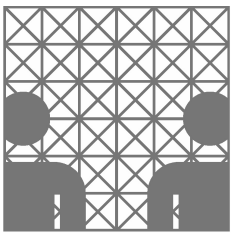

Department Informatik
Fakultät für Mathematik,
Informatik und Naturwissenschaften
Universität Hamburg

Dissertation zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)

Interaktionsorientierte Komposition virtueller Dienstleistungsprozesse

Vorgelegt von
Christian Zirpins
aus Hamburg

Hamburg 2007

kassel university press

Genehmigt von der MIN-Fakultät, Department Informatik
an der Universität Hamburg auf Antrag der Gutachter

Prof. Dr. W. Lamersdorf, Universität Hamburg (Betreuer und 1. Gutachter)
Prof. Dr. W. Emmerich, University College London (2. Gutachter)
Dr. D. Moldt, Universität Hamburg (3. Gutachter)

Hamburg, den 21. Juni 2007 (Tag der Disputation)

Prof. Dr. W. Lamersdorf (Department-Leiter)

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de/ abrufbar

ISBN 978-3-89958-341-0
URN urn:nbn:de:0002-3416

Druck und Verarbeitung: Unidruckerei der Universität Kassel
Printed in Germany

Zusammenfassung

Die vorliegende Arbeit untersucht die Anwendung Service-orientierter Software-Techniken im Kontext der Planung und Steuerung von Produktionsvorgängen für Dienstleistungen in virtuellen Unternehmensnetzwerken und entwickelt einem Ansatz zur *interaktionsorientierten Komposition virtueller Dienstleistungsprozesse*. Den Hintergrund bilden zwei simultane Trends in der Betriebswirtschaft und Organisation von Dienstleistungen sowie der Software-Technik verteilter betrieblicher Anwendungssysteme: Zum einen wurden für die Produktion von Dienstleistungen moderne Prinzipien virtueller Organisation vorgeschlagen, die eine informationstechnische Unterstützung von Planungs- und Steuerungsfunktionen des Produktionsmanagements voraussetzen. Zum anderen eröffnet die Entwicklung Service-orientierter Software-Techniken neue Möglichkeiten zur Realisierung zwischenbetrieblicher Anwendungssysteme auf Basis Service-orientierter Software-Architekturen. Die Motivation dieser Arbeit besteht darin, die Zusammenführung der komplementären Ansätze wissenschaftlich zu untersuchen und eine entsprechende Lösung zu entwickeln.

Das diesbezügliche Vorgehen basiert auf einer allgemeinen Untersuchung der Themengebiete virtueller Dienstleistungsproduktion und Service-orientierter Techniken zur Prozessintegration in virt. Dienstleistungsunternehmen. Hierfür wird ein generischer Ansatz in Form eines Rahmenwerks mit konzeptionellen und technischen Anteilen entwickelt. Auf konzeptioneller Ebene wird ein vielschichtiger Ansatz zur Anwendung Service-orientierter Techniken für die Planung und Steuerung virtueller Dienstleistungsproduktion erarbeitet. Dies beruht darauf, Service-orientierte Techniken zur Spezifikation und Automatisierung zwischenbetrieblicher Interaktionen zur Virtualisierung von Dienstleistungen im Hinblick auf den inhärenten Interaktionsaspekt des zentralen Dienstleistungsprozesses zu nutzen. In der Folge wird dann mittels Komposition virtueller Dienstleistungsprozesse die Koordination kooperativer Vorgänge zwischen den Teilnehmern im virt. Dienstleistungsunternehmen geregelt. Auf technischer Ebene werden praktische Mittel realisiert, die das konzeptionelle Rahmenwerk operational manifestieren. Hierbei wird das Prinzip einer Software-Entwicklung zum Produktionsmanagement mittels Konzeption konkreter Entwicklungsmethoden und -mechanismen umgesetzt. Sie verbinden Planung und Steuerung virtueller Dienstleistungsproduktion mit Entwurf und Implementierung Service-orientierter Anwendungssysteme. Die allgemeine Untersuchung und das generische Rahmenwerk werden durch ein Szenario der Güterlogistik und dessen Unterstützung durch das im Forschungsprojekt FRESCO realisierte Rahmenwerk exemplarisch veranschaulicht und evaluiert.

Abstract

The work at hand describes an examination of the application of service-oriented software techniques in the context of planning and control of production processes for economic services in virtual enterprises and a subsequent approach of *Interaction-oriented composition of virtual service processes*. It builds on two parallel trends in the areas of business administration and organization of economic services as well as software engineering of distributed information systems. On the one hand side, modern principles of virtual organization have been proposed for production of economic services that generally demand for information technology support of planning and control functions within production management. On the other hand, development of service-oriented software techniques opens up new possibilities for realizing cross-organizational information systems based on service-oriented software architecture. The motivation for the work at hand is to scientifically examine the combination of these two obviously complementary approaches and to develop an accordant solution.

The course of the work starts with an examination in the fields of virtual production of economic services and service-oriented techniques for process integration in virtual service enterprises. As such a technique, a generic approach in the form of a framework with conceptual and technical parts is developed. The conceptual part prepares ground for applying service-oriented techniques to the planning and control functions of virtual service production. It is based on the potential of service-oriented architecture to describe and automate cross-organizational interactions. This potential is applied toward the virtualization of economic services as regards the inherent interactive characteristic of service processes. Subsequently, composition of virtual service processes is used to govern the coordination of cooperative procedures between participants of a virtual service enterprise. The technical part realizes practical means to put the conceptual framework into operation. Therefore a principle of software development for production management is realized by concrete development methods and mechanisms that connect planning and control of virtual service production with design and implementation of service-oriented information systems. The general examination and generic framework are illustrated and evaluated by a scenario of freight logistics and its support by a framework that has been developed in the research project FRESCO.

Danksagung

Im Zusammenhang mit der Erstellung der vorliegenden Arbeit gebührt verschiedenen Personen mein persönlicher Dank. An erster Stelle danke ich Professor Doktor Winfried Lamersdorf für die Betreuung der vorliegenden Arbeit und die langjährige Begleitung in allen Belangen meiner akademischen Entwicklung. Zudem möchte ich insbesondere Professor Doktor Wolfgang Emmerich und Doktor Daniel Moldt für ihr Interesse an den Ergebnissen meiner Arbeit und deren für mich sehr wichtige Diskussion danken.

Ein wichtiger Bestandteil wissenschaftlicher Forschung und Entwicklung ist immer auch die Interaktion, der Austausch und die Kooperation mit Kolleginnen und Kollegen. In diesem Zusammenhang danke ich zunächst allen meinen Kollegen in der VSIS-Gruppe des Department Informatik an der Universität Hamburg. Insbesondere danke ich Tobias Baier, Frank Griffel, Giacomo Picinelli und Harald Weinreich für die inhaltliche Zusammenarbeit sowie Anne Hansen-Awizen und Volker Nötzold für ihre unverzichtbare organisatorische und technische Unterstützung.

Gerade für einen Doktoranden ist zudem die Zusammenarbeit mit Studenten ein entscheidender Faktor. Auch in den verschiedenen Projekten, die letztendlich zur vorliegenden Arbeit geführt haben, war die intensive Zusammenarbeit mit engagierten studentischen Teams von großer Bedeutung. Hierfür danke ich insbesondere Heiko Adam, Henning Brandt, Olaf Fischer, Benjamin Kirchheim, Thomas Plümpe und Bernhard Wenzel, deren Arbeiten in das FRESCO-Projekt eingeflossen sind. Des Weiteren danke ich Matthias Ferdinand, Olaf Gregor, Sonja Heuer, Jewgeni Kravets, Kay Kröger, Sven Offermann, Andre Schmidt, Robertino Solanas und besonders Kevin Schütt für die vielen fruchtbaren Diskussionen im Rahmen ihrer Arbeiten im weiteren Themenbereich.

An herausragender Stelle ist schließlich meine Familie zu nennen. Allen voran danke ich meiner Frau Ursula Zirpins, an der die Mühen der Dissertation nicht spurlos vorbeigehen konnten und die mich das nie hat spüren lassen. Ihrer Geduld und tagtäglichen Fürsorge verdanke ich die Kraft, die nötig war, um diese Arbeit fertig zu stellen. Am Ende, aber eigentlich als Grundlage von allem, danke ich meinen Eltern Elke und Werner Zirpins, die mir mein Leben lang mit Rat und Förderung beigestanden haben. Durch sie bin ich an den Punkt gelangt diese Arbeit verfassen zu können.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

.Net	"The Microsoft Web Services Strategy"
2PC	Two-Phase Commit Protocol
A2A	Application-to-Application
A2Ai	A2A-Integration (EAI-Synonym)
Abb.	Abbildung
ABWL	allg. Betriebswirtschaftslehre
ACP	Algebra of Communicating Processes with Abstractions
Apache	Apache Software Foundation
API	Application Program Interface
AS	Anwendungssystem
ASP	Active Server Pages
B2B	Business-to-Business
B2Bi	B2B-Integration
BO	Business Object
BPEL	Business Process Execution Language (OASIS WS-Standard)
BPM	Business Process Modelling
BPMI	Business Process Modelling Initiative
BPR	Business Process Reengineering
BPSS	Business Process Specification Schema (ebXML-Standard)
BPWS4J	IBM BPEL Java Run Time
BWL	Betriebswirtschaftslehre
bzgl.	bezüglich

bzw.	beziehungsweise
CALS	Computer Aided Acquisition and Logistics Support
CCM	CORBA Component Model (Teil des OMG-CORBA-Standards)
CCS	Calculus of Concurrent Systems
CGI	Common Gateway Interface
CICS	Customer Information and Control System (IBM-TP-Monitor)
CIM	Computer Integrated Manufacturing
CIMOSA	Open Systems Architecture for CIM
CI/Srv	Client/Server
COM	Common Object Model (Microsoft Standard)
CORBA	Common Object Request Broker Architecture (OMG-Standard)
COTS	Commercial of-the-Shelf
CPA	Collaboration Protocol Agreement(ebXML-Standard)
CSCW	Computer Supported Cooperative Work
CSP	Communicating Sequential Processes
CSS	Community Support System
CVS	Concurrent Versions System (auch: Concurrent Versioning System)
cXML	Commerce XML
DAML-S	DARPA Agent Markup Language for Web Services
DARPA	Defense Advanced Research Projects Agency
DB	Datenbank
DBMS	Datenbankmanagementsystem
DBP	Distributed Business Process (zwischenbetrieblich) verteilter Geschäftsprozess

DCE	Distributed Computing Environment (OSF-Standard)
DCOM	Distributed COM (Microsoft Standard)
Def.	Definition
d.h.	das heißt
DLP	Dienstleistungsprozess
DLU	Dienstleistungsunternehmen
DMS	Dokumentenmanagementsystem
DMZ	Demilitarisierte Zone
DSML	domänenspezif. Modellierungssprache (Domain-Specific Modeling Language)
DTD	Document Type Definition
EA	Enterprise-Architektur
EAI	Enterprise Application Integration
EBNF	Erweiterte Backus-Naur-Form
E-Business	Electronic Business (auch 'B2B E-Commerce')
ebXML	Electronic Business XML (UN-Standard)
ECA	Event-Condition-Action
E-Commerce	Electronic Commerce (elektr. Handel)
EDI	Electronic Data Interchange (UN-Standard)
EDIFACT	Electronic Data Interchange for Administration, Commerce and Transport (UN-Standard)
EEM	Enterprise Engineering Methodology (aus GERAM)
EET	Enterprise Engineering Tool (aus GERAM)
EGA	Enterprise Grid Alliance
EJB	Enterprise Java Beans (J2EE-Standard)
EM	Enterprise Model (aus GERAM)

EMEIS	Enterprise Model Execution and Integration Services (aus GERAM)
EML	Enterprise Modelling Language (aus GERAM)
EMO	Enterprise Module (aus GERAM)
EOS	Enterprise Operative System (aus GERAM)
E-Service	Service-orient. IT-Repräsentation einer virt. Dienstleistung
ESMS	E-Service-Management-System
et al.	und andere
etc.	und so weiter
EU	Europäische Union
EVA	Eingabe-Verarbeitung-Ausgabe
f.	folgend
FAQ	frequently asked questions
ff.	fortfolgend
FIS	Führungsinformationssystem
FreightMixer	Name eines Szenarios sowie des darin betrachteten VDU
FRESCO	Foundational Research on Service Composition (Forschungsprojekt)
FRESCO-SOM	Service-orient. FRESCO-Modell zur Realisierung virt. Dienstleistungen
FRESCO-TK	FRESCO Toolkit
FSM	FRESCO E-Service-Metamodell
FSMV	FRESCO E-Service-Management-Vorgehensmodell
FTP	File Transfer Protocol (Internet Standard)
GEMC	Generalised Enterprise Modelling Concepts (aus GERAM)
GERA	Generalised Enterprise Reference Architecture (aus GERAM)

GERAM	Generalised Enterprise Reference Architecture And Methodology
ggf.	gegebenenfalls
GGF	Global Grid Forum
Globus	Globus Alliance
GMT	Greenwich Mean Time
GSH	Grid Service Handle (aus OGSI)
GSR	Grid Service Reference (aus OGSI)
GT3 Core	GT3 Core Infrastructure
GT3	Globus Toolkit 3
GUI	Graphical User Interface
GWSDL	Grid WSDL
HP	Hewlett-Packard
HTML	Hypertext Markup Language (W3C-Standard)
HTTP	Hypertext Transfer Protocol (IETF-Standard)
IBL	Industriebetriebslehre
IBM	International Business Machines
IDE	Integrated Development Environment
IDL	Interface Definition Language
IETF	Internet Engineering Task Force
IEWS	Inter-Enterprise (zwischenbetriebliches) Workflow-System
IIOP	Internet Inter-ORB Protocol (Teil des OMG-CORBA-Standards)
IS	Informationssystem
ISO	International Organisation for Standardization
IT	IV-Technik
IV	Informationsverarbeitung

J2EE	Java 2 Platform, Enterprise Edition
J2SE	Java 2 Platform, Standard Edition
JAAS	Java Authentication and Authorization Service (J2EE-Standard)
JAXP	Java API for XML Processing
JAX-RPC	Java API for XML-Based RPC
JCA	Java Connector Architecture (J2EE-Standard)
JDBC	JDBC Technology (Java-API)
JDK	J2SE Development Kit
JMS	Java Messaging Service (J2EE-Standard)
JNDI	Java Naming and Directory Interface (J2EE-Standard)
JSP	Java Server Pages (Java-API)
KI	Künstliche Intelligenz
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LKW	Lastkraftwagen
MAS	Multiagentensystem
MDA	modellgetriebene Architektur (Model-Driven Architecture, OMG-Standard)
MDD	modellgetrieb. Entwicklung (Model-Driven Development)
MOM	Message Oriented Middleware
N-Form	Netzwerkorganisation (Synonym)
NIST	U.S. National Institute of Standards and Technology
Nr.	Nummer
OASIS	Organization for the Advancement of Structured Information Standards

OCL	Object Constraint Language (UML-Bestandteil)
ODBC	Open Database Connectivity
ODP	Open Distributed Processing (ISO-Standard)
OGF	Open Grid Forum
OGSA	Open Grid Service Architecture (OGF-Standard)
OGSI	Open Grid Service Infrastructure (OGF-Standard)
OLTP	Online Transaction Processing
OMG	Object Management Group
OO	Objektorientierung (Programmierparadigma)
ORB	Object Broker (bzw. Object Request Broker im CORBA Kontext)
orient.	orientiert
OSF	Open Software Foundation
OSI	Open Systems Interconnect (ISO-Standard)
OWL	Web Ontology Language (W3C-Standard)
P2P	Peer-to-Peer (Punkt-zu-Punkt)
PC	Personal Computer
PEM	Partial Enterprise Model (aus GERAM)
PIF	Process Interchange Format
PIM	plattformunabhäng. Modell (Platform-Independant Model, MDA-Konzept)
PIP	Partner Interface Process (RosettaNet-Standard)
PPS	Produktionsplanung und -steuerung
PSL	Process Specification Language (NIST-Standard)
PSM	plattformspezif. Modell (Platform-Specific Model, MDA-Konzept)

PuK	Planungs- und Kontrollsystem
QName	Qualified Name (XML-Schema-Typ für global eindeutige Bezeichner)
QoS	Quality of Service
RDF	Ressource Description Framework (W3C-Standard)
RMI	Remote Method Invocation (Java-API)
RosettaNet	RosettaNet Konsortium
RPC	Remote Procedure Call
S.	Seite
SDE	Service Data Element (aus OGSI)
SGF	strat. Geschäftsfeld
SMTP	Simple Mail Transfer Protocol (IETF-Standard)
SO	Service-orient.
SOA	Service-orient. Architektur
SOAP	Simple Object Access Protocol (W3C-Standard)
SOC	Service Oriented Computing
SOM	Service-orient. Modell
s.(o./u.)	siehe (oben/unten)
Shoe&Shoe	Name eines Akteurs im FreightMixer-Szenario
STEP	Standard for the Exchange of Product Model Data (ISO-Standard)
strat.	strategisch
Sun	Sun Microsystems
SWAP	Simple Workflow Access Protocol (IETF-Standard)
Tab.	Tabelle

TCP/IP	Transmission Control Protocol/Internet Protocol (Bez. für Protokollstandards sowie das aufbauende Referenzmodell)
TP	Transaction Processing (im Kontext von TP-Monitor Middleware)
u.	und
u.a.	unter anderem
UBR	Universal Business Registry (Teil von UDDI)
UDDI	Universal Description, Discovery and Integration (Standard des UDDI-Konsortiums)
UML	Unified Modelling Language (OMG-Standard)
UN	United Nations
URI	Unified Ressource Identifier (IETF-Standard)
URL	Unified Ressource Locator (IETF-Standard)
VDL	virt. Dienstleistung
VDLP	virt. Dienstleistungsprozess
VDPN	virt. Dienstleistungsproduktionsnetzwerk
VDU	virt. Dienstleistungsunternehmen
vert.	verteilt
vgl.	vergleiche
virt.	virtuell
VM	virtual Machine
VPN	virt. Produktionsnetzwerk
VS	vert. System
VU	virt. Unternehmen
W3C	World Wide Web Consortium
WAN	Wide Area Network

WAPI	Workflow Application Program Interface
Web	World Wide Web
WfMC	Workflow Management Coalition
WfMC-RM	Workflow Reference Model (WfMC-Standard)
WfMS	Workflow-Management-System
WF-net	Workflow Net
WS	Web Service (W3C-Initiative)
WS-CDL	Web Services Choreography Description Language (W3C-Standard)
WSCI	Web Service Choreography Interface (W3C-Standard)
WSCL	Web Service Conversation Language (W3C-Standard)
WSDL	Web Service Definition Language (W3C-Standard)
WSDL4J	Java API for WSDL
WSFL	Web Service Flow Language (IBM-Standard)
WS-I	Web Service Interoperability Organization
WSS	Workgroup Support System
WWW	World Wide Web
xCBL	XML Common Business Library
XML	Extensible Markup Language (W3C-Standard)
XPath	XML Path Language (W3C-Standard)
XPDL	XML Process Definition Language (WfMC-Standard)
z.B.	zum Beispiel

1. Einleitung

Die vorliegende Arbeit setzt sich zum Ziel, virt. Dienstleistungsunternehmen (VDU) durch *E-Services* zu unterstützen, die eine flexible Komposition von Dienstleistungsprozessen auf Basis wechselnder organisatorischer Interaktionsbeziehungen erlauben.

Im Folgenden wird zunächst der thematische Kontext eingeführt und die Problemstellung abgeleitet. Dann werden die Untersuchungsziele abgegrenzt, was sowohl den Umfang des Lösungskonzepts als auch die dabei verwendeten Methoden betrifft. Schließlich werden der Gang der Untersuchungen und die Struktur der vorliegenden Arbeit dargestellt.

