services Computing (BPSC 2010)*, Seiten 139–152. Bonner Köllen Verlag, 2010.*

[ZBHN+04] ZENG, LIANGZHAO*,* BOUALEM BENATALLAH*,* ANNE H.H. NGU*,* MARLON DUMAS*,* JAYANT KALAGNANAM *und* HENRY CHANG*:* QoS-Aware Middleware for Web Services

Composition. *IEEE Transactions on Software Engineering, 30:311–327, 2004.*

[ZBV09] ZAPLATA, SONJA, DIRK BADE *und* ANTE VILENICA*:* Service-based Interactive Workflows for Mobile Environments. *In:* Business Services: Konzepte, Technologien, Anwendungen - 9. Internationale Tagung Wirtschaftsinformatik (WI 2009)*, Seiten 631–640. Österreichische Computer Gesellschaft, 2009.*

[ZDL09a] ZAPLATA, SONJA, VIKTOR DREILING *und* WINFRIED LAMERSDORF*:* Realizing Mobile Web Services for Dynamic Applications. *In:* GODART, CLAUDE, NORBERT GRONAU, SUSHIL SHARMA *und* GEROME CANALS *(Herausgeber):* Proceedings of the 9th IFIP Conference on e-Business, e-Services, and e-Society (I3E 2009)*, Seiten 240–254. Springer, 2009.*

[ZDL09b] ZAPLATA, SONJA, VIKTOR DREILING *und* WINFRIED LAMERSDORF*:* Realizing Mobile Web Services for Dynamic Applications. *AIS Transactions on Enterprise Systems, 2009(2):3–12, 2009.*

[ZHKK10] ZAPLATA, SONJA, KRISTOF HAMANN *und* WINFRIED LAMERSDORF KRISTIAN KOTTKE*:* Flexible Execution of Distributed Business Processes based on Process Instance Migration. *Journal of System Integration (JSI), 1(3):3–16, 2010.*

[ZK07] ZAPLATA, SONJA *und* CHRISTIAN P. KUNZE*:* Prozessmanagement im Mobile Computing - Kooperative Ausführung von Geschäftsprozessen im Umfeld serviceorientierter Architekturen. *VDM Verlag Dr. Müller, Saarbrücken, 2007.*

[ZKL09a] ZAPLATA, SONJA, CHRISTIAN P. KUNZE *und* WINFRIED LAMERSDORF*:* Context-based Cooperation in Mobile Business Environments: Managing the Distributed Execution of Mobile Processes. *Business and Information Systems Engineering (BISE), 2009(4):301–314, 2009.*

[ZKL09b] ZAPLATA, SONJA, CHRISTIAN P. KUNZE *und* WINFRIED LAMERSDORF*:* Kontextbasierte Kooperation für mobile Geschäftsanwendungen: Dezentrale Ausführung und Management von mobilen Prozessen. *WIRTSCHAFTSINFORMATIK, 2009(4):347–362, 2009.*

[ZKML10] ZAPLATA, SONJA, KRISTIAN KOTTKE, MATTHIAS MEINERS *und* WINFRIED LAMERSDORF*:* Towards Runtime Migration of WS-BPEL Processes. *In:* 5th International Workshop on En-

gineering Service-Oriented Applications (WESOA'09)*, Seiten 477–487. Springer, 2010.*

[Zül98] ZÜLLIGHOVEN, HEINZ*:* Das objektorientierte Konstruktionshandbuch nach dem Werkzeug und Material- Ansatz*. Dpunkt, Heidelberg, 1998.*

[ZL10] ZAPLATA, SONJA *und* WINFRIED LAMERSDORF*:* Towards Mobile Process as a Service*. In:* 25th ACM Symposium On Applied Computing (SAC 2010)*, Seiten 372–379. ACM, 2010.*

[zM04] MUEHLEN, MICHAEL ZUR*:* Workflow-based Process Controlling. Foundation, Design, and Implementation of Workflow-driven Process Information Systems*. Logos-Verlag, Berlin, 2004.*

[ZML11] ZAPLATA, SONJA*,* MATTHIAS MEINERS *und* WINFRIED LAMERSDORF*:* Designing Future-Context-Aware Dynamic Applications with Structured Context Prediction*. Software – Practice and Experience (SPE), 10.1002/spe.1126:1–18, 2011.*

[zMR00] MUEHLEN, MICHAEL ZUR *und* MICHAEL ROSEMANN*:* Workflow-Based Process Monitoring and Controlling - Technical and Organizational Issues*. In:* Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS 2000)*, Seiten 6032–6042. IEEE Computer Society, 2000.*

[Zuk97] ZUKUNFT, OLAF*:* Adaptation in Mobile Workflow Management Systems*. Personal and Ubiquitous Computing, 1(3):197–202, 1997.*

[ZVBK09] ZAPLATA, SONJA*,* ANTE VILENICA*,* DIRK BADE *und* CHRISTIAN P. KUNZE*:* Abstract User Interfaces for Mobile Processes*. In:* 16. Fachtagung Kommunikation in Verteilten Systemen (KiVS 2009)*, Seiten 129–140. Springer, 2009.*

Eidesstattliche Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht.

Hamburg, den 25. Oktober 2012

Sonja Zaplata