1.1. Kontext und Problemstellung

Produktion von Dienstleistungen in virt. Unternehmen

In der ökonomischen Theorie werden Dienstleistungen durch die nutzenstiftende Wirkung weitgehend immaterieller und stark interaktiver Leistungsprozesse charakterisiert [Cor01]. Die Produktion solcher Dienstleistungen ist maßgeblich durch den Einfluss externer Produktionsfaktoren bestimmt: Der Nachfrager bringt sich unmittelbar in den Produktionsprozess ein und bezieht kontinuierlich dessen Ergebnisse. Aus diesem Grund kommt der prozessorientierten Organisation der Dienstleistungsproduktion grundsätzlich eine entscheidende Bedeutung zu [Har03]. Neben einer generellen Prozessoptimierung muss der Produktionsprozess von Dienstleistungen jedoch wettbewerbsbedingt möglichst individuell an den Anforderungen des einzelnen Nachfragers ausgerichtet werden.

Die dazu notwendige Flexibilität lässt sich nach den Erkenntnissen moderner Managementforschung durch eine Virtualisierung der produktiven Organisationsstrukturen (als virt. Unternehmen) erreichen [PN98]. Im Rahmen strategischer Netzwerke bringen sich die teilnehmenden Unternehmen dabei jeweils mit ihren Kernkompetenzen ein. Sie bilden einen Kompetenzpool, der vielfältige Möglichkeiten zur Komposition individueller Leistungen bietet. Die häufige Rekonfiguration der unternehmensübergreifenden Wertschöpfungsketten wird dabei zum einen durch wechselseitiges Vertrauen der strategischen Partner und zum anderen durch die Integration ihrer betrieblichen IT-Systeme möglich. Auf eine Institutionalisierung zentraler Managementfunktionen wird dabei verzichtet.

Hinter der Idee virt. Dienstleistungsunternehmen steht das Ziel, die Produktion kundenspezifischer, immaterieller und interaktiver Leistungsprozesse optimal zu unterstützen [PN98]. Die Produktion in Unternehmensnetzwerken (so genannte Pro-

duktionsnetzwerke) stellt aber auch neue Anforderungen an die Produktionsplanung und -steuerung [PN00, Fis00]. Hier werden u. a. neue Konzepte für die Regelung der Koordination von kooperativen Tätigkeiten der Netzwerkunternehmen benötigt. Auf der Netzwerkebene müssen hierarchische Anweisungen dabei weitgehend durch Selbstabstimmung der autonomen Netzwerkunternehmen ersetzt werden. Die Regelung der Koordination konzentriert sich dann auf die Interaktion der Akteure [SK00].

Steuerung virt. Dienstleistungsproduktion mittels SOC-Techniken

Zur Umsetzung einer flexiblen Produktionssteuerung für VDU untersucht die vorliegende Arbeit informationstechnische Grundlagen zur Virtualisierung von Dienstleistungen. Solche virt. Dienstleistungen (VDLs) sollen dabei die kompositorische Struktur des Leistungsprozesses informationstechnisch abbilden. Dies betrifft in erster Linie einen Steuerprozess, der die Interaktion autonomer Hilfsprozesse einzelner Netzwerkunternehmen im Sinne eines virtuellen Kernprozesses regelt. Während der Dienstleistungsproduktion soll die virtuelle Dienstleistung parallel zur realen ablaufen. Dabei soll sie die Informationssysteme (d. h. Mensch/Aufgabe/Technik-Systeme) der Netzwerkunternehmen koppeln und darüber deren Interaktionen aktiv koordinieren.

Auf systemtechnischer Ebene wird die unternehmensübergreifende Kopplung produktiver Anwendungssysteme u. a. durch aktuelle Techniken des *Service Oriented Computing (SOC)* [PG03] ermöglicht. Hier sind besonders *Web Services (WS)* [ACK+04] und WS-basierte *Grids* [FKN+02a] zu nennen. Diese Techniken nutzen XML-basierte Protokolle für die Interaktion zwischen autonomen Anwendungssystemen. Sie erlauben dadurch interoperablen Zugriff auf deren Funktionen und Prozesse über das Internet. SOC erscheint daher grundsätzlich geeignet, das Konzept virt. Dienstleistungen (System-) technisch zu untermauern, d. h. dass eine Konzeption und Umsetzung von VDL auf Basis spezifischer Service-orient. Anwendungssysteme möglich erscheint. Diese können als Form kooperativer Informationssysteme[1] klassifiziert werden, die das Management virt. Dienstleistungen (im Wesentlichen Planung und Steuerung der Dienstleistungsproduktion) wirkungsvoll Unterstützen können. Konkret besteht ihr Potenzial darin, den technischen Rahmen zur flexiblen Prozessintegration in VDU zu schaffen.

Für derartige Anwendungssysteme bieten die konzeptionellen Modelle aktueller Standards[2] oder Forschungsprojekte[3] im Bereich prozessorientierter SOC- bzw. E-Business Ansätze jedoch noch keine spezifischen Lösungen. Auf konzeptioneller Ebene wird nämlich ein Rahmen benötigt, der es teilnehmenden Dienstleistungs-

[1] Im Sinne des englischen Begriffs des „cooperative informationsystem“, der ein integriertes betriebliches Anwendungssysteme über die Grenzen kooperierender Organisationen hinweg meint [DMDJ+97].

[2] Z. B. RosettaNet Partner Interface Process, ebXML Business Process Specification Schema etc. – siehe Übersicht in [MBB+03].

[3] Z. B. eFlow [CS01], CrossFlow [GAH+00] etc.

unternehmen erlaubt, die zusammenhängenden Interaktionsprozesse mit anderen Dienstleistungsunternehmen, Kunden und Lieferanten innerhalb und außerhalb des Produktionsnetzwerks zu erfassen sowie bei Rekonfiguration der virtuellen Organisation anzupassen. Zurzeit existiert aber kein ausreichend integratives und flexibles Modell solcher Prozessketten.

Darüber hinaus stoßen die gängigen prozessorientierten Unterstützungsmechanismen der Web Service-Architektur[4] an ihre Grenzen. Auf technischer Ebene wird eine Ausführungsplattform benötigt, die eine durchgehende Steuerung und Kontrolle ganzheitlicher Prozessketten erlaubt und deren parallelen Ablauf sowie häufige Änderungen unterstützt. Die zurzeit verfügbaren Ausführungsplattformen erschweren eine derartige dynamische Prozesskopplung zum Teil erheblich.

1.2. Zielsetzung und Methodik

Die wissenschaftliche Untersuchung zielt generell auf die Konzeption und Realisierung eines IT-basierten Rahmenwerks zur Unterstützung von VDU durch Koordination und Steuerung der Produktion virt. Dienstleistungen. Hierbei soll das Instrumentarium des SOC zum einen als Grundlage dienen und zum anderen erweitert werden. Die Basis wird sowohl durch generische Konzepte des Service-orient. Modells (SOM) [PG03] für verteilte zwischenbetriebliche Anwendungssysteme als auch durch konkrete Service-orient. Architektur (SOA) auf Basis von Web Service-Techniken gebildet. Darauf soll das Rahmenwerk mit sowohl konzeptionellen als auch technischen Anteilen aufsetzen. Diese sollen zum einen auf abstrakter Ebene allgemeingültig entwickelt und zum anderen anhand einer exemplarischen Umsetzung im Rahmen des FRESCO-Projekts[5] konkret veranschaulicht werden. Die Realisierung des Rahmenwerks ist das konkrete Ziel der vorliegenden Arbeit und wird im Folgenden beschrieben.

Konzeption von E-Services als prozessbasierte SOC-Technik für VDL

Der konzeptionelle Anteil des Rahmenwerks zielt auf eine Erweiterung aktueller Prozessmodelle des Service Oriented Computing. Der Fokus liegt auf Konzepten zur Modellierung und Realisierung flexibel verketteter Interaktionsprozesse im Kontext virtueller Dienstleistungsproduktion. Um diese Konzepte zu begründen, muss in einem ersten Schritt die Ausarbeitung eines konzeptionellen Modells für virt. Dienstleistung erfolgen. Hierbei sollen die Interaktionen zwischen den Akteuren virt. Dienstleistungsproduktion als Grundlage eines Virtualisierungskonzepts für Dienstleistungen

[4] Hiermit sind im Wesentlichen Laufzeitmechanismen für den BPEL-Standard [ACD+03] gemeint.

[5] Foundational Research on Service Composition (FRESCO) [PZL03] ist ein kooperatives Forschungsprojekt der Universität Hamburg mit den HP-Labs Bristol, in dem der Autor der vorliegenden Arbeit die wissenschaftliche Leitung inne hatte. Eine nähere Beschreibung folgt in Kapitel 4.

herangezogen werden. VDL sollen diese Interaktionen als virt. Dienstleistungsprozesse (VDLP) sowohl explizit herausstellen als auch aktiv steuern. Ziel ist es, auf diese Weise die Koordination von kooperativen Vorgängen der Akteure zu regeln und dies zur Produktionsplanung und -steuerung einzusetzen. Ein entsprechender Dienstleistungsbegriff soll auf Basis der ökonomischen Grundlagen informal eingeführt und als konzeptionelles Modell fixiert werden.

In einem zweiten Schritt ist das Modell virt. Dienstleistungen dann in den Kontext Service-orient. Anwendungssysteme zu übertragen. Das Ziel ist hier zunächst, generelle Prinzipien für die Service-orient. Entwicklung von Anwendungssystemen zur Steuerung von Interaktionsprozessen im Rahmen des VDLP abzuleiten. Die Klasse dieser Anwendungssysteme grenzt den Begriff des *E-Service* ab und deren Entwicklung im VDU-Kontext begründet das *E-Service-Management*.[6] Im Anschluss soll die Kategorie der E-Services in Form eines Service-orient. Metamodells konkretisiert werden. Als Basis hierfür ist das Referenzmodell der *Workflow Management Coalition (WfMC)* [WfM04] vorgesehen, das ein generisches, erweiterbares und gleichsam in der Anwendungspraxis anerkanntes Prozessmodell bietet [Hol95]. Aufbauend sollen fundamentale Konzepte der Service-orient. Koordination und Komposition erfasst und durch (i) interaktionsspezifische Abstraktionen und (ii) Strukturierungsmittel im Sinne des VDL-Modells erweitert werden. Ziel dieser Vorgehensweise ist die Erschaffung eines generischen Metamodells, das einen organisationsübergreifenden Konsens erlaubt und technikneutral ist.

Zur Komplettierung des konzeptionellen Rahmenwerks soll auf Basis des E-Service-Metamodells ein E-Service-Management-Vorgehensmodell entwickelt werden. Hierzu ist zunächst ein genereller Service-orient. Entwicklungslebenszyklus für E-Services zu bestimmen, bei dem Entwurf, Implementierung, Ausführung und Anpassung von E-Services den Lebenszyklus virt. Unternehmen zur Dienstleistungsproduktion optimal unterstützen. Ein wichtiger Aspekt liegt dabei in der Agilität und Flexibilität des Entwicklungsprozesses zur schnellen Realisierung virtueller Dienstleistungen bei Rekonfiguration der virtuellen Dienstleistungsorganisation.

Realisierung von E-Services auf Basis von OGSA und BPEL

Der praktische Anteil des Rahmenwerks soll das Konzept virt. Dienstleistung zur VDU-Steuerung technisch umsetzen. Dazu soll eine Service-orient. IT-Umgebung zur Unterstützung des E-Service-Entwicklungslebenszyklus realisiert werden. Für solche E-Service-Management-Systeme (ESMS) ist sowohl ein genereller Entwurf als auch eine prototypische Implementierung zu erarbeiten.

[6] Hiermit wird klar, dass die vorliegende Arbeit den Begriff des „E-Service“ zunächst ohne Bezug auf bestehende Konzepte mit gleichem oder ähnlichen Namen verwendet. Er wird stattdessen grundsätzlich und eigenständig als Service-orientierter Ansatz für virt. Dienstleistungen eingeführt und im Verlauf der Arbeit ausgestaltet.

ESMS müssen sowohl Anteile einer horizontalen Infrastruktur als auch vertikaler Fachfunktionen für VDU umfassen. Als horizontale Infrastruktur soll eine Plattform zur Ausführung von E-Services entwickelt werden, die die automatische Steuerung und Kontrolle von Interaktionen im Rahmen virt. Dienstleistungsprozesse erlaubt. Die Plattformarchitektur soll dabei auf der Service-orient. OGSA-Architektur[7] aufsetzen, die als Infrastruktur für virt. Organisationen ausgelegt ist [FKT01]. Zur automatisierten Steuerung von Interaktionsprozessen soll die OGSA-Architektur durch eine BPEL-Laufzeitumgebung ergänzt werden. Hierzu müssen die Komponentenmodelle von OGSA und BPEL durch technische Maßnahmen angeglichen werden. Aufbauend sind Konzepte und Mechanismen zur Aggregation von OGSA-Komponenten und BPEL-Prozessen eines E-Service zu ergänzen.

Daneben müssen ESMS vertikale Fachfunktionen zur Unterstützung des E-Service-Managements bereitstellen. In Bezug auf die geforderte Agilität und Flexibilität des Vorgehensmodells soll als Grundkonzept der E-Service-Entwicklung ein modellgetriebener Ansatz verfolgt werden. Dieser soll eine möglichst integrierte Methodik für Entwurf, Implementierung und Änderung von E-Services auf Basis des entwickelten Metamodells umfassen. Konkret sollen Methoden zum UML-basierten Entwurf und zur XPDL-basierten Spezifikation von E-Service-Modellen, zur automatischen Transformation dieser Modelle in ausführbare BPEL-Prozesse sowie zur systematischen und konsistenten Modelltransformation erarbeitet werden.

Die Entwürfe der horizontalen Infrastruktur und vertikalen Fachfunktionen von ESMS sollen schließlich in einer prototypischen Implementierung umgesetzt werden. Die ESMS-Plattform soll dabei auf der OGSA-Referenzimplementierung[8] basieren. Daneben soll eine BPEL-fähige WS-Plattform eingebunden werden. Diese systemtechnische Basis soll dann um die spezifischen Konzepte des ESMS erweitert werden. Dies sind zum einen homogene ESMS-Plattform- und -Komponentenarchitekturen mit entsprechenden Software-Rahmenwerken sowie Laufzeitmechanismen zur Integration und dynamischen Bindung multipler BPEL-Prozesse und OGSI-Komponenten. Zum anderen sind das integrierte Entwicklungsmechanismen und interaktive Werkzeuge zur Umsetzung von Fachfunktionen des E-Service-Managements.

1.3. Ergebnisse und Beiträge

Die oben beschriebenen Zielsetzungen werden in der vorliegenden Arbeit mithilfe der vorgesehenen Methoden eingehend untersucht. Bevor im Folgenden die detaillierte Dokumentation dieser Untersuchung erfolgt, sollen an dieser Stelle die wichtigsten Untersuchungsergebnisse und deren Beiträge zur Wissenschaft und Praxis vorweggenommen werden.

[7] Open Grid Service Architecture (OGSA) [FKN+02a].
[8] Globus Toolkit 3 [Glo06].

Als Gesamtergebnis der Untersuchung ist es gelungen, das angestrebte konzeptionelle und technische Rahmenwerk zur Unterstützung von Planung und Steuerung der Produktion in virt. Dienstleistungsunternehmen auf Basis des SOC-Paradigmas zu realisieren und damit die Hypothese der vorliegenden Arbeit zu bestätigen. Die in diesem Zusammenhang erbrachten Beiträge betreffen vor allem die folgenden Teilergebnisse:

- *Konzeptionelles Modell virt. Dienstleistungen* (Sektion 4.2, S. 212 ff.) – Eine generelle Theorie zur Virtualisierung von Dienstleistungsprozessen auf Basis interaktionsorientierter Koordinationsmodelle der Organisationslehre bildet die Grundlage für spezifische Dienstleistungsmodelle. Das FRESCO-Dienstleistungsmodell liefert einen konkreten Ansatz zur direkten Anwendung [PZL03, GPZ03].

- *Service-orient. Architektur für virt. Dienstleistungen* (Sektion 4.3, S. 240 ff.) – Richtlinien für die informationstechnische Realisierung von E-Services zeigen die Integration virtueller Dienstleistungen in die IV-Infrastruktur von VDU. FRESCO SOM und SOA liefern konkrete Entwürfe Service-orient. VDL-Realisierung, deren Workflow-Basis die Anwendbarkeit begünstigt [ZLP04, ZL04a].

- *Service-orient. Entwicklungslebenszyklus für virt. Dienstleistungen* (Sektion 4.4, S. 279 ff.) – Ein generisches Entwicklungslebenszyklusmodell für Service-orient. E-Services zeigt deren Einsatzprinzipt im VDU-Kontext. Das konkrete FRESCO E-Service-Management-Vorgehensmodell bildet eine Anleitung zur Anwendung von FRESCO-E-Services zur Regelung der VDU-Produktion [ZBL03, ZLP04].

- *Modellgetriebene Entwicklungsmethodik für virt. Dienstleistungen* (Sektion 5.2.1, S. 313 ff.) – Eine generelle Strategie zur Realisierung des Entwicklungslebenszyklus für VDL zeigt das allgemein verwendbare Prinzip modellgetriebener E-Service-Entwicklung. Die konkrete FRESCO-Methodik spezifiziert Mechanismen zur modellgetriebenen Entwicklung von FRESCO-E-Services [ZLP04, ZP04].

- *Grid-basierte Plattformarchitektur für virt. Dienstleistungen* (Sektion 5.2.2, S. 347 ff.) – Ein Architekturprinzip für Service-orient. E-Service Grids gibt Komponenten von E-Service-Implementierungen und die Ressourcenverwaltung durch virtuelle Ausführungsumgebungen vor. Die konkrete FRESCO-Plattformarchitektur spezifiziert Anwendungs- und Plattformkomponenten zur Implementierung und Aggregation von FRESCO-E-Services (ZLP:SOA-04, [ZL04a]).

- *Prototypische Implementierung des technischen Rahmenwerks* (Sek. 5.3, S. 374 ff.) – Das FRESCO Toolkit implementiert die FRESCO-Plattformarchitektur als umfassendes Software-Rahmenwerk, in dem die Mechanismen der FRESCO-Methodik als Plattformkomponenten umgesetzt sind. Es liefert ein praktisches Studienobjekt, das die Erprobung und Evaluation der E-Service-Technik erlaubt [ZLP04, ZL04a, ZP04].

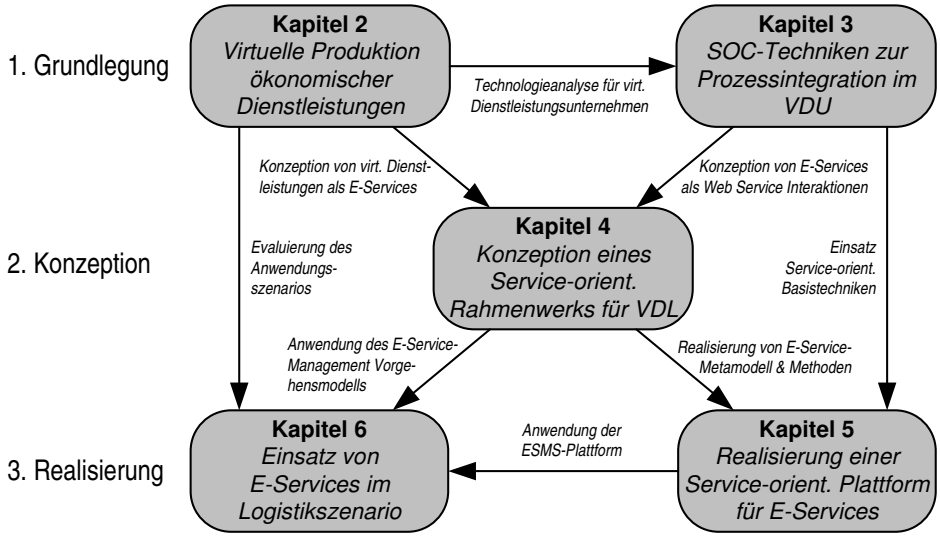

Abbildung 1.1.: Struktur der Arbeit

1.4. Gang der Untersuchungen

Um die zu untersuchenden Fragestellungen der vorliegenden Arbeit darzustellen, hat diese Einleitung zunächst den ökonomischen Kontext der Produktion von Dienstleistungen in virtuellen Unternehmen vorgestellt. Dabei zeigte sich die Produktionssteuerung durch die damit verbundenen Notwendigkeit zur Koordination kooperativer Tätigkeiten als aufwendig. Mit der Virtualisierung von Dienstleistungen durch E-Services wurde ein Konzept zur Effizienzsteigerung skizziert. Gleichzeitig wurde auf die Notwendigkeit von Erweiterungen aktueller prozessorientierter Interaktionsmodelle und Integrationstechniken zur Umsetzung von E-Services hingewiesen. Als Ziel der Arbeit ergab sich die Realisierung einer Systemunterstützung für virt. Dienstleistungen. Als Lösungsansatz wurde ein IT-basiertes Rahmenwerk mit konzeptionellen und technischen Anteilen skizziert.

Die zur Ausarbeitung des Lösungsansatzes notwendigen Untersuchungen gliedern sich in die drei Bereiche *Grundlegung*, *Konzeption* und *Realisierung*. Hierbei sind zunächst Basiskonzepte zu erarbeiten, dann das konzeptionelle Rahmenwerk auszuarbeiten und schließlich das technische Rahmenwerk zu realisieren und zu evaluieren. Abb. 1.1 zeigt die entsprechende Struktur der Arbeit mit Bereichen, den dazugehörigen Kapiteln sowie deren Zusammenhang im Überblick.

Die Grundlegung umfasst sowohl ökonomische als auch technische Aspekte. Der ökonomische Anwendungskontext ist hinsichtlich Organisation und Management der Dienstleistungsproduktion insbesondere in Form von VDU zu untersuchen. Daneben soll in der vorliegenden Arbeit das Szenario eines virtuellen Logistikdienstleisters (genannt *FreightMixer*) zur Anforderungsanalyse und als Evaluationsgrundlage her-

angezogen werden. Beide ökonomischen Anteile werden in Kapitel 2 dargestellt. Nachfolgend werden dann in Kapitel 3 Grundlagen zur Unterstützung von VDU durch IV dargestellt. In Hinblick auf die Zielsetzung der vorliegenden Arbeit werden dabei verschiedene grundsätzliche Techniken einer IV-Infrastruktur zur zwischenbetrieblichen Prozessintegration hervorgehoben. Aufbauend erfolgt dann eine Analyse von SOC-Techniken. Diese werden zunächst in Hinblick auf ihre generellen Potenziale eingeführt. Darüber hinaus werden spezifische Fähigkeiten zur zwischenbetrieblichen Interaktion (mittels WS-Koordination und -Komposition) und Prozessintegration hervorgehoben.

Die Konzeption wird in Kapitel 4 behandelt. Die Ausarbeitung der konzeptionellen Anteile startet dabei auf Anwendungsebene. Zunächst wird auf Basis der ökonomischen Grundlagen und Anforderungen die Virtualisierung von Dienstleistungen diskutiert. Diese wird dann auf Basis virt. Dienstleistungsprozesse exemplarisch vollzogen und als konzeptionelles Modell virt. Dienstleistungen festgehalten. Das VDL-Modell dient als Ausgangspunkt zur Konzeption einer Service-orient. IT-Repräsentation. Hierzu wird im Anschluss an eine generelle Diskussion VDL-spezifischer Anforderungen und SOC-basierter Potenziale das exemplarische FRESCO E-Service-Metamodell spezifiziert. Schließlich werden die Lebenszyklen virt. Unternehmen und Service-orient. AS gegenübergestellt und deren Integrationspotenziale diskutiert. Die Ergebnisse führen zur Formulierung eines abstrakten Vorgehensmodells zur Entwicklung von E-Services in VDU und dessen Konkretisierung durch das FRESCO E-Service-Management-Vorgehensmodell.

Bei der Realisierung erfolgt die Entwicklung eines praktischen Software-Systems sowie eine Bewertung der Ergebnisse. Dies beginnt in Kapitel 5 mit dem technischen Entwurf verschiedener Aspekte einer ESMS-Plattform. Dazu gehören Mechanismen zur Unterstützung des E-Service-Entwicklungslebenszyklus sowie eine E-Service-Ausführungsplattform zur automatischen Produktionssteuerung. Dann wird exemplarisch der in FRESCO erarbeitete ESMS-Entwurf sowie eine prototypische Implementierung der FRESCO ESMS-Konzepte durch das FRESCO Toolkit (FRESCO-TK) beschrieben. Dabei werden verschiedene konkrete SOC-Techniken und deren alternative Umsetzungen berücksichtigt. Die exemplarischen Betrachtungen finden ihren Abschluss in Kapitel 6, welches das FRESCO-TK auf das FreightMixer-Szenario anwendet und dadurch evaluiert.

Am Ende der Arbeit wird in Kapitel 7.1 eine Zusammenfassung und abschließende Bewertung der Ergebnisse vorgenommen. Ein Ausblick auf ergänzende und erweiternde Fragestellungen rundet die Arbeit ab.

2. Virtuelle Produktion ökonomischer Dienstleistungen

2.1. Zielsetzung

Das 2. Kapitel eröffnet den Grundlagenteil der vorliegenden Arbeit und bildet dort den ersten von zwei Abschnitten. Das Ziel des ersten Abschnitts besteht darin, den ökonomischen Anwendungskontext der vorliegenden Arbeit zu erfassen; auf diese Weise soll eine Basis für nachfolgende informationstechnische Konzepte bereitet werden. Konkret werden Grundlagen und aufbauende Themenfelder einer ökonomischen Dienstleistungssicht erläutert. Die Schwerpunkte liegen auf den in der vorliegenden Arbeit fokussierten Aspekten der Interaktivität zwischen Anwendungsakteuren, den dabei ablaufenden Prozessen und deren Kopplung zum Zweck einer kooperativen Dienstleistungserstellung. Dazu werden sowohl etablierte als auch innovative Modelle und Mechanismen der Volks- und vor allem Betriebswirtschaft herangezogen.

Im Anschluss an diese Zielsetzung soll zunächst in Sektion 2.2 der breite wirtschaftswissenschaftliche Hintergrund des Dienstleistungsbegriffs aufgearbeitet werden. Das Ziel besteht hier darin, aus der spezifischen Charakteristik ökonomischer Dienstleistungen Wirkungen auf die betriebliche Produktion und Prämissen für ein entsprechendes Produktionsmanagement abzuleiten. Motiviert durch den zentralen Aspekt des Dienstleistungsprozesses soll in Sektion 2.3 dann eine Untersuchung zur Anwendung moderner Prinzipien prozessorientierter und virtueller Organisation auf die Dienstleistungsproduktion folgen. Um die theoretischen Grundlagen schließlich im Rahmen der vorliegenden Arbeit nutzbar zu machen, soll Sektion 2.4 einen praktischen Kontext im Logistiksektor bereiten. Dazu gehört ein exemplarisches Szenario, das im weiteren Verlauf als Orientierungshilfe und Evaluationsgrundlage dienen kann.

2.2. Ökonomische Grundlagen der Dienstleistungsproduktion

Ganz allgemein werden in der ökonomischen Theorie des Wirtschaftskreislaufs [Has92, S. 1 ff.] *Aktivitäten* betrachtet, welche direkt (im Sinne von Produktion, Konsum) oder indirekt (im Sinne von Vermögens-, Kreditbildung) der Befriedigung von Bedürfnissen durch *Güter* dienen. Güter sind in diesem Sinne alle Mittel, die Bedürfnisse des Menschen direkt oder indirekt befriedigen. Ökonomische Aktivitäten bilden aus makroökonomischer Sichtweise vor allem die Ursache für die Durchführung von *Transaktionen*. Dabei tauschen handelnde *Wirtschaftssubjekte* auf Märkten spezifische

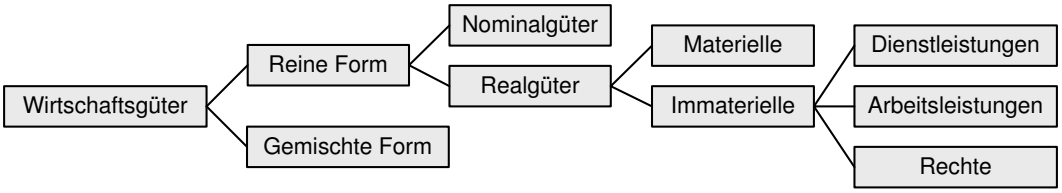

Abbildung 2.1.: Gütersystematik [Cor01, S. 20 f.]

Objekte aus. Wirtschaftssubjekte sind hier *Unternehmen* sowie private und öffentliche *Haushalte*. Bei den ausgetauschten Objekten handelt es sich entweder um *Produktionsgüter*, *Faktorleistungen* oder *Forderungen*. Unternehmen, die den Mittelpunkt mikroökonomischer Betrachtungen darstellen, produzieren Güter, indem sie andere Güter, nämlich Faktorleistungen und *Vorleistungen*, einsetzen und kombinieren. Als Faktorleistungen werden hierbei nicht produzierte Leistungen wie *Arbeitsleistung* oder Nutzung dauerhafter *Produktionsmittel* (z. B. Maschinen, Gebäude) bezeichnet. Vorleistungen gehen hingegen als nicht-dauerhafte Produktionsmittel (z. B. Rohstoffe, Energie) sowie als *Dienstleistungen* in die Produktion ein. Dienstleistungen werden dabei heute allgemein als Produktionsgüter aufgefasst.

Die ökonomische Theorie hat Dienstleistungen allerdings lange Zeit nicht als Objekte im wirtschaftswissenschaftlichen Sinn anerkannt. Dies ist darauf zurückzuführen, dass die Klassiker, so z. B. Adam Smith, nur materiellen Gegenständen die Fähigkeit zusprachen, den Bedarf von Wirtschaftssubjekten zu decken. Dienstleistungen wurden somit ohne bedarfsdeckende Funktion nicht als produktive Werte angesehen. Eine Änderung dieser Sichtweise resultierte erst aus der Güterdefinition von Jean Baptiste Say, die auf dem Kriterium der Nutzenstiftung basierte. Ein Dienst ist mit Aktivitäten bzw. Prozessen verbunden, die dadurch Nutzen stiften, dass sie die Änderung eines Konsumenten selbst, seines materiellen Besitzes oder seiner nicht materiellen Vermögenswerte hervorrufen. Auf Grund des offensichtlichen Nutzens von Diensten wurden diese fortan als Güter betrachtet und Dienstleistung galten somit als produktive Arbeit.[1] In Bezug auf die Einordnung in eine allgemeine *Gütersystematik* werden Dienstleistungen heute in der Regel zusammen mit Arbeitsleistungen und Rechten als reine, immaterielle Realgüter betrachtet (vgl. Abb. 2.1).

Aus makroökonomischer Sicht der Volkswirtschaftslehre sind Dienstleistungen vor allem bei der Erfassung von Strukturänderungen des Wirtschaftssystems (so genannte *intersektoraler Schwankungen*) von Bedeutung. Die *Drei-Sektoren-Theorie* unterscheidet dabei zwischen primärem, sekundärem sowie tertiärem Sektor. Auf Basis von Nachfrage- und Angebotshypothese wird argumentiert, dass langfristige Entwicklungsprozesse eine gesetzmäßige Verlagerung von Produktion und Beschäftigung vom primären *Agrarsektor* über den sekundären *Industriesektor* hin zum tertiären *Dienstleistungssektor* aufweisen. Verschiedene Ansätze gehen von verschie-

[1] Vgl. [Kör92, S. 95 f.].

denen Kriterien wie Einkommenselastizität der Nachfrage (Fischer) oder Dominanz des Produktionsfaktors (Wolfe) zur Charakterisierung der Sektoren aus. Die *tätigkeitsorientiert-institutionale* Sicht dieser Vorgehensweisen erfährt in der Literatur erhebliche Kritik auf Grund ihrer unscharfen Abgrenzung des Dienstleistungssektors, des hohen Abstraktionsniveaus sowie der institutionellen Gliederung [Cor01, S. 1 ff.]. Aus diesen Gründen liefert sie auch keine konkreten Anhaltspunkte im Sinne der untersuchten Fragestellung und wird nicht weiter betrachtet.

Eine grundsätzlich andere Perspektive findet sich in der mikroökonomischen Theorie der Betriebswirtschaftslehre. Auch hier wird dem Dienstleistungsbegriff erst seit relativ kurzer Zeit Aufmerksamkeit gewidmet.[2] Dabei sind solche Unternehmen von Interesse, bei denen eine Kombination produktiver Faktoren zur Erstellung von Dienstleistungen erfolgt. Betriebswirtschaftliche Betrachtungen des *Dienstleistungsmanagements* befassen sich mit den Problemstellungen von *DLU* in verschiedenen Funktionsbereichen wie Beschaffung, Produktion und Marketing. Da sich die allgemeine Betriebswirtschaftslehre auf Grund der traditionellen Bedeutung industrieller Fertigung schwerpunktmäßig auf die Sachgüterproduktion konzentriert, ist für eine solche Betrachtung eine *produktorientiert-funktionale* Sicht auf Dienstleistungen wesentlich, welche die besonderen *konstituierenden Merkmale* und deren Auswirkungen auf die Funktionsbereiche herausstellt.

Die vorliegende Arbeit nimmt die mikroökonomische Perspektive ein. Aus dieser Sicht wird angestrebt, die Steuerung und Kontrolle der arbeitsteiligen Produktionsprozesse von unternehmensbezogenen Dienstleistungen durch informationstechnische Mechanismen zu unterstützen. Als Grundlage betrachtet die folgende Untersektion zunächst einige generelle Ansätze zur Definition und Systematisierung von Dienstleistungen. Anschließend erfolgt ein Überblick über den Funktionsbereich des Produktionsmanagements.

2.2.1. Definition und Systematisierung von Dienstleistungen

Zur Erfassung des ökonomischen Dienstleistungsbegriffs wird im Folgenden zunächst ein Überblick grundsätzlicher wirtschaftswissenschaftlicher Definitionsansätze für Dienstleistungen gegeben. Dann werden die daraus hervorgehenden konstituierenden Dienstleistungsmerkmale zusammengefasst. Schließlich erfolgt die Einführung ausgewählter Systematisierungsansätze des Dienstleistungsspektrums.

2.2.1.1. Ansätze für Dienstleistungsdefinitionen

Obwohl die zunehmende wirtschaftliche Bedeutung der Dienstleistungsproduktion unumstritten ist, gibt es bislang keine allgemein akzeptierte Definition für den Begriff der Dienstleistung an sich.[3] Bei den verschiedenen existierenden Varianten können

[2] Vgl. z. B. Übersicht in [Mal97].
[3] Vgl. z. B. [Mal97].

die drei grundsätzlichen Definitionsansätze *negativer*, *enumerativer* und *expliziter* Definitionen unterschieden werden.[4] Negativdefinitionen charakterisieren Dienstleistungen dadurch, dass sie sie von materiellen Sachleistungen abgrenzen. Enumerative Definitionen umfassen Aufzählungen von konkreten Beispielen. Beide Varianten sind zur Analyse von Dienstleistungseigenschaften und deren Implikationen nur sehr bedingt geeignet. Wesentlich aufschlussreicher sind die produktorientiert-funktionalen Ansätze der expliziten Definitionen, die den Dienstleistungsbegriff durch seine wesentlichen konstituierenden Merkmale charakterisieren. Dabei kann zwischen *potenzialorientierten*, *prozessorientierten* und *ergebnisorientierten* Merkmalen unterschieden werden.

Aus potenzialorientierter Perspektive sind Dienstleistungen menschliche oder maschinelle *Leistungspotenziale* des Dienstleistungsanbieters, mit denen am Nachfrager oder an dessen Objekten gewollte Veränderungen vorgenommen oder Zustände erhalten werden sollen. Diese Sicht betont besonders den *immaterieller Charakter* von Dienstleistungen, die hier *Leistungsversprechen* eines gegenseitigen (Dienst-)Leistungsvertrags mit entsprechendem Kaufrisiko darstellen.

Die prozessorientierte Perspektive betont besonders die Zeitraumbezogenheit bzw. die zu vollziehende Tätigkeit. Aus dieser Perspektive sind Dienstleistungen *Prozesse* zur Bedarfsdeckung der Nachfrager. Da im Zuge des Dienstleistungsprozesses ein Kontakt zwischen dem Anbieter und dem Nachfrager oder dessen Objekten notwendig ist, hebt diese Betrachtungsweise den *Synchronisationsaspekt* hervor. Dabei ist zum einen ein gewisser Grad an zeitlicher und örtlicher Synchronisation notwendig, zum anderen wird eine Simultanität von Produktion und Absatz deutlich, die auch als *Uno-actu-Prinzip* bezeichnet wird. Daneben impliziert diese Sicht wiederum die Immaterialität, da Prozesse immaterieller Natur sind.

Aus ergebnisorientierter Perspektive wird eine Dienstleistung schließlich als immaterielles Resultat des Dienstleistungsprozesses verstanden. Es wird also die *nutzenstiftende Wirkung* hervorgehoben, die unter Einsatz externer Produktionsfaktoren des Nachfragers zu dessen Bedarfsdeckung produziert wurde und sich an diesem selbst oder seinen Objekten zeigt. Materielle Sachgüter wie Trägermedien von Informationen (z. B. Bauplan des Architekten), die oft am Ende des Dienstleistungsprozesses stehen, werden dabei meist nicht zur Charakterisierung herangezogen. Daher betrachtet die Definition die Resultate stets als immateriell und unterscheidet zwischen Erhaltung bzw. Wiederherstellung, Schaffung und Vernichtung von Merkmalen und deren Ausprägungen bei Gütern oder Personen.

Insgesamt wird empfohlen, die drei Ausprägungen der expliziten Dienstleistungsdefinition nicht als Alternativen, sondern als zusammenhängendes *Ablaufmodell* für Dienstleistungstransaktionen zu verstehen [Hen92, S. 21]. Die drei fokussierten Aspekte Potenzial, Prozess und Ergebnis implizieren danach eine Einteilung in *Vorkontakt-*, *Kontakt-* und *Nachkontaktphase* der Akteure (vgl. Abb. 2.2).

[4] Vgl. [Lan04].

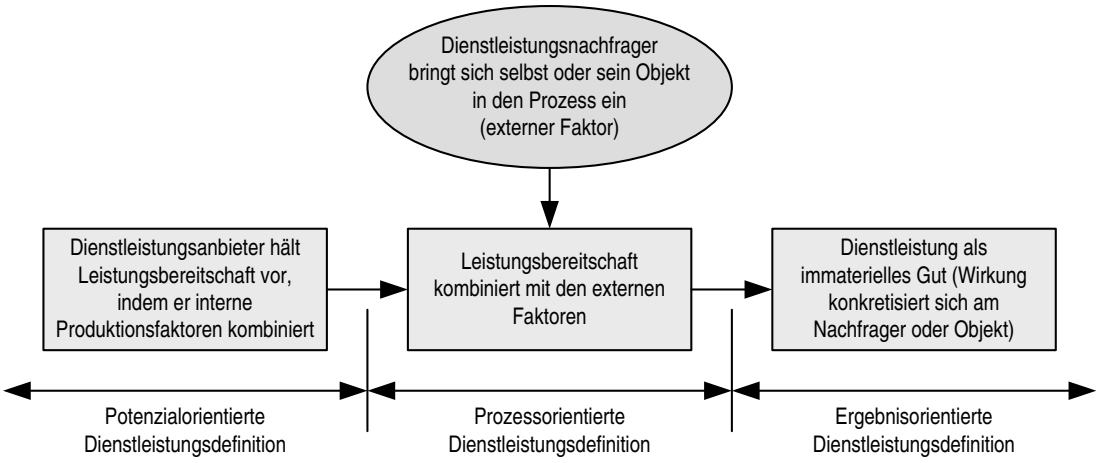

Abbildung 2.2.: Ansätze expliziter Dienstleistungsdefinitionen [Cor01, S. 26]

2.2.1.2. Konstituierende Dienstleistungsmerkmale

In der betriebswirtschaftlichen Literatur finden sich über 30 explizite Dienstleistungsdefinitionen.[5] Aus der Gesamtheit der Definitionen lassen sich zwei wesentliche Merkmale extrahieren: Zum einen wird die mehr oder weniger ausgeprägte *Immaterialität* von Dienstleistungen in vielen Definitionen in den Mittelpunkt gestellt. Zum anderen ist die Integration *externer Faktoren* des Nachfragers in den Produktionsprozess stets ein wichtiger Aspekt. Auf dieser Basis diskutiert die Literatur zahlreiche abgeleitete Merkmale zum Teil kontrovers.[6] Die wichtigsten Ergebnisse werden im Folgenden zusammengefasst:

- Alle expliziten Definitionsansätze betonen – jeweils aus ihrer spezifischen Perspektive des Potenzials, Prozesses oder Ergebnisses – den *immateriellen Charakter* von Dienstleistungen. Gerade dieser Aspekt erscheint jedoch fragwürdig und wird äußerst kontrovers diskutiert, denn praktisch keine reale Dienstleistung ist ohne stoffliche Komponente. Im Gegenteil ist die Erbringung von Dienstleistungen in den meisten Fällen mit materiellen Hilfsmitteln verbunden oder resultiert in materiellen Ergebnissen (z. B. Trägermedien). Da Dienstleistungen auch oft mit Sachgütern gebündelt werden (z. B. Kundendienst), liegt der Fokus heute eher auf den dadurch gebildeten Produkten, die sich als so genannte *Leistungsbündel* in ein Sachgut-Dienstleistungs-Kontinuum zwischen rein materiellen Gütern und rein immateriellen Diensten einordnen lassen. Aus diesem Grund wird heute eine *Lagerunfähigkeit* auf Grund von Immaterialität des Ergebnisses, die bei überwiegend immateriellen und meist personenbezogenen Produkten (z. B. Massage) zu beobachten ist, nicht mehr

[5] Vgl. Diskussion in [Mal97] oder Übersicht in [Ram03, S. 4 f.].
[6] Vgl. [Cor01, S. 27 f.], [Kör92, S. 101 ff.], [Ram03, S. 7 ff.].

generell propagiert. Als Gegenbeispiel muss das Ergebnis einer Beratung nicht sofort genutzt werden, wenn es zuvor schriftlich erfasst wurde. Darüber hinaus wird noch auf eine geistige Dimension von Immaterialität hingewiesen, die ausdrückt, dass Dienstleistungen *konzeptionell* schwer zu erfassen und zu vermitteln sind. Aus Sicht des Diensterbringers führt dies vor allem zu Problemen beim Marketing, wo oft mit kreativen Visualisierungen gearbeitet wird. Aus Kundensicht sind Dienste ohne direkte Erfahrung schwer einschätzbar oder vergleichbar.

- Die Notwendigkeit zur Integration des *externen Faktors* (auch *Kontaktzwang* genannt) bei Dienstleistungen ist weitestgehend unumstritten. Hintergrund ist die prozessorientierte Perspektive, nach der Dienstleistungen durch den Nutzen ihrer Tätigkeit gekennzeichnet sind. Um diesen Prozessnutzen dem Kunden zukommen zu lassen, muss die Produktion sowohl mit dem Absatz (Uno-actu-Prinzip) als auch mit einer Kundeninteraktion einhergehen. Genau wie bei der Immaterialität kann auch die *Integrativität* des externen Faktors mehr oder weniger stark ausgeprägt sein. Man kann für Dienstleistungen daher orthogonal zu dem Sachgut-Dienstleistungs-Kontinuum ein zweites Autonomie-Integrations-Kontinuum der *Kontaktintensität* zwischen integrativ-kontaktintensiv (z. B. Beratungsgespräch) und autonom-kontaktextensiv (z. B. Zustelldienst) einordnen. Die zum Teil aus der Kundeninteraktion abgeleitete Notwendigkeit zur zeitlichen und örtlichen *Synchronisation* von Erbringer und Nachfrager gilt daher nur für kontaktintensive – und meist personenbezogene – Dienstleistungen (z. B. medizinische Untersuchungen), wird aber besonders in Hinblick auf moderne Informations- und Kommunikationstechniken heute nicht mehr generell propagiert. Unabhängig von der Kontaktintensität muss der Dienstleistungserbringer auf Grund der Simultanität von Produktion und Absatz *Leistungspotenziale* vorhalten, deren Abruf zur Produktion sich erst unmittelbar bei Nachfrage ergibt. Die Individualität des externen Faktors durch verschiedene Kunden bedingt zudem die *Variabilität* der Dienstleistungserbringung. Dies führt zur *Heterogenität* von Dienstleistungen und erschwert deren Standardisierung.

Zum Abschluss dieser Ausführungen soll der ökonomischen Dienstleistungsbegriff nach [BM98, S. 5] als zusammenfassende Arbeitsdefinition festgehalten werden:

Definition 1 (Dienstleistung) *Dienstleistungen sind selbstständige und marktfähige Leistungen, die mit Bereitstellung und/oder dem Einsatz von Leistungsfähigkeiten verbunden sind. Interne und externe Faktoren werden im Rahmen des Leistungserstellungsprozesses kombiniert. Die Faktorkombination des Dienstleistungsanbieters wird mit dem Ziel eingesetzt, an den externen Faktoren – Menschen oder deren Objekten – nutzenstiftende Wirkungen zu erzielen.*

2.2.1.3. Systematisierung von Dienstleistungen

In Ergänzung zu den vorangegangenen Definitionen, die den generellen Dienstleistungsbegriff *charakterisieren*, soll nun die *Systematisierung* des Dienstleistungsspektrums betrachtet werden. Betriebswirtschaftliche Systematisierungsansätze zielen auf eine Erklärung realer Phänomene, indem sie die Menge dieser Phänomene (Ausgangsklasse von Elementen) in aussagekräftige Systeme von Teilmengen (Unterklassen) strukturieren. Als Basis der Strukturierung werden je nach Erklärungszweck verschiedene sachbezogene Merkmale gewählt. Eine Unterklasse kennzeichnet dann solche Elemente, die bestimmte charakteristische Erscheinungsformen dieser Merkmale aufweisen. Die ökonomische Literatur unterscheidet grundsätzlich zwischen *Klassifikation* und *Typologie*, wobei Klassifikationen sich auf genau ein Merkmal und Typologien sich auf mehrere Merkmale stützen. Alle Systematisierungen müssen jedoch drei generelle Kriterien erfüllen: sie müssen echt (mindestens zwei nichtleere Unterklassen), vollständig (jedes Element ist mindestens einer Unterklasse zugeordnet) und eindeutig (ein Element ist nicht mehr als einer Unterklasse zugeordnet) sein.

Spezifische Systematisierungen orientieren sich in der Regel an den Fragestellungen einer speziellen Untersuchung. In der vorliegenden Arbeit geht es speziell um die *Produktion* von Dienstleistungen sowie Dienstleistungen als *Produktionsfaktoren*.[7] Diesbezüglich wurden verschiedene Systematisierungen vorgeschlagen. Einige relativ generelle Klassifikationsschemata sind in Tab. 2.1 aufgeführt.[8]

Die Kategorien sind weitgehend selbsterklärend. Die beiden Klassifikationsmerkmale *Faktordominanz* und *Leistungsverwertung* sollen aber auf Grund ihrer Relevanz für die vorliegende Arbeit dennoch erläutert werden: Dienstleistungen können zunächst nach den zu ihrer Produktion benötigten Einsatzfaktoren unterschieden werden. Dazu können drei Klassifikationsansätze unterschieden werden: Die Unterscheidung *dispositiv* vs. *objektbezogen* zielt im Wesentlichen darauf ab, dass die eingebrachte Leistung einmal geistig-intellektueller und einmal handwerklicher Natur sein kann. Die Unterscheidung von *maschinenintensiv* vs. *personalintensiv* zielt hingegen auf die Bedeutung der Elementarfaktoren: Betriebsmittel bzw. objektbezogene menschliche Arbeit. Darüber hinaus kann noch eine Unterscheidung von *sachbezogen* vs. *personenbezogen* erfolgen. Hier steht der externe Faktor im Mittelpunkt an dem die Nutzenstiftung ansetzt: Dieser kann eine (materielle/immaterielle) Sache oder ein Lebewesen (Mensch/Tier) sein.[9]

Für die Merkmale der *Leistungsverwertung* werden ebenfalls verschiedene Klassifikationsansätze vorgeschlagen. Die Unterscheidung *direkt* vs. *indirekt* deutet die mögliche Anordnung von Dienstleistungen in Wertschöpfungsketten an. Indirekte

[7] Näheres zum Produktionsaspekt folgt in der nächsten Untersektion.

[8] Vgl. [Cor01, S. 32 ff.] für eine umfassendere Liste.

[9] Diese Merkmale der internen und externen Produktionsfaktoren vereinigt Corsten in einer produktionswirtschaftlich orientierten zweidimensionalen Typologie [Cor01, S. 43 ff.].

Merkmal	Erscheinungsformen
Einsatzfaktoren (Faktordominanz)	- dispositive Dienstleistungen - objektbezogene Dienstleistungen
	- maschinenintensive Dienstleistungen - personalintensive Dienstleistungen
	- sachbezogene Dienstleistungen - personenbezogene Dienstleistungen
Integrationsgrad des externen Faktors	- Dienstleistungen mit direkter Abhängigkeit - Dienstleistungen mit indirekter Abhängigkeit
Kontaktzwang	- gebundene (embodied) Dienstleistungen - ungebundene (disembodied) Dienstleistungen
Leistungsverwertung (Mittelbarkeit zum Konsum)	- direkte Dienstleistungen - indirekte Dienstleistungen
	- producer services - consumer services
	- konsumtive Dienstleistungen - investive Dienstleistungen
Individualität	- individuelle Dienstleistungen - standardisierte Dienstleistungen

Tabelle 2.1.: Auswahl gängiger Dienstleistungsklassifikationen

Dienstleistungen werden dabei – im Gegensatz zu direkten – nicht unmittelbar konsumiert, sondern gehen als Vorleistung in eine aufbauende Produktion ein (z. B. Transportversicherung). Während *Consumer* bzw. *Producer Services* lediglich synonyme Anglizismen darstellen, enthält die sehr gängige Trennung in *konsumtiv* und *investiv* einen zusätzlichen Aspekt: Mit investiven Dienstleistungen sind hier Investitionsgüter gemeint, die als Faktorleistung in eine aufbauende Produktion eingehen (z. B. strategische Unternehmensberatung).

2.2.2. Produktionsmanagement im Dienstleistungsunternehmen (DLU)

Generell nimmt sich die betriebswirtschaftliche Theorie der Perspektive von Wirtschaftsunternehmen an. Sie untersucht aus dieser Sicht sowohl die ökonomische Aktivität der Produktion von Gütern als auch die damit einhergehenden Transaktionen zur Beschaffung von Vor- und Faktorleistungen sowie zum Absatz der produzierten Güter. Eine Spezialisierung der Betrachtungen erfolgt dabei vor allem im Sinne betrieblicher Funktionsbereiche. Das Management von Beschaffung, Produktion, Marketing und Qualität bildet hierbei den üblichen Kern der allg. Betriebswirtschaftslehre (ABWL). Darüber hinaus werden Fragestellungen spezifischer Wirtschaftszweige in speziellen BWLs wie z. B. von Industrieunternehmen in der Industriebetriebslehre (IBL) untersucht.

Eine spezifische *Betriebswirtschaft der Dienstleistungsunternehmen (DLU)* ist erst seit Anfang der siebziger Jahre Gegenstand der Betrachtungen [Kuh03, S. 44]. Hierbei werden Erkenntnisse aus den BWLs verschiedener Dienstleistungsbranchen (z. B.

Banken, Versicherungen, Handel, Krankenhaus etc.) über die Funktionsbereiche der ABWL zusammengefasst. Ziel ist es, möglichst allgemeingültige Aussagen für beliebige DLU treffen zu können und diese nach Möglichkeit zu einem integrierten *Dienstleistungsmanagement* zu vereinen. Der wesentliche Aspekt einer BWL der DLU liegt in den Einflüssen der konstituierenden Dienstleistungsmerkmale: Immaterialität und Integrativität. Diese bedingen im Vergleich zur Sachgüterproduktion eine Anpassung der unternehmerischen Funktionsbereiche. In diesem Sinne wurde besonders das Management von Beschaffung, Produktion, Kosten, Marketing und Qualität für DLU betrachtet [Cor98, S. 119].

Für die Eigenschaften der Dienstleistungsproduktion sowie die daraus abzuleitenden Managementprinzipien finden sich in der Literatur verschiedene Ansätze[10]. Da diese Aspekte für die vorliegende Arbeit von besonderem Interesse sind, werden die relevanten Ergebnisse im Folgenden dargestellt. Hierzu wird erst das generelle Produktionsprinzip der Faktorkombination für den speziellen Fall von Dienstleistungen betrachtet. Es folgen dann Überlegungen zur Produktivität der Dienstleistungsproduktion und dem Gestaltungspotenzial zu deren Optimierung.

2.2.2.1. Aspekte der Dienstleistungsproduktion

Die Produktion stellt den Kern der unternehmerischen Tätigkeit dar und wird als werteschaffender Prozess der Konsumption gegenübergestellt. Inhaltlich wird sie in der Regel grob als *Faktorkombinationsprozess* charakterisiert. Eine Abgrenzung von den übrigen betrieblichen Funktionsbereichen geschieht durch Festlegung der Phase des Betriebsprozesses zwischen Beschaffung und Absatz. Diese sehr allgemeine Sicht wird durch Corsten äußerst treffend konkretisiert, der *Produktion* als „... die sich in betrieblichen Systemen vollziehende Bildung von Faktorkombinationen im Sinne einer Anwendung technischer oder konzeptioneller Verfahren zur Transformation der dem Betrieb zur Verfügung stehenden originären und derivativen Produktionsfaktoren in absetzbare Leistungen oder in derivative Produktionsfaktoren, die dann in weiteren Faktorkombinationsprozessen unmittelbar genutzt oder in absetzbare Leistungen transformiert werden, zur Erfüllung des Sachziels unter der Maßgabe der Formalzielerfüllung ...“ definiert [Cor01, S. 120]. Ein entsprechendes *Produktionssystem* lässt sich grob in einen *Input*, *Throughput* und *Output* Anteil strukturieren [Cor01, S. 120] (vgl. Abb. 2.3).

Der Input des Produktionssystems besteht aus so genannten *Produktionsfaktoren*. Dies sind Güter, die in einem Transformationsprozess miteinander wiederum zu Gütern kombiniert werden. Man unterscheidet *Potenzialfaktoren*, die dabei gebraucht werden, von *Repetierfaktoren*, die verbraucht werden. Per Definition sind Produktionsfaktoren erstens Güter, die zweitens ursächlich für die Entstehung des Outputs sind (causa efficiens), wobei es drittens zu einem Güterverzehr ihrer selbst kommt.

[10] Vgl. z. B. [Cor01], [Kuh03], [Mal97], [BM98].

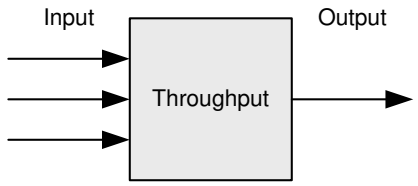

Abbildung 2.3.: Produktionssystem [Cor01, S. 120]

Die verschiedenen Faktoren der Produktion werden in so genannten *Produktionsfaktorsystemen* strukturiert die sich je nach Branche bzw. spezifischer Produktion zum Teil erheblich unterscheiden können.[11] In der durch Gutenberg eingeführten Urform für die Sachgüterproduktion wird zunächst zwischen elementaren Faktoren, die in den Produktionsprozess einfließen und dispositiven Faktoren, die den Prozess gestalten, unterschieden. Zu den Elementarfaktoren gehören Werkstoffe, Betriebsmittel und objektbezogene menschliche Arbeit. Die dispositiven Faktoren gliedern sich in originäre (Geschäftsleitung) sowie derivate (Planung, Organisation) Anteile. In späteren Ergänzungen sind zahlreiche weitere Differenzierungen vorgeschlagen worden. Ein aus informationstechnischer Sicht interessanter Aspekt ergibt sich z. B. aus der expliziten Betrachtung von *Information* als realem, immateriellem und elementarem Potenzialfaktor.

Bei der Dienstleistungsproduktion sind nun einige Besonderheiten dienstleistungsspezifischer Produktionsfaktoren zu beachten. Durch die Immaterialität von Dienstleistungen ist zunächst die Rolle der Werkstoffe umstritten. Es wird zwischen einem *Zwei-* und *Drei-Faktoren-Fall* unterschieden, wobei ersterer nur aus Arbeitskräften und Betriebsmitteln besteht und bei letzterem Werkstoffe berücksichtigt werden. Ob nun wie bei Verkehrsbetrieben keine, bei Handwerksbetrieben eindeutige, oder bei Banken und Versicherungen Zenralbankgelder als Werkstoffe berücksichtigt werden, so herrscht doch Einigkeit über die stets relativ geringe Bedeutung dieser Faktoren.

Wesentlich bedeutsamer zeigt sich das Merkmal der Integrativität von Dienstleistungen, das sich in einem so genannten *externen Produktionsfaktor* niederschlägt. Dieser externe Faktor wird vom Nachfrager – der dadurch in gewissem Umfang selbst zum Produzenten wird – aktiv oder passiv in den Produktionsprozess eingebracht (z. B. übergibt ein Versender seine Warensendung an ein Transportunternehmen). Die Erscheinungsformen sind sehr verschieden und können Menschen, Tiere sowie materielle (Sachgüter) oder immaterielle Objekte (Informationen) umfassen. Grundsätzlich besitzt der externe Faktor Gutscharakter (z. B. Warensendung als Sachgut), ist ursächlich für die Erstellung der Dienstleistung (ohne Warensendung kein Transport) und bedingt meist einen Güterverzehr in Form ungenutzter Zeit (keine Nutzungsmöglichkeit während des Transports). Folglich handelt es sich nach der Definition um einen echten Produktionsfaktor.

[11] Vgl. [Cor01, S. 121].

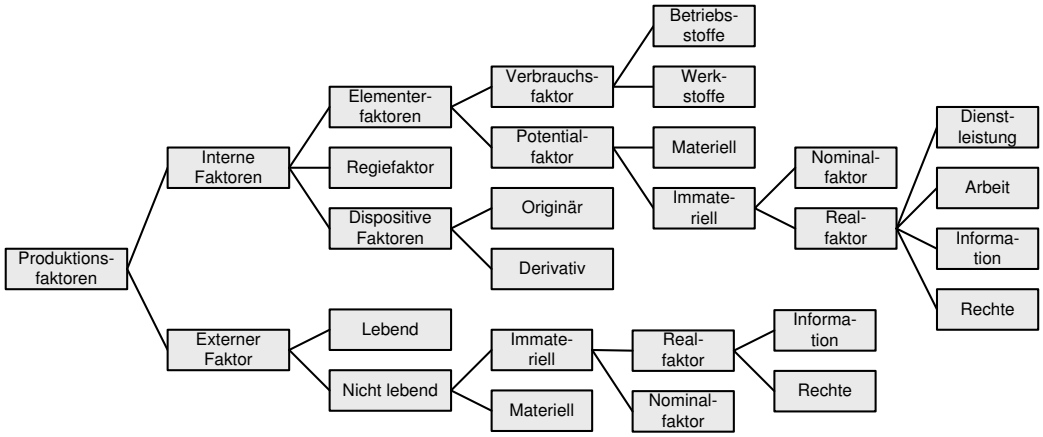

Abbildung 2.4.: Generischer Kern des Produkionsfaktorsystems nach Corsten

Da der externe Faktor direkt auf die Produktion einwirkt und sich zudem der autonomen Disponierbarkeit des DLU entzieht, ist eine gesonderte Berücksichtigung im Sinne des Produktionsmanagements nötig. Hier ist zunächst die Unterscheidung in *präsenzbedingte* und *informationsbedingte* Integration von Bedeutung (die jedoch meist einhergehen). Die präsenzbedingte Form meint die physikalische Beteiligung des externen Faktors (z. B. Transport der Warensendung). Die informationsbedingte Form zielt hingegen auf Anpassung des Produktionsprozesses an individuelle Anforderungen des Nachfragers (z. B. Transport mit/ohne Versicherung) und kann in ihrer verunsichernden Wirkung durch Standardisierung gedämpft werden. Des Weiteren kann durch Gestaltung des externen Produktionsfaktors – in gewissem Umfang – der Produktionsprozess des DLU optimiert werden, indem gewisse Aktivitäten zum Nachfrager verlagert werden. In diesem Zusammenhang spricht man auch vom *Aktivitätsgrad* der Beteiligten und unterstellt eine wechselseitige Substituierbarkeit der Aktivitäten von Anbieter und Nachfrager. In der Regel sind diesen Verlagerungen aber Grenzen in Form eines *Mindestaktivitätsgrads* des Anbieters gesetzt. Unterhalb dessen ist eine Dienstleistung nicht mehr nutzenstiftend.

In Hinblick auf die Besonderheiten der Dienstleistungsproduktion sind verschiedene dienstleistungsspezifische Produktionsfaktorsysteme sowohl für spezielle Dienstleistungsbranchen als auch in übergreifender Form entstanden. In [Cor01, S. 132 ff.] schlägt Corsten eine hybride Lösung vor, die einen allgemeingültigen Kern enthält und mit branchenspezifischen Modulen erweiterbar ist (vgl. Abb. 2.4).

Der Prozess der Faktorkombination ist bei der Dienstleistungsproduktion wesentlich durch den externen Produktionsfaktor geprägt. Grundsätzlich umfasst die Faktorkombination ein für die spezifische Produktion geeignetes technisch-konzeptionelles Verfahren, mit dem die Menge der eingehenden Produktionsfaktoren in eine gewünschte Leistung transformiert wird. Da für ein DLU der externe Produktionsfak-

tor nicht autonom disponierbar ist, wird die Faktorkombination in eine autonome *Vorkombination* und eine interaktive *Endkombination* unterteilt (vgl. Abb. 2.5). Es liegt also eine mehrstufige Produktion mit zwei generellen Phasen vor:

- Die Vorkombination bereitet die letztendliche Erstellung von Dienstleistungsprodukten in der Endkombination vor, indem sie dafür sowohl eine absolute (langfristige) *Kapazität* als auch eine unmittelbar abrufbare *Leistungsbereitschaft* aufbaut. Dies geschieht ohne Integration des externen Produktionsfaktors und ist daher durch das DLU autonom disponierbar. Zu beachten ist, dass das Vorhalten von Leistungsbereitschaft mit (fixen) Leerkosten in Zeiten submaximaler Beanspruchung einhergeht und daher sorgsam durch die dispositiven Produktionsfaktoren zu gestalten ist. Die Leistungsbereitschaft ist jedoch mit einem zusätzlichen *Bereitstellungsnutzen* für den Nachfrager verbunden. Dieser stellt über den reinen *Beanspruchungsnutzen* der Dienstleistung hinaus einen wichtigen Aspekt der Dienstqualität dar. Das zeigt sich etwa in einer größeren Sicherheit des Nachfragers in Bezug auf Verfügbarkeit.

- Die Endkombination stellt die zweite Stufe der Faktorkombination dar. Hier erfolgt die Erstellung von Dienstleistungsprodukten auf Basis von a) der durch die Vorkombination aufgebauten Leistungsbereitschaft, b) der Integration des externen Produktionsfaktors und c) weiterer interner Produktionsfaktoren. Bei *Auftragsproduktion* übt der externe Produktionsfaktor dabei in der Regel einen direkten Einfluss aus. Die Produktionsprozesse haben dann die präsenz- bzw. informationsbedingte Integration des Nachfragers zu leisten. Bei *kollektiven Dienstleistungen* für einen anonymen Markt (z. B. öffentlicher Nahverkehr) ist die Endkombination davon zwar frei, es geht damit jedoch auch ein erhöhtes Risiko der Leerkosten bei fehlender Inanspruchnahme einher.

Der Output des Produktionssystems ist für Dienstleistungen schwer zu fassen. Hierbei stellt Corsten in [Cor01, S. 141] fest: „... eine für alle Dienstleistungen gültige Output-Konzeption [...] dürfte beim derzeitigen Stand der Forschung kaum realisierbar sein ...“. Einige allgemeine Aussagen lassen sich jedoch treffen: Output ist aus produktionswirtschaftlicher Sicht das angestrebte Ergebnis einer Faktorkombination, das aus absatzwirtschaftlicher Sicht als Gut zur Bedarfsdeckung Dritter bestimmt ist. Allgemein kann man *zeitraumbezogene* und *zeitpunktbezogene* Produkte unterscheiden [Kuh03, S. 52]. Erstere bereiten Nutzen im Sinne eines *tätigkeitsbezogenen Leistungsbegriffs* direkt aus einem andauernden Prozess (z. B. Theateraufführung). Bei Letzteren liegt der Nutzen als Endergebnis mit Abschluss des Kombinationsprozesses vor (z. B. Haarschnitt). In jedem Fall stellt die Quantifizierung des Outputs ein zentrales Problem dar. Während dies bei manchen Dienstleistungen weniger Mühe macht (z. B. Transporte in Frachtvolumen/Kilometer), sind vor allem personenbezogene Dienstleistungen schwer zu bewerten (z. B. Beratungsgespräch). Oft wird die zeitliche Komponente als Bewertungsmaßstab herangezogen (z. B. Dauer

einer Massage, Arbeitsstunden eines Handwerkers) oder es werden pragmatische Kennzahlen verwendet (z. B. ärztliche Gebührenordnung). Der wirkliche Nutzen kann dadurch jedoch oft nur grob abgeschätzt werden.

2.2.2.2. Produktivitätsmanagement für Dienstleister

In Bezug auf das Produktionsmanagement von DLU finden sich in der Literatur vor allem Überlegungen zum Kapazitätsmanagement[12] und zum Produktivitätsmanagement[13]. Des Weiteren wurden auch die produktionsrelevanten Aspekte des Qualitätsmanagements[14] und die prozessorientierte Organisation[15] betrachtet.

Während das Kapazitätsmanagement vor allem auf die Gestaltung der Vorkombination hinausläuft, liegt der primäre Dispositionsaspekt des Produktivitätsmanagements in der Endkombination. Während dieser Endkombination vollzieht sich die wesentliche Interaktion von DLU und Konsumenten, die in der vorliegenden Arbeit von Interesse ist. Dabei liegt das wesentliche Werkzeug, wie sich zeigen wird, in der Gestaltung und Kontrolle der Interaktionsprozesse. Aus diesem Grund wird im Folgenden zunächst das Produktivitätsmanagement vertieft und auf seine Interaktionsaspekte untersucht. Aufbauende Überlegungen zur prozessorientierten Organisation folgen dann in der nächsten Untersektion.

Produktivität bezeichnet im Zusammenhang mit der Dienstleistungsproduktion ganz allgemein das Verhältnis zwischen dem Input eingebrachter Produktionsfaktoren und dem Output des Dienstleistungsprodukts (vgl. Abb. 2.5) bzw. nach Gutenberg die „Ergiebigkeit der betrieblichen Faktorkombination“. Das wertmäßige Verhältnis von Inputwerten zu Outputwerten wird als *Produktivität im weiteren Sinne* bezeichnet und ist gleichbedeutend mit der *Wirtschaftlichkeit* eines DLU. Dies ist zwar für Unternehmen von zentraler Bedeutung, doch spiegelt sich darin auch der externe Einfluss von Beschaffungs- und Absatzmärkten wider. Soll hingegen die reine betriebliche Leistungsfähigkeit fokussiert werden, muss eine güterwirtschaftliche Perspektive eingenommen werden. Bei dieser *Produktivität im engeren Sinne* werden die ein- und ausgehenden Gütermengen gegenübergestellt, was auch als *Technizität* bezeichnet wird.

Rein rechnerisch müssen Technizitäten für alle Produktionsfaktoren getrennt ermittelt werden, da hier in der Regel unterschiedliche Maßeinheiten vorliegen. Selbst dann sind die dadurch gewonnenen Zahlen mit äußerster Vorsicht zu interpretieren. Wie schon erwähnt, ist vor allem die quantitative Erfassung des Outputs bei Dienstleistungen äußerst schwierig und meist nur bedingt aussagekräftig. So spiegeln auch die Produktivitätszahlen nur die in den Maßeinheiten enthaltenen Aspekte wider und vernachlässigen die fehlenden Anteile. Die Relevanz wird deutlich, wenn

[12] Vgl. [Kuh03, S. 54 ff.], [Cor01, S. 165 ff.].
[13] Vgl. [Kuh03, S. 61 f.], [Cor01, S. 146 ff.].
[14] Vgl. [Cor01, S. 292 ff.].
[15] Vgl. [Kuh03, S. 58 ff.], [Har03, S. 23 ff.].

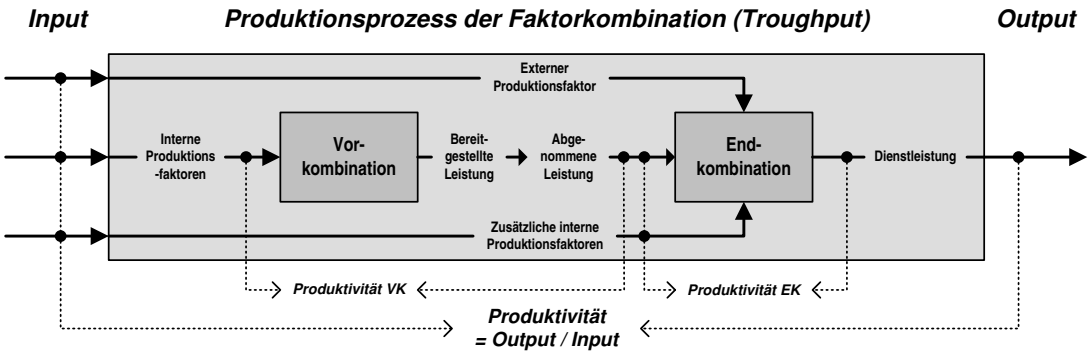

Abbildung 2.5.: Prinzip der Dienstleistungsproduktion

man bedenkt, dass die Maßeinheiten für den Output in der Regel nicht dessen Qualität berücksichtigen. So ist es z. B. fraglich, ob die an der steigenden Zahl der pro Jahr unterrichteten Schüler festgemachte Produktivitätssteigerung einer Schule positiv interpretiert werden soll. Zieht man diesen Umstand jedoch in Betracht und beachtet auch die Festlegung von sekundären Einflussgrößen (Organisation des Produktionsprozesses, technischer Fortschritt etc.), so lässt sich durch Vergleich der Produktivitätskennzahlen mit geeigneten Vergleichswerten eine wertvolle Indikatorfunktion für die betriebliche Leistungsfähigkeit gewinnen.

Ein besonderer Aspekt bei der *Produktivitätsanalyse* in DLU ist durch die Mehrstufigkeit der Produktion gegeben. Die Aufteilung in Vor- und Endkombination geht analog mit einer entsprechenden Trennung der Produktivitätskennzahlen einher (vgl. Abb. 2.5). Dabei lassen sich verschiedene, für die Dienstleistungsproduktivität spezifische, Aspekte feststellen:

- Für die Produktivität der Vorkombination gilt zunächst, dass sie von der autonomen Gestaltung der entsprechenden Produktionsprozesse durch das Unternehmen beeinflusst wird. Darüber hinaus ist jedoch zu bedenken, dass nicht die gesamte erzeugte Leistungsbereitschaft als Output gewertet werden kann, sondern nur der wirklich in Anspruch genommene Anteil. Die so genannte *nutzgradinduzierte Produktivitätsdifferenz* deutet auf Schwächen (z. B. qualitative) des Dienstleistungsangebots hin und kann als Ansatzpunkt zur Optimierung dienen.

- Auch die Produktivität der Endkombination ist nicht allein von den Handlungen des Unternehmens abhängig. Hier besteht ein unmittelbarer Einfluss des Konsumenten, der vom externen Produktionsfaktor ausgeht. Schwankt dieser Input, dann ändert sich folglich auch die Produktivität. Üblicherweise wird der externe Produktionsfaktor jedoch nicht *direkt* in die Produktivitätskennzahl aufgenommen. Diese stellt nur interne Produktionsfaktoren und abgenommene Leistungsbereitschaft dem Output gegenüber. Dadurch sind *Externalisierung*

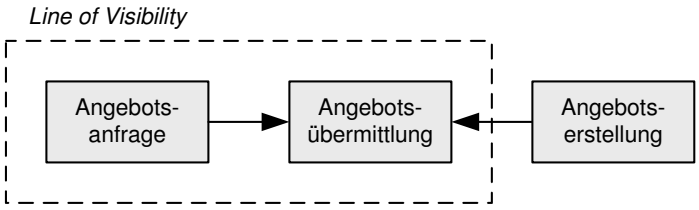

Abbildung 2.6.: Blueprint eines Dienstleitungsproduktionsprozesses

und *Internalisierung* von Aktivitäten zum bzw. vom Konsumenten produktivitätswirksam. Durch diesen Effekt wird die Gestaltung der Interaktionsprozesse zwischen Unternehmen und Konsument während der Dienstleistungsproduktion zum zentralen Einflussfaktor des Produktivitätsmanagements.

Zur Ableitung gestalterischer Maßnahmen sind die bisher betrachteten Kennzahlen allesamt zu grob. Die Produktivitätskennzahlen der gesamten Vor- und Endkombination lassen keine Rückschlüsse auf deren Zustandekommen im Zuge des komplexen Produktionsprozesses zu. Um hier zu konkreten Ergebnissen zu kommen, tritt die Methode der *Prozessanalyse* in den Vordergrund. Sie erlaubt es, die Abläufe bis auf atomare Tätigkeiten herunterzubrechen und letztendlich feingranulare Produktivitätskennzahlen zuzuordnen. Auf dieser Ebene können daraus dann konkrete Prozessoptimierungen abgeleitet werden. Eine dafür häufig verwendete Modellierungsmethode ist das *Blueprinting* (vgl. Abb. 2.6). Hierbei wird ein Flussdiagramm des Prozesses erstellt, bei dem die Aktivitäten mit Kundenkontakt durch eine Sichtbarkeitslinie (*Line of Visibility*) gekennzeichnet sind.[16]

Das Produktivitätsmanagement zielt darauf ab, die Produktivität der Dienstleistungsproduktion zu optimieren, d. h. sie zu halten und nach Möglichkeit zu erhöhen. Dabei wurde in jüngerer Vergangenheit besonders der so genannte *Kundenkontakt-Ansatz* propagiert.[17] Dieser nutzt die oben beschriebene Wirkung des externen Produktionsfaktors auf die Produktivität aus und macht den Interaktionsprozess zwischen Anbieter und Nachfrager zum zentralen Gestaltungsbereich. Für das Produktivitätsmanagement bieten sich dann im Wesentlichen drei zusammenhängende Ansatzpunkte:

- Externer Produktionsfaktor
- Ablauforganisatorische Maßnahmen
- Einsatz technischer Hilfsmittel

[16] Siehe auch Sektion 2.3.1.2.
[17] Vgl. [Cor01, S.159 ff.].

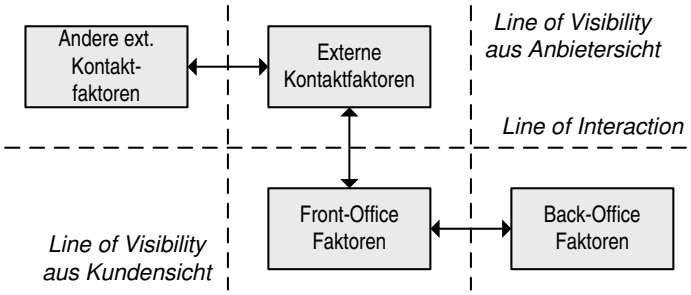

Abbildung 2.7.: Leistungserstellungsfaktoren [Cor01, S. 164]

Zentraler Ansatzpunkt ist der *externe Produktionsfaktor*. Das Ziel ist eine möglichst optimale Verteilung der Produktionsaufgaben zwischen Anbieter und Nachfrager. Dabei ist zu bedenken, dass eine Externalisierung von Aktivitäten zwar den Input des Unternehmens reduziert, sich aber auch die Unsicherheit erhöht, da Steuerungs- und Kontrollmöglichkeiten des Prozesses tendenziell abnehmen. Bei der Internalisierung kehrt sich dieser Effekt um und so ist die optimale Aufteilung vom Einzelfall abhängig.

Ein zweiter wesentlicher Ansatzpunkt liegt in *ablauforganisatorischen Maßnahmen*. Die oben angesprochene Prozessanalyse ist die Basis für eine mögliche Umstrukturierung des Produktionsprozesses hinsichtlich Menge, Art und Abfolge von Aktivitäten. Es ist dabei zweckmäßig, zwischen *kundennahen* und *kundenfernen* Bereichen zu unterscheiden. Hinsichtlich der Gestaltungspotenziale mittels Ex- und Internalisierung kann dabei die im Blueprinting verwendete Sichtbarkeitslinie zur Trennung so genannter *Back* bzw. *Front Office Faktoren* durch eine Interaktionslinie (*Line of Interaction*) erweitert werden (vgl. Abb. 2.7). Diese markiert zusätzlich den direkten Übergang zwischen Aufgaben die vom Anbieter und Nachfrager getrennt verrichtet werden (so genannte *Kontaktfaktoren*). Mit einem solchen Modell lassen sich somit die ganzheitlichen Produktionsprozesse erfassen und optimieren.

Der Einsatz so genannter *sachlich-technischer Hilfsmittel* (z. B. IV-Technik) soll schließlich zum einen die Interaktivität zwischen Anbieter und Nachfrager unterstützen und zum anderen den Nachfrager zur Nutzung technischer Hilfsmittel motivieren. Ein positiver Effekt derartiger Mittel liegt in der Konkretisierung und Offenlegung der Produktionsprozesse, wodurch der Nachfrager an diesen tendenziell besser mitwirken kann. Dies wirkt sich potenziell positiv auf die Unsicherheit bei Externalisierung aus.

Diese Ansatzpunkte – und darunter insbesondere der letzte – sind für die vorliegende Arbeit von zentraler Bedeutung. Es lässt sich daraus ableiten, dass eine Automatisierung der Interaktionsprozesse bei der Dienstleistungsproduktion zu einer Produktivitätssteigerung des DLU führt. Im Folgenden werden diese schon generell eingeführten Prozesse weiter konkretisiert.

2.3. Prozess-Organisation virt. Dienstleistungsunternehmen (VDU)

In Sektion 2.2 wurde dargestellt, dass Dienstleistungen im Wesentlichen durch immaterielle, interaktive Leistunsprozesse gekennzeichnet sind. Sie gehen aus dem Potenzial von DLU hervor und sind zunächst nur durch Leistungsversprechen gegeben. Die Leistungserstellung erfolgt dann im überwiegenden Fall nachfrageinduziert als Auftragsfertigung. Sie manifestiert sich in einem Leistungsprozess, der meist schon das Dienstleistungsprodukt darstellt. Die Wertschöpfung basiert auf dem Nutzeneffekt dieses Vorgangs für den Kunden. Der Leistungsprozess erfolgt dazu unter interaktiver Einbeziehung des externen Produktionsfaktors. Mit dem Ende des Leistungsprozesses endet auch die Dienstleistung. Es bleibt als Ergebnis ihre Nutzenwirkung.

In Bezug auf die spezifischen Merkmale der Dienstleistungsproduktion stellt sich die Frage nach einer passenden *Organisationsform* für DLU. Generell kennzeichnet der Organisationsbegriff [SZ99] ein zielgerichtetes Handeln. Der Rahmen ist dabei die Organisation im *institutionellen* Sinne (z. B. das Unternehmen). Die Organisation im *instrumentalen* Sinne gibt innerhalb dieses Rahmens die Struktur von Funktionen und Prozessen vor. Organisation im *funktionalen* Sinne (auch *Organisieren*) beinhaltet die Gestaltung der Organisationsstrukturen. Die Aufgabe der Organisationsgestaltung besteht darin, die Gesamtaufgabe einer Organisation im Rahmen der *Arbeitsteilung* auf die Organisationsmitglieder zu verteilen und ihre *Koordination* sicherzustellen.

Nach Picot und Neuburger können aus der Dienstleistungscharakteristik drei *Anforderungen* für die instrumentale Organisation von DLU abgeleitet werden:[18]

- Kundenorientierte Konfiguration eines Dienstleistungsprodukts als weitgehend vertrauensbasiertes Zukunftsversprechen
- Unterstützung eines hohen Grades an Immaterialität bei der Leistungserstellung
- Unterstützung eines hohen Grades an Interaktion mit dem Kunden

Darüber hinaus ist die Organisation (im instrumentalen Sinne) von Unternehmen ganz allgemein so zu gestalten, dass die Leistung effektiv und effizient [SZ99, S. 5] bzw. bedarfs- und wettbewerbsgerecht erbracht werden kann. In Bezug auf den Kundenbedarf sind heute als wichtigste Anforderungen *Funktionalität* und *Flexibilität* sowie Ökologie und Image der Leistung zu nennen [Brü99, S. 9 f.]. Die Funktion soll in der Regel umfangreich sein und verschiedene Aspekte kombinieren. Gleichzeitig sollen die spezifischen Bedürfnisse des Kunden Berücksichtigung finden. Die Wettbewerbsfähigkeit des Unternehmens ist dabei wesentlich vom Preis-Leistungsverhältnis abhängig [Brü99, S. 11]. Zur Verbesserung sind die Kosten zu senken oder die Leistung zu erhöhen; optimalerweise simultan. Zudem müssen verschiedene

[18] Vgl. [PN98, S. 517].

Wettbewerbsfaktoren beachtet werden: eine befriedigende Menge, Qualität, Zeit und Flexibilität der Leistungserbringung ist erfolgsentscheidend.

In der Managementforschung besteht weitgehende Einigkeit darüber, dass die klassische *funktionale Organisation* von Unternehmen nicht mehr mit den Anforderungen an moderne Leistungserstellung korrespondiert. Dies gilt ganz allgemein[19] und insbesondere für Dienstleistungen[20]. Funktionale Unternehmen sind zu Zeiten und in Anbetracht industrieller Massenfertigung entstanden. Sie zeichnen sich u. a. durch ihre Expansionsfähigkeit sowie die besondere Eignung zur Kontrolle und Planung aus. Die Verdichtung und Aufbereitung von Informationen von den verschiedenen Stellen hin zu einer zentralen Spitze ist äußerst effektiv. Durch Verfügbarkeit moderner IV-Technik ist dies jedoch nicht mehr zwingend notwendig [Brü99, S. 14 ff.]. Die Prozesse funktionaler Unternehmen sind zudem kompliziert und lassen sich kaum direkt steuern. Entscheidungen werden relativ kundenfern getroffen und durch einen meist aufwendigen organisatorischen Mittelbau geleitet. Dies führt zu erheblichen Nachteilen im zeitlichen Wettbewerb.

Infolgedessen trat mit dem Aufkommen neuer Wettbewerbsfaktoren (insbesondere mit dem *Zeitwettbewerb*) die *Ablauforganisation* in den Vordergrund. Hierbei wird insbesondere die Gestaltung von Prozessketten in den Vordergrund gestellt.[21] Bei dieser *prozessorientierten Organisation* erfolgt die Strukturierung von Unternehmen nicht mehr in Bezug auf Funktionen, sondern bezüglich wertschöpfender Prozesse.[22] Die unmittelbare Gestaltung orientiert sich dabei primär am Kunden: bei ihm beginnen und enden stets alle Wertschöpfungsprozesse. Der Gesamtprozess wird dann in eigenständige zusammenhängende Teilprozesse strukturiert, die jeweils einer dezentralen Stelle verantwortlich zugeteilt sind (so genannte *Process Owner*). Damit werden de facto Entscheidungskompetenzen dezentralisiert. Diese konzentrieren sich stattdessen am Ort der Problemlösung. Das Ergebnis sind klar definierte, mess- und steuerbare Prozesse. Zudem ergibt sich insgesamt eine Verschlankung der Organisation. Die Dezentralisierung von Kompetenzen hat jedoch auch Nachteile: Zum einen ist die Kompetenzbildung hier aufwendiger. Zum anderen können sich organisatorische Redundanzen ergeben. Für DLU ist die Prozessorientierung jedoch ein entscheidender Fortschritt, denn ihre spezifischen Organisationsanforderungen werden hier schon besser berücksichtigt: Die Organisation richtet sich am immateriellen Dienstleistungsprozess aus und optimiert diesen in Bezug auf die Kundeninteraktion.[23]

Im Zuge des verschärften internationalen Wettbewerbs und des zusätzlichen Wettbewerbsfaktors der Flexibilität scheint auch die prozessorientierte Organisation heute nicht mehr ausreichend. Aktuelle Ansätze der *Netzwerkorganisation* kombinieren daher die Vorteile funktionaler und prozessorientierter Organisationsformen [Brü99,

[19] Vgl. [Brü99, S. 14 ff.], [KRR97, S. 19 ff.].
[20] Vgl. [PN98, S. 517], [Har03, S. 24 f.].
[21] Vgl. z. B. [HC93], [OF96], [GS01], [SZ99].
[22] So genannte *Wertschöpfungsketten*, siehe 2.3.1.1.
[23] Vgl. [Har03].

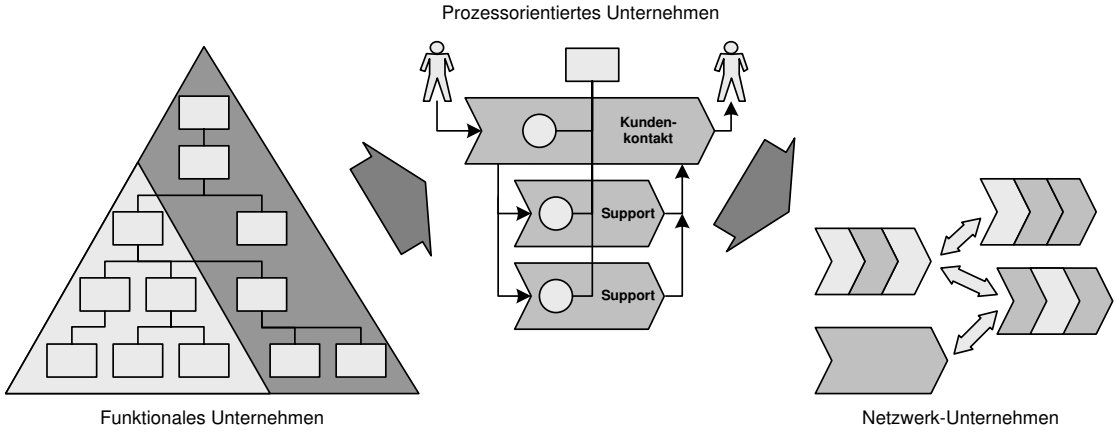

Abbildung 2.8.: Evolution betrieblicher Organisationsformen (nach [Brü99])

S. 16] (vgl. Abb. 2.8). Das Ergebnis sind netzwerkartige Strukturen, bei denen relativ autonome und spezialisierte Organisationseinheiten effizient miteinander verknüpft werden. Dies kann unternehmensin- oder -extern geschehen. Die Verknüpfung wird dann je nach Ausprägung durch hybride Mechanismen zwischen Markt und Hierarchie erreicht. Hintergrund für die in der Regel dauerhaft ausgelegten Verbindungen ist das gemeinsame Ziel erhöhter Wirtschaftlichkeit [Sie99b]. Dies resultiert aus Flexibilitäts- und Synergiepotenzialen sowie Risikoteilung und Transaktionskostenvorteilen [KRR97, S. 206 ff.]. Es besteht aber auch die Gefahr, dass sich die genannten Vorteile bei unangemessener Gestaltung, Lenkung und Entwicklung des Netzwerks genau ins Gegenteil umkehren können [KRR97, S. 226 ff.].

Ein zusätzlicher Aspekt der Netzwerk-Organisation erwächst aus der parallelen Tendenz zur *Virtualisierung* im ökonomischen Kontext.[24] Dabei werden Objekte oder Aktivitäten nicht real, sondern nur noch von ihrer Möglichkeit her betrachtet. Zur Veranschaulichung mag das Beispiel des virtuellen Arbeitsspeichers aus der Informatik dienen. Dieser Speicher wird als Ganzes ausgewiesen, ist aber nur zum Teil physikalisch vorhanden. Im Bedarfsfall besteht die Möglichkeit zur Ausweitung durch Sekundärspeicher. Im wirtschaftlichen Umfeld wird nun u. a. versucht, Organisationsformen zu virtualisieren. *Virtuelle Wertschöpfungsketten* und insbesondere *virtuelle Unternehmen* sind Organisationsformen, die nur von der Möglichkeit her vorhanden sind. Sie basieren auf einer flexiblen, problem- und aufgabenbezogenen Verknüpfung modularer prozessorientierter Organisationseinheiten im Sinne eines dynamischen Netzwerks [Syd92]. Eine derartige Virtualisierung der Organisation bietet sich auch im Zusammenhang mit Dienstleistungen an [PN98]. Insbesondere wird hierbei schließlich auch die Forderung nach flexibler Konfiguration kundenspezifischer Leistung im Vertrauenskontext des Netzwerks berücksichtigt.

[24] Vgl. z. B. [Sch94], [Brü99, S. 37 ff.], [KRR97, S. 3 ff.].

Die Betrachtung unternehmerischer Organisationsformen zeigt Prozess- und (virtuelle) Netzwerkorganisation als Ergebnisse einer wettbewerbsgetriebenen sukzessiven Entwicklung [Brü99, S. 14 ff.]. Dieser *evolutionäre* Ansatz [KRR97, S. 11 ff.] wird im Folgenden durch eine *situative* Betrachtung [KRR97, S. 9 ff.] ergänzt. Dabei werden dann Charakteristika und Merkmale fokussiert; von deren Entstehungsgeschichte wird jedoch weitgehend abstrahiert. Der Schwerpunkt liegt auf Prozess- und Netzwerkorganisation insbesondere von DLU. Zunächst betrachtet Sektion 2.3.1 grundlegende Aspekte der Prozessorganisation und -gestaltung der Dienstleistungsproduktion. Danach führt Sektion 2.3.2 virtuelle Unternehmen als organisatorische Netzwerke ein und wendet das Konzept auf den Dienstleistungssektor an. Schließlich werden die organisatorischen Betrachtungen auf das Produktionsmanagement übertragen. Sektion 2.3.1 führt dazu Produktionsnetzwerke ein und untersucht sie in Hinsicht auf deren Managementaufgaben sowie insbesondere in Hinsicht auf deren Koordinationsaspekt.

2.3.1. Prozessorganisation der Dienstleistungsproduktion

Organisatorische Leistungserstellung wird in einzelnen, miteinander verbundenen Arbeitsschritten durchgeführt [Har03, S. 23]. Die Durchführung aller notwendigen Arbeitsschritte durch die Organisationsmitglieder ergibt eine *Prozesskette*, die auch als *arbeitsteiliger Produktionsprozess* bezeichnet wird. Das hierbei grundlegende Konzept der Arbeitsteilung wurde bereits im 19. Jahrhundert durch Babbage als vorteilhaft erkannt. Die Optimierung (handwerklicher) *Arbeitsprozesse* wurde dann in Taylors Prinzip des *Scientific Management* [Tay11] zu einer der klassischen Organisationstheorien. Seit den Erfolgen bei der industriellen Massenproduktion (u. a. durch Ford) wird die organisatorische Relevanz arbeitsteiliger Produktionsprozesse nicht mehr bezweifelt.

Im deutschsprachigen Raum ist der Organisationsbegriff heute stark durch die instrumentale Sicht der betriebswirtschaftlichen Organisationslehre u. a. nach Nordsieck, Kosiol und Grochla geprägt. Grundlage ist die Zweiteilung in Beziehungs- und Ablauflehre [Nor34]. Die Beziehungslehre untersucht die *Aufbauorganisation*, d. h. die Gliederung von Unternehmen in organisatorische Teileinheiten mit Aufgaben und Kompetenzen sowie deren Koordination. Die Ablauflehre befasst sich hingegen mit der *Ablauforganisation*, d. h. der eigentlichen Arbeit – als Ausübung betrieblicher Funktionen innerhalb der Teileinheiten – sowie Zuteilung von Informationen und Sachmitteln. Zur organisatorischen Gestaltung dient im Wesentlichen das Analyse-Synthese-Konzept [Kos62]. Grundlage ist die Aufgabe (im Sinne von Handlungsziel) eines Unternehmens. Die Aufgabenanalyse beinhaltet zunächst die geordnete Zerlegung der Oberaufgabe in Teilaufgaben [SZ99, S. 40]. Die anschließende Aufgabesynthese beinhaltet die Zusammenfassung von Teilaufgaben zu verteilungsfähigen Aufgabenkomplexen [SZ99, S. 41]. Hierbei werden die verschiedenen Aufgaben in Bezug auf ihre Merkmale zentralisiert bzw. dezentralisiert. Das Ergebnis ist eine

spezifische Form der Arbeitsteilung durch Stellen- und Abteilungsbildung. Nach der Aufbauorganisation wird die Ablauforganisation lediglich zweitrangig behandelt: Die Arbeitsanalyse setzt bei den Elementaraufgaben an. Diese werden als Arbeitsteile höchster Ordnung (Arbeitsgänge) aufgefasst. Arbeitsgänge werden mehrfach bis auf Arbeitsteile niedrigster Ordnung (Gangelemente) zerlegt. Bei der Arbeitssynthese erfolgt schließlich die Gestaltung der Arbeitsprozesse in Bezug auf personale, temporale und lokale Aspekte, d. h. Bildung stellenbezogener Arbeitsgänge sowie deren stellenübergreifende Anordnung in Bezug auf Zeit und Raum.

Die betriebliche Organisationslehre ist sowohl wegen ihrer Trennung von Aufbau- und Ablauforganisation als auch wegen der Vorgehensweise bei der Organisationsgestaltung starker Kritik ausgesetzt. Die Aufbauorganisation beinhaltet mit Arbeitsteilung und Koordination schon die wesentlichen organisatorischen Aspekte, ohne die Abläufe einzubeziehen. Entsprechend setzt die Organisationsgestaltung nach dem Analyse-Synthese-Konzept ihren Schwerpunkt auf die Aufbauorganisation und vernachlässigt den Einfluss der Abläufe. Man spricht auch von *funktionaler Organisation*. Ausgehend von den bei der Aufgabenanalyse gebildeten Teilaufgaben werden bei der Arbeitssynthese funktional spezialisierte Stellen und Abteilungen gebildet. Dann erfolgt noch vor der Arbeitsanalyse eine koordinative Hierarchiebildung durch formale Über- und Unterordnungen der Stellen. Ablauforganisatorische Betrachtungen dienen danach lediglich noch zur Optimierung von Auslastung und Durchlaufzeiten der Funktionen. Bei der funktionalen Organisation wird vernachlässigt, dass für ein Unternehmen letztendlich nicht die Abwicklung einzelner Aufgaben, sondern kompletter Transaktionen wichtig ist. Durch die funktionale Spezialisierung beziehen die Abläufe von Transaktionen (bzw. *Geschäftsprozessen*) Funktionen mehrerer Stellen ein. Dies macht nicht-wertschöpfende Funktionen (Dysfunktionen) zur Kontrolle und Abstimmung nötig.

Neuere Formen der *prozessorientierten Organisation* stellen den ganzheitlichen Geschäftsprozess in den Mittelpunkt.[25] Im Vergleich zur funktionalen Organisationsgestaltung ergibt sich hier ein Verbesserungspotenzial dadurch, *„... Stellen in erster Linie nicht nach dem Anforderungsprofil einer hierarchisch orientierten Aufgabenteilung zu bilden, sondern nach einer durch die Wertschöpfungskette vorgegebenen Prozessnotwendigkeit“* [GSV+94, S. 5]. Es ergibt sich eine vorgangsbezogene Aufbauorganisation zur Optimierung von Geschäftsprozessen. Für die verschiedenen prozessorientierten Organisationskonzepte lassen sich eine Reihe gemeinsamer Merkmale feststellen [SZ99, S. 46]: Im Vordergrund steht jeweils die Bildung von prozess- und ablauforientierten Strukturen. Von diesen geht die Bildung überschaubarer und transparenter Organisationseinheiten aus. Das Resultat sind einfach zu koordinierende Verantwortungsbereiche mit ganzheitlichen Aufgabenkomplexen und entsprechenden Kompetenzen. Das Ziel liegt in der Minimierung von Schnittstellen entlang betrieblicher Prozesse.

[25] Vgl. z. B. [GSV+94], [HC93].

Prozessorientierung stellt sich schon aus rein organisationstheoretischen Gesichtspunkten vorteilhaft dar. Darüber hinaus betont Harms in [Har03, S. 24 f.] die Bedeutsamkeit des Konzepts für zwei spezifische Unternehmensmerkmale, die in der vorliegenden Arbeit von großer Bedeutung sind: Netzwerkorganisation und Dienstleistungsproduktion. Netzwerkorganisation, auf der einen Seite, ist im Wesentlichen durch eine intensive Arbeitsteilung zwischen Unternehmen sowie deren Koordination durch Vertrauen und Informationsintegration gekennzeichnet [Sie99b]. Dies erfordert eine Orientierung an durchgängigen Produktionsprozessen und deren geregelte Strukturierung in modulare Prozesseinheiten. Dienstleistungen, auf der anderen Seite, sind per Definition (nutzenstiftende) Prozesse. Der Produktionsprozess ist in vielen Fällen schon das Dienstleistungsprodukt. Die bislang wenig beachtete Gestaltung und Unterstützung von Dienstleistungsprozessen wirkt sich somit unmittelbar auf das Produkt aus.

Auf Grund dieser Zusammenhänge wird die Prozessorganisation im Folgenden noch etwas vertieft. Zunächst werden grundlegende Eigenschaften ökonomischer Prozesse beleuchtet und insbesondere auf die Dienstleistungsproduktion ausgeweitet. Dann wird die organisatorische Gestaltung dieser Prozesse erläutert.

2.3.1.1. Geschäftsprozesse zur Dienstleistungsproduktion

Als Basis der weiteren Überlegungen soll im Folgenden zunächst ein genereller organisatorischer Prozessbegriff eingeführt werden. Daraufhin erfolgt eine Systematisierung organisatorischer Prozesse sowie die Erweiterung zum allgemeinen Geschäftsprozessbegriff. Schließlich werden Geschäftsprozesse bei der Produktion von Dienstleistungen abgegrenzt und charakterisiert.

Ein generisch-organisatorischer Prozessbegriff Ein abstrakter Prozess besteht grundsätzlich aus einer Folge logisch zusammenhängender Aktivitäten. Diese werden in einer Zeitspanne nach bestimmten Regeln durchgeführt. Der Anstoß erfolgt in der Regel durch ein Ereignis. Im Verlauf bildet sich aus einer definierten Eingabe ein Wert. Dieser wird als definierte Ausgabe weitergereicht. Der Prozessverlauf ergibt somit eine inhaltlich abgeschlossene Vorgangskette. Die Abgrenzung der Prozessinhalte unterliegt dabei der subjektiven Problemsicht des Organisators. Dieser fasst damit einen Teil der betrieblichen Leistungserstellung und -verwertung zusammen, der isoliert betrachtet werden soll. Die folgende Definition nach [SZ99, S. 49] gibt eine Zusammenfassung:

Definition 2 (Prozess) *Ein Prozess beinhaltet die Erstellung einer Leistung oder Veränderung eines Objekts durch eine Folge logisch zusammenhängender Aktivitäten.*

Die Untersuchung spezifischer Prozesseigenschaften kann durch eine systemtheoretische Betrachtung erfolgen. Dabei werden Abläufe menschlicher Arbeit als *Arbeitssystem* abstrahiert. Abbildung 2.9 zeigt dessen Elemente im Überblick.

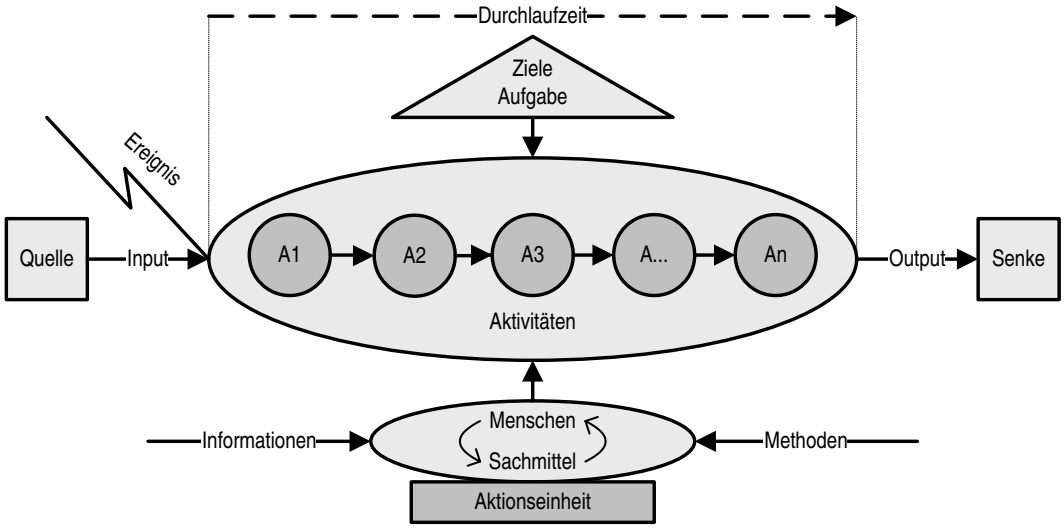

Abbildung 2.9.: Elemente eines Arbeitssystems [SZ99, S. 50]

Den Kern des Systems bilden Vorgänge, die hier als *Arbeitsprozesse* bezeichnet werden [Kos62, S. 185 f.]. Diese sind nach Schulte-Zurhausen durch folgende Merkmale gekennzeichnet [SZ99, S. 50]:

- Prozesse transformieren Inputs in einen definierten Output.
- Prozesse haben eine Aufgabe und sind auf Ziele ausgerichtet.
- Transformation erfolgt durch inhaltlich verknüpfte Aktivitäten.
- Prozesse werden durch ein Ereignis angestoßen.
- Prozessvollzug (Arbeit) erfolgt durch Menschen und Sachmittel (Aktionsträger).
- Prozesse haben eine Quelle (z. B. Lieferant) und eine Senke (z. B. Kunde).
- Aktivitäten werden nach geregelten Methoden durchgeführt.
- Aktionsträger benötigen zur Arbeit Methodenwissen und Informationen.
- Prozesse vollziehen sich in einem Zeitrahmen (Durchlaufzeit).

Ein Arbeitsprozess ist durch seinen definierten Output immer zweck- und letztendlich kundenbezogen. Der Output hängt u. a. von Ressourcen zur Leistungserstellung und -verwertung ab: Menschen, Sachmitteln, Methoden und Informationen. Durch sachliche und räumliche Zuordnung von Menschen und Sachmitteln werden handlungsfähige Aktionseinheiten gebildet. Diese stellen die kleinsten produktiven Einheiten im Prozess dar. Sie sind ferner durch eine quantitative und eine qualitative Kapazität

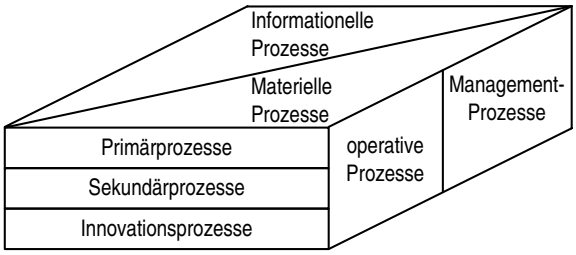

Abbildung 2.10.: Prozessarten [SZ99, S. 52]

begrenzt. In einem Arbeitssystem konkretisiert sich die Prozessorganisation nun in folgender Weise:

Definition 3 (Prozessorganisation) *Der Begriff der Prozessorganisation umfasst eine dauerhafte Strukturierung von Arbeitsprozessen unter der Zielsetzung, das geforderte Prozessergebnis möglichst effizient zu erstellen [SZ99, S. 57].*

Grundsätzlich sind nur solche Prozesse strukturierbar und daher organisierbar, die (zumindest vom Schema her) wiederholt vorkommen [Kos62, S. 31]. Zudem ist der strukturierende Eingriff je nach Inhalt und Stellung der Prozesse mehr oder weniger erfolgswirksam.

Systematisierung organisatorischer Prozesse Zur Systematisierung von Arbeitsprozessen unter organisatorischen Gesichtspunkten schlägt Schulte-Zurhausen eine Typologie mit dreifacher Differenzierung von Prozessarten vor [SZ99, S. 51 ff.] (vgl. Abb. 2.10). Die drei Dimensionen lassen sich wie folgt charakterisieren:

- *Leistungsform* – Die erste Differenzierung unterscheidet Arbeitsprozesse nach der mit ihnen in Beziehung stehenden Form der betrieblichen Leistung. *Materielle Prozesse* betreffen im Wesentlichen materielle Leistungen. Sie sind hauptsächlich durch physische Verrichtungen an stofflichen Objekten geprägt. *Informationelle Prozesse* betreffen im Wesentlichen immaterielle Leistungen. Sie umfassen hauptsächlich Austausch und Verarbeitung von Informationen.

- *Leistungserstellung* – Die zweite Differenzierung unterscheidet Arbeitsprozesse nach ihrer Leistungserstellung. *Operative Prozesse* haben unmittelbar die eigentliche Leistungserstellung zur Aufgabe. *Managementprozesse* umfassen hingegen die Planung und Kontrolle von Zielen und Maßnahmen, die Personalführung und Organisationsgestaltung.

- *Wertschöpfung* – Die dritte Differenzierung unterscheidet Arbeitsprozesse schließlich in Bezug auf die Wertschöpfung. *Primärprozesse* sind Marktprozesse. Bei gegebenen Potenzialfaktoren tragen sie direkt zur Erstellung, Vermarktung

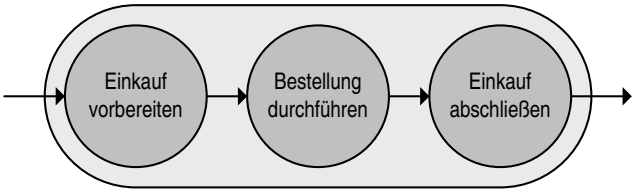

Abbildung 2.11.: Geschäftsprozess als Prozesskette [SZ99, S. 53]

und Betreuung von Produkten bei. Sie dienen somit unmittelbar der Wertschöpfung. *Sekundärprozesse* sind Infrastrukturprozesse, die nicht direkt an der Wertschöpfung beteiligt sind. Sie unterstützen jedoch die kontinuierliche Ausführung der Primärprozesse, z. B. durch Beschaffung bzw. Bereitstellung und Pflege der Potenzialfaktoren. *Innovationsprozesse* dienen der Entwicklung neuartiger Produkte, Verfahren oder Strukturen im technischen oder administrativen Bereich. Sie dienen weder direkt noch indirekt der (aktuellen) Wertschöpfung.

Komplexe organisatorische Prozesse In Bezug auf die verschiedenen Arbeitsprozesse einer Organisation ist zu bemerken, dass diese in der Regel nicht isoliert nebeneinander stehen. Sie sind aus funktionalen Gründen ablaufmäßig zu komplexen Prozessen verknüpft. Eine Menge derart verknüpfter Prozesse wird im Allgemeinen als *Prozesskette* bezeichnet.

Spezifische Prozessketten werden oft auch als *Geschäftsprozesse* umschrieben (vgl. Abb. 2.11). Über deren exakte Auslegung besteht jedoch bislang keine Einigkeit [Mer97, S. 210]. Eine gängige Variante findet sich im Umfeld der Prozessorganisation: Schulte-Zurhausen [SZ99, S. 54] charakterisiert Geschäftsprozesse z. B. als solche Prozessketten, die alle wesentlichen Aktivitäten zur Erstellung und Vermarktung eines Produkts, zur Steuerung und Verwaltung von Ressourcen sowie zur Beeinflussung der Umwelt verknüpfen. Derartig ausgelegte Geschäftsprozesse sind unmittelbar entscheidend für den Erfolg eines Unternehmens. Sie stehen daher im Mittelpunkt der prozessorientierten Organisationsgestaltung. Neben dem Geschäftsprozessbegriff aus dem Umfeld prozessorientierter Organisationsgestaltung ist des Weiteren eine Auslegung *allgemeiner Geschäftsprozesse* als inhaltlich abgeschlossene Prozesskette gebräuchlich. Diese Sicht soll auch im weiteren Verlauf der vorliegenden Arbeit eingenommen werden. Eine entsprechende Definition findet sich in [SZ99, S. 54]:

Definition 4 (Geschäftsprozess) *Unter einem Geschäftsprozess ist eine Kette von funktional zusammenhängenden Aktivitäten zu verstehen, die zu einem inhaltlich abgeschlossenen Ergebnis führen.*

Dem realen Geschäftsprozess steht dessen zweckorientierte, vereinfachte Abbildung durch ein *Geschäftsprozessmodell* gegenüber. Solche Modelle dienen der Dokumenta-

tion, Analyse und Gestaltung von Geschäftsprozessen. Bei der *Geschäftsprozessmodellierung* sind zunächst Modellierungszweck, Modelladressat und Geschäftsprozess zu bestimmen. Die sich hieraus ergebenden Anforderungen bestimmen die Wahl einer Modellierungsmethode und die in das Modell abzubildenden Merkmale des Geschäftsprozesses. Neben Funktionen und deren kausalen Beziehungen werden dabei oft weitere Merkmale erfasst. In der Literatur werden Organisationseinheiten, Input, Output, Ressourcen, Informationen, Medien, Transaktionen, Ereignisse, Zustände, Bedingungen, Operationen und Methoden erwähnt [Mer97, S. 211]. Je nach Zweck und Adressat der Modellierung kommen informale, semiformale oder formale Typen von Geschäftsprozessmodellen zum Einsatz. Informale Modelle liefern meist eine textuelle Prozessbeschreibung und dienen in erster Linie zur Erläuterung allgemeiner Konzepte. Semi-formale Modelle sind meist grafisch und dienen zur übersichtlichen Visualisierung komplexer Prozesse. Formalsprachliche Modelle dienen schließlich der detaillierten Erfassung aller notwendigen Prozessmerkmale zur formalen Analyse, Simulation oder automatisierten Ausführung.

Zur Bedeutung von Dienstleistungsproduktionsprozessen Die bisherige Betrachtung organisatorischer Prozesse erfolgte im Rahmen allgemeiner Unternehmen über alle Funktionsbereiche. Im Folgenden soll der Fokus auf die Produktionsprozesse in DLU gerichtet werden. Die speziellen Dienstleistungsmerkmale führen hier nämlich zu besonderen Bedingungen, etwa im Vergleich zur Sachgüterproduktion. Der Prozessbegriff hat bei der Dienstleistungsproduktion eine ganz natürliche Relevanz: Dienstleistungen manifestieren sich als nutzenstiftende Prozesse. Der Produktionsprozess führt nicht erst zur Leistung (wie bei Sachgütern), sondern stellt sie schon selbst dar. Er muss dabei neben internen auch nicht autonom disponierbare externe Produktionsfaktoren einbeziehen. Ihm sind deshalb weitere Prozesse vorgelagert, um eine andauernde Leistungsbereitschaft zu erzeugen. Die Leistungsübergabe erfolgt hier kontinuierlich (Uno-actu-Prinzip) in materieller oder immaterieller Form.

Zur Klassifikation von Dienstleistungsproduktionsprozessen können diese zunächst insgesamt als operative Prozesse eingestuft werden. Die Einschränkung auf informationelle Prozesse würde jedoch an der Realität vorbeilaufen; noch weniger liegen rein materielle Prozesse vor. Zum Teil wird hier der Begriff „Dienstleistungsprozess“ zur Kennzeichnung einer Mischform der Leistung verwendet [SZ99, S. 52]. Der Begriff soll in der vorliegenden Arbeit jedoch nicht verwendet werden. Es liegt weiterhin nahe, Dienstleistungsproduktionsprozesse bezüglich Vor- und Endkombination in eine Prozesskette zu zerlegen. Dabei kann zunächst festgestellt werden, dass die Vorkombination durch Sekundär- und die Endkombination durch Primärprozesse gekennzeichnet ist.[26] Insgesamt ist die Prozesskette dann als Geschäftsprozess einzuordnen, denn sie führt zum inhaltlich geschlossenen Ergebnis der Dienstleistung. Dieser Geschäftsprozess ist neben Anteilen an Unterstützungsprozessen sogar zu

[26] Eine genauere Strukturierung erfolgt später.

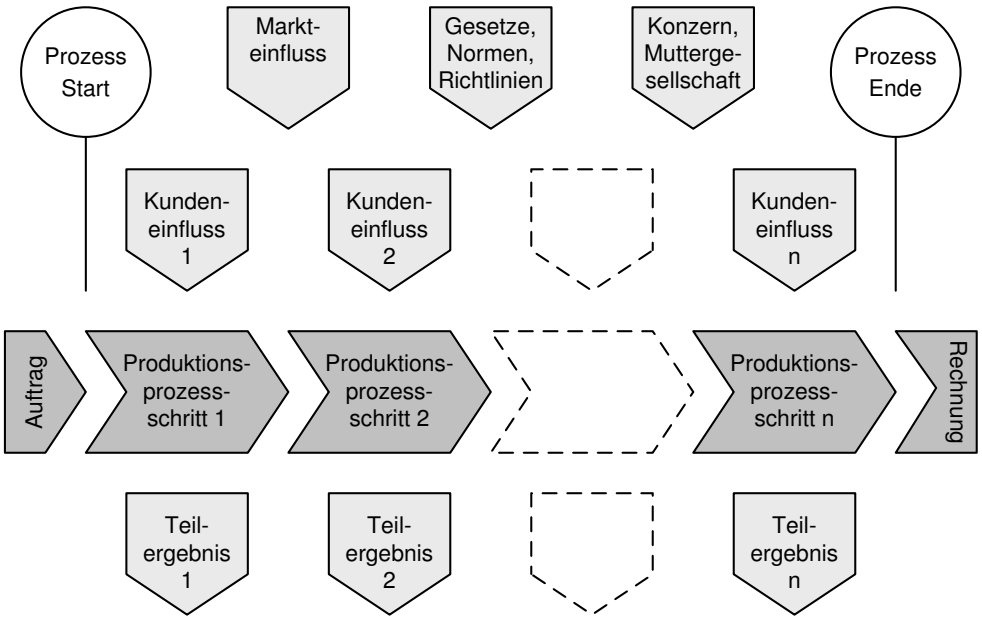

Abbildung 2.12.: Merkmale und Einflussfaktoren eines DLP

großen Teilen als Schlüsselprozess zu bezeichnen. Auf Grund ihrer zentralen Rolle sollen Geschäftsprozesse zur Dienstleistungsproduktion kurz als *Dienstleistungsprozesse* bezeichnet werden:

Definition 5 (Dienstleistungsprozess) *Als Dienstleistungsprozess (DLP) werden solche Geschäftsprozesse bezeichnet, die alle notwendigen Aktivitäten zur Produktion einer Dienstleistung umfassen.*

Abb. 2.12 illustriert noch einmal die praktischen Merkmale eines DLP nach Harms [Har03, S. 26 ff.]. Danach beginnt jeder DLP mit einem Kundenauftrag. Dieser Auftrag kann den Produzenten theoretisch jederzeit erreichen. Da sich der vom Kunden gewünschte Nutzen nicht vom DLP trennen lässt (Harms: *„der Prozess ist bereits das Produkt“*), kann der Produzent nicht auf Vorrat produzieren. Er muss also ständig Kapazitäten vorhalten, um den DLP vollziehen zu können. Dieser Vollzug basiert überwiegend auf Arbeitsleistungen von Menschen unter mehr oder weniger stark ausgeprägter Nutzung von Betriebsmitteln und vergleichsweise (in Bezug auf Sachgüter) geringer Bedeutung von Material.

Der DLP ist im Gegensatz zur gekapselten Sachgüterproduktion ein *offener Produktionsprozess*, d. h. dass der Kunde Einblick in den Produktionsprozess hat und in der Regel aktiv daran teilnimmt. Dementsprechend sind *Front Office Funktionen* mit Kundeninteraktion von unterstützenden *Back Office Funktionen* ohne Kundeninteraktion zu unterscheiden. Harms weist darauf hin, dass für die Wechselwirkung dieser Funktionen eine exzellente Information und Kommunikation benötigt wird. Der

Interaktionsgrad verstärkt sich noch dadurch, dass die Ergebnisse des Produktionsprozesses meist in dessen Verlauf entstehen. Als Beispiel sei ein Börseninformationsdienst genannt. Dieser übermittelt ständig aktualisierte Kurse der vom Kunden gewünschten Wertpapiere. Die Auslieferung der Ergebnisse erfolgt während des laufenden Produktionsprozesses. Am Ende des Prozesses sind in der Regel alle Ergebnisse übergeben. Abschließend erfolgt die Abrechnung und Bezahlung der Leistungen.

Weiterhin ist zu beachten, dass neben den in- und externen Produktionsfaktoren weitere Einflussfaktoren auf den DLP einwirken: Intern geht eine generelle Wirkung von indirekten dispositiven Faktoren (z. B. Personalmanagement) aus. Extern sind vor allem Markteinflüsse wie die Wettbewerbssituation sowie konjunkturelle und politische Einflüsse zu nennen. Einkommens- und Dienstleistungsentwicklung von Marktsegmenten und Volkswirtschaften spielen dabei ebenfalls eine Rolle. Zudem sind ggf. bestehende Gesetze, Normen und Richtlinien zu beachten. Einwirkungen, Vorgaben und Rahmenbedingungen können schließlich auch von einer Muttergesellschaft kommen.

Insgesamt wird die zentrale Bedeutung des DLP deutlich: Durch die Offenheit des Prozesses werden Schwächen unmittelbar und öffentlich sichtbar. Die Leistungsfähigkeit des Prozesses wirkt sich in direkter Weise auf Leistung und Qualität des Dienstes aus. Durch die Synchronizität von Leistungserstellung und Abnahme lassen sich Fehler im Prozessablauf kaum rückgängig machen, sondern bestenfalls kompensieren. Insgesamt kommt daher dem Organisationsmanagement, d. h. der Gestaltung des DLP, *die* wesentliche Rolle beim Management der Dienstleistungsproduktion zu.

2.3.1.2. Gestaltung von Dienstleistungsprozessen

Die Aufgabe der Prozessgestaltung besteht darin, eine strukturierte Folge von Aktivitäten zu bestimmen, die zu einem spezifischen Ergebnis für den Kunden führt. Diese Vorgehensweise stellt das Resultat bzw. Kundenergebnis in den Vordergrund. Es wird dabei die Hypothese vertreten, dass die Optimierung von Funktionsfolgen stärker auf das Ergebnis wirkt als diejenige von isolierten Funktionen. In der materiellen Fertigung wird dies seit längerem praktiziert (etwa bei Ford nach Taylor). Bei informationellen Prozessen und insbesondere Sekundärprozessen findet hingegen oft keine explizite Prozessgestaltung statt. Als Grund wird meist die intuitive oder kreative Natur der Vorgänge angeführt. Es finden sich jedoch auch hier eine Reihe organisierbarer und automatisierbarer Aktivitäten. Empirische Untersuchungen zeigen hier ein hohes Optimierungspotenzial informationeller Prozesse durch Anwendung prozessorientierter Organisationsgestaltung.[27] Dies trifft insbesondere auch auf die überwiegend informationellen DLP zu.

[27] Vgl. [SZ99, S. 75].

Im Allgemeinen ist die *Gestaltung* von Geschäftsprozessen[28] von der prozessorientierten *Reorganisation* funktionaler Organisationen[29] zu unterscheiden. In jedem Fall wird unterstellt, dass Durchlaufzeiten, Kosten und Qualität der Leistungserstellung nur bei ganzheitlicher Prozessorganisation zu optimieren sind.[30] Die Prozessgestaltung wirkt dabei auch auf die interne und externe Reaktionsgeschwindigkeit. Zudem lassen sich derart gestaltete Geschäftsprozesse durch Informationssysteme unterstützen. In diesem Sinne formuliert Harms in [Har03] die folgenden Gestaltungsziele für Dienstleistungsprozesse:

- Senkung der Prozesskosten und Verbesserung der Wirtschaftlichkeit
- Verbesserung des Kundenfokus
- Vereinfachung und Verschlankung des DLP
- Vermeidung nicht-wertschöpfender Tätigkeiten
- Gewährleistung von Prozessqualität
- Flexibilisierung der Prozesskette aus Kundensicht
- Automatisierung von Prozessanteilen
- Prozessstrukturierung zur zeitlichen Optimierung
- Mitarbeiterorientierte Prozessgestaltung und Einbezug von Mitarbeitern
- Ressourcenschonende Prozessgestaltung

Ein Vorgehensmodell zur Prozessgestaltung Prozessorientierte Organisationsgestaltung ist komplex und sollte einem Vorgangsmodell folgen. Schulte-Zurhausen bemerkt, dass die hierfür vorgeschlagenen Ansätze bislang uneinheitlich sind und gibt selbst ein sehr elementares Verfahrensschema an [SZ99, S. 76]. Dieses startet mit der Analyse der strategischen Geschäftsfelder eines Unternehmens. Es schließen sich die Definition, Strukturierung und Integration von Geschäftsprozessen an. Nach Ausgestaltung einzelner Prozessketten werden diese zum einen extern verkettet und zum anderen stellenmäßig zugeordnet. Nach der Implementierung erfolgt früher oder später eine Prozessverbesserung. Abbildung 2.13 skizziert das gesamte Vorgehensmodell, dessen Anteile nun erläutert werden.

[28] Englisch Business Process Modelling (BPM); siehe z. B. [GSV+94].
[29] Englisch Business Process Reengineering (BPR); siehe z. B. [HC93], [OF96].
[30] Vgl. z. B. [GSV+94].

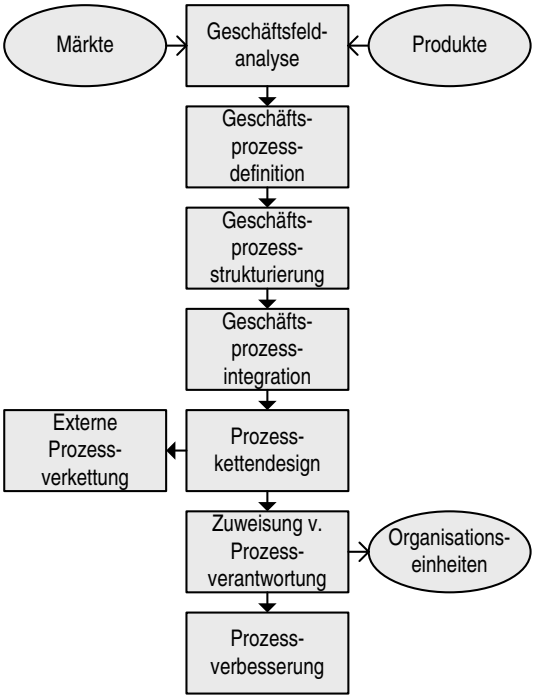

Abbildung 2.13.: Vorgehensmodell zur Prozessgestaltung (nach [SZ99, S. 76])

Vom Geschäftsfeld zum Geschäftsprozess Grundsätzlich soll durch die Gestaltung von Geschäftsprozessen eine Optimierung des Kundenergebnisses erfolgen. Zuvor müssen jedoch die Anforderungen von Kunden und Märkten bekannt sein. Dazu dient eine *Geschäftsfeldanalyse*. Gegenstand der Betrachtung ist hierbei das strat. Geschäftsfeld (SGF): eine homogene Produktgruppe mit identifizierbarer Kundengruppe und eigenständigem Unternehmensbeitrag, die insbesondere strategisch unabhängig ist. Während der Geschäftsfeldanalyse werden die SGF des Unternehmens identifiziert.

Bei der *Geschäftsprozessdefinition* werden aus den Anforderungen individuelle Geschäftsprozesse einzelner SGF abgeleitet. Diese Geschäftsprozesse werden dabei also problemorientiert statt funktionsorientiert festgelegt. Je nach Bezug zur Problemlösung werden Primär- (in der Regel ca. 8 pro SGF), Sekundär- und Innovationsprozesse bestimmt (s. o.). Der funktionale Umfang ist jeweils problemspezifisch. Neben der problemspezifischen Funktionalität wird auch die Planung und Steuerung einbezogen. Geschäftsprozesse sind dadurch in der Regel stellenübergreifend. Sie sollten dabei einerseits nicht zu groß und unübersichtlich werden, andererseits aber auch nicht zu zahlreich und koordinationsintensiv. Aus diesem Grund ist bei der Abgrenzung auch *Selbstständigkeit* zu beachten: Einzelne Geschäftsprozesse sollten unabhängig und ohne Wechselwirkungen mit anderen Geschäftsprozessen gestaltbar sein.

Nach erfolgter Abgrenzung werden die Kunden- und Marktanforderungen an Primärprozesse bestimmt. Dazu werden aus den Erfolgsfaktoren des SGF prozessspe-

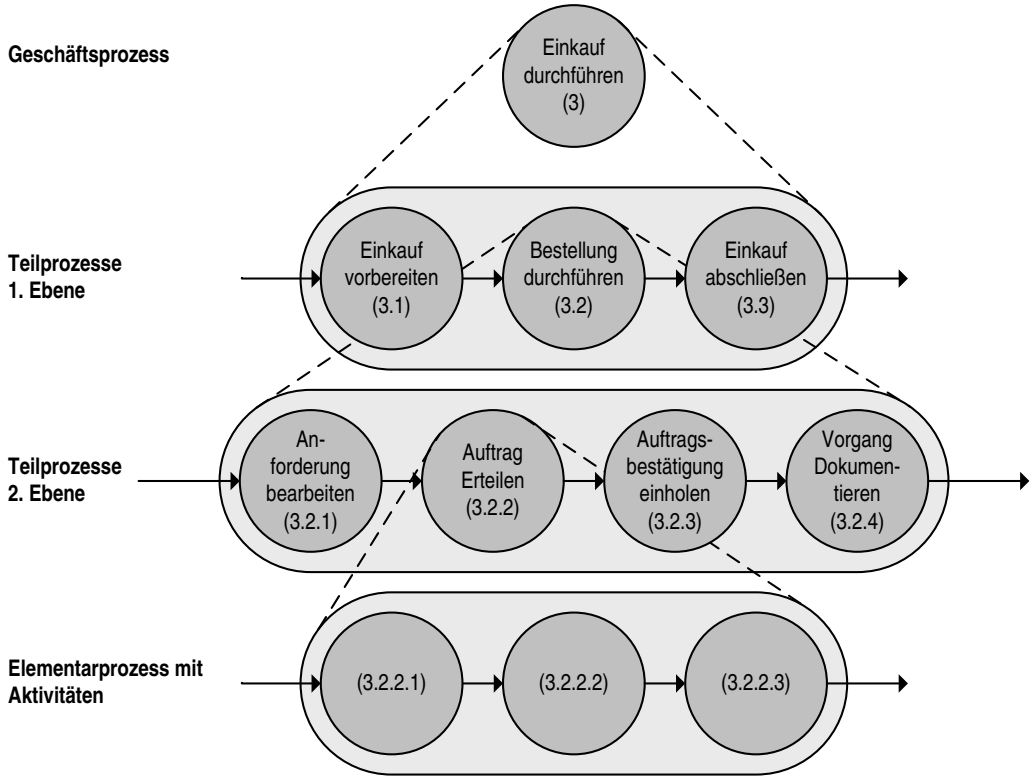

Abbildung 2.14.: Strukturierung eines Geschäftsprozesses (vgl. [SZ99, S. 89])

zifische Erfolgsbedingungen abgeleitet. Des Weiteren erfolgt eine Grobspezifikation der Geschäftsprozesse: für jeden werden Leistung und Hauptaktivitäten angegeben. Dabei sind insbesondere auch Leistungsmengen, Anfangs-/Endpunkt, auslösendes Ereignis und operationale Ziele festzulegen. Am Ende der Geschäftsprozessdefinition werden noch die Schlüsselprozesse hervorgehoben: das sind diejenigen Prozesse, die für das SGF maßgeblich erfolgsbestimmend sind. Sie bilden (zunächst) den Schwerpunkt der weiterführenden Prozessgestaltung.

Strukturierung von Geschäfts- und Dienstleistungsprozessen Die *Geschäftsprozessstrukturierung* zielt auf eine exakte Ausarbeitung einzelner Prozesse bis hin zu konkreten Handlungsanweisungen. Hierbei erfolgt zunächst eine mehrstufige Dekomposition des Geschäftsprozesses in Teilprozesse. Es entsteht eine hierarchische Prozessstruktur (vgl. Abb. 2.14). Sie endet bei *Elementarprozessen*. Diese können durch Aktionsträger von Anfang bis Ende ohne Unterbrechung durchgeführt werden.

Dekomposition geht mit der Arbeitsanalyse der betriebswirtschaftlichen Organisationslehre konform. Sie zieht allerdings nur Objekt und Verrichtung als Kriterium heran. Die Wahl der Gliederungstiefe richtet sich individuell nach Prozessart und organisatorischer Aufgabenstellung. Eine sehr feine Gliederung bis hin zu den Ele-

mentarprozessen ist insbesondere bei gewünschter Automatisierung von Prozessen notwendig. Dies ist vor allem bei der Unterstützung informatorischer Prozesse durch betriebliche Informationssysteme hervorzuheben.

Entsprechend ist auch hier und im weiteren Verlauf die Verwendung informaler, semiformaler oder formaler Prozessmodelle zu prüfen. Für die Planung im betrieblichen Management eignen sich u. a. semiformale *Flow Charts*. Zielt die Modellierung hingegen auf eine informationstechnische Automatisierung von Prozessanteilen, so können Prozessmodelle für betriebswirtschaftliche Standardsoftware wie z. B. *ARIS* [Sch98] verwendet werden. Eine weitere Möglichkeit, die im Laufe der vorliegenden Arbeit noch ausführlich betrachtet wird, ist die Verwendung (semi-)formaler Workflow-Modelle.

Sind schließlich alle Teilprozesse und Aktivitäten bestimmt, müssen noch deren kausale Zusammenhänge festgelegt werden. Diese ergeben sich in der Regel durch wechselseitige Input-Output-Beziehungen, d. h. die Ergebnisse eines Prozesses dienen als Grundlage von anderen. Daher sind die Prozesse in entsprechender Reihenfolge auszuführen. Insgesamt ergibt sich die so genannte *Prozessarchitektur*:

Definition 6 (Prozessarchitektur) *Die Prozessarchitektur beinhaltet die hierarchische Darstellung aller in einem Geschäftsprozess enthaltenen Teilprozesse und Aktivitäten sowie ihrer Input-Output-Beziehungen [SZ99, S. 89].*

Aus der Prozessarchitektur ergeben sich insbesondere Aussagen über Ablaufkontinuität, Redundanzen und Schnittstellen zwischen Aktivitäten und Teilprozessen. Dies lässt sich noch durch Berücksichtigung von Varianten erweitern. Varianten treten oft in Bezug auf die Leistungen auf. Sie übertragen sich auf die Prozesse. Um Transparenz zu gewährleisten ist es ratsam, Varianten relativ zu einem Standardablauf zu spezifizieren, d. h. es werden nur die individuellen Unterschiede vermerkt.

Für die Geschäftsprozessstrukturierung sind im Falle von DLP noch einige Besonderheiten hervorzuheben. Insgesamt empfiehlt Harms dabei, sich zunächst auf die Grobstruktur reiner *Vorwärtsprozesse* zu konzentrieren [Har03, S. 32 ff.]. Hier sind nur die wichtigsten (d. h. wertschöpfenden) Schritte zur Dienstleistungsproduktion enthalten. Die kausalen Beziehungen dieser Basisaktivitäten sind auf eine lineare Abfolge beschränkt. Insbesondere werden keine Schleifen oder Verzweigungen betrachtet, die zur Wiederholung von Aktivitäten führen. Zudem wird die Kennzeichnung von *Kernprozessen*, *Supportprozessen* und *Steuerungsprozessen* vorgeschlagen; Abbildung 2.15 veranschaulicht dies am Beispiel eines Restaurants. Diese Strukturierung gründet sich auf die *Prozessaufgaben*. Die einzelnen Aufgabenbereiche können dabei wie folgt charakterisiert werden:

- *Kernprozesse* sind diejenigen Prozessanteile, die den wesentlichen Kundenkontakt abdecken und überwiegend im Front Office Bereich ablaufen werden. Diese Prozessanteile wirken sich unmittelbar auf die Dienstleistung aus. Sie

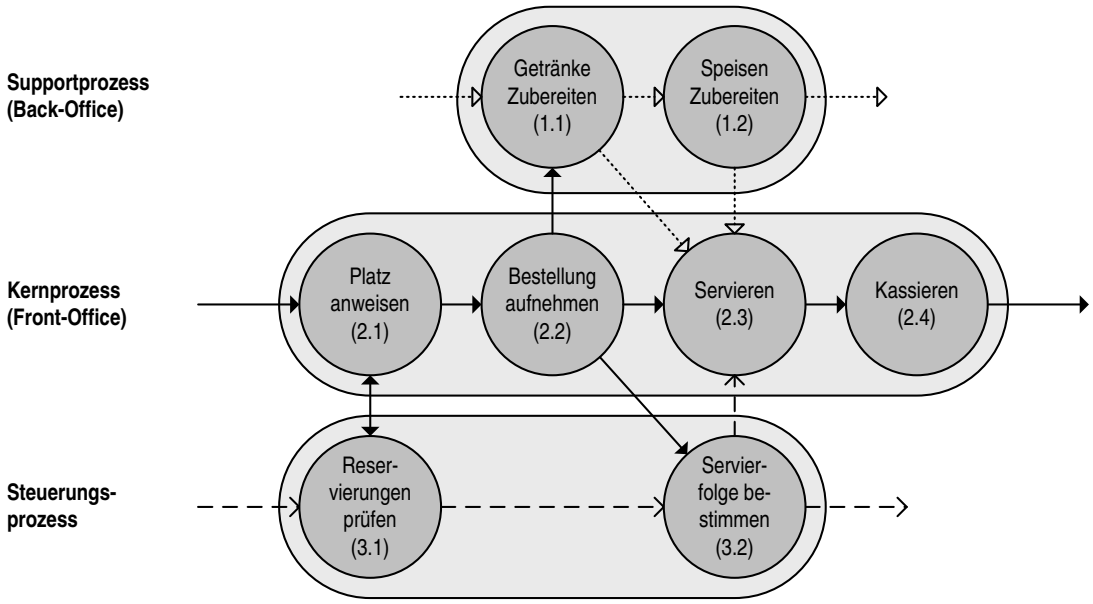

Abbildung 2.15.: Strukturierung von DLP (nach [Har03, S. 33])

müssen daher sorgfältig gestaltet werden. Dies gilt umso mehr, weil sich über die Gestaltung auch die Produktivität beeinflussen lässt.[31]

- *Supportprozesse* enthalten Aktivitäten, die zur Unterstützung des Kernprozesses dienen. Diese Aktivitäten beinhalten dabei überwiegend keinen Kundenkontakt – d. h. sie sind größtenteils zur Ausführung im Back Office Bereich gedacht. Dadurch sind diese Prozessanteile weniger kritisch. Diese gestalterische Maßnahme gleicht der Klassifizierung, die in Blueprints mittels *Line of Visibility* vorgenommen wird.[32]

- *Steuerungsprozesse* setzen sich schließlich aus administrativen Aufgaben zusammen. Hier wird die *Koordination* und *Synchronisation* von Aktivitäten des DLP realisiert. Insbesondere muss eine reibungslose *Kommunikation* zwischen den Stellen gewährleistet sein. Ferner kann hier mithilfe von Informationsverarbeitung die Generierung von Steuergrößen erfolgen. Die Einteilung von Aktivitäten oder Prozessanteilen als Support- oder Steuerungsprozess ist dabei nicht immer eindeutig zu treffen. Es wird darauf hingewiesen, dass die Offenlegung dieser Prozesse zu einer Erhöhung der durch den Kunden wahrgenommenen Dienstleistungsqualität führen kann.

Zur weiteren Verfeinerung des DLP schlägt Harms u. a. vor, zusätzlich eine orthogonale, inhaltliche Klassifizierung in *Startprozesse*, *Kernleistungsprozesse* und

[31] Vgl. 2.2.2.2.
[32] Vgl. 2.2.2.2.

administrative Prozesse vorzunehmen. Werden solche Prozessanteile im DLP identifiziert, so kann in der Regel von brancheninternen bzw. -externen Ähnlichkeiten profitiert werden. In diesem Fall kann unter Umständen sogar ein Recycling bestehender Prozessmuster erfolgen. Im Einzelnen gilt:

- *Startprozesse* eröffnen die Dienstleistungsproduktion. In der Regel wird hier der Kunde identifiziert und ein Auftrag generiert. Diese Prozessanteile weisen sogar branchenübergreifend einen hohen Ähnlichkeitsgrad auf.
- *Kernleistungsprozesse* erbringen den nutzenstiftenden Anteil der Dienstleistung. Sie sind dabei meist spezifisch für ein Branche.
- *Administrative Prozesse* umfassen Tätigkeiten wie Auftragseröffnung, Auftragsvervollständigung und Auftragsabschluss mit Rechnungsstellung. Auch diese Prozesse sind oft branchenübergreifend.

Von Geschäftsprozessstrukturen zu konkreten Prozessketten Zurück auf allgemeiner Ebene liegen am Ende der Geschäftsprozessstrukturierung pro SGF eigene Prozessketten vor. Diese sind zwar schon strukturiert, aber noch nicht voll ausgestaltet. Bevor jedoch der detaillierte Prozessentwurf erfolgt, sollten zunächst Redundanzen zwischen den SGF analysiert und ggf. beseitigt werden. Redundanzen sind nämlich durch die bislang rein extern getriebene Prozessgestaltung kaum zu vermeiden.

Deshalb folgt nun eine Phase der *Geschäftsprozessintegration*. Hier werden mehrfach vorhandene Prozesse in verschiedenen SGF auf Ähnlichkeiten und Stellenwert untersucht. Grundsätzlich sind vor allem ähnliche Prozesse und solche mit geringem Stellenwert anzugleichen und zusammenzufassen. Dies kann durch Variantenbildung bei der Prozessstrukturierung geschehen. Parallele Varianten (als getrennte Geschäftsprozesse) sind geschäftsfeldbedingt nur bei Primärprozessen zu tolerieren.

Nach dieser abschließenden definitorisch-strukturellen Optimierung kann der endgültige *Prozesskettenentwurf* folgen. Hierbei werden detaillierte, insbesondere quantitative Merkmale bestimmt und in den Prozessketten umgesetzt. Dies beinhaltet u. a. eine Ermittlung des Zeitaufwands, Festlegung von Leistungsanforderungen, Bestimmung von Leistungsmerkmalen und Kontrollpunkten, Gestaltung der Informationsinfrastruktur, zeitliche und räumliche Gestaltung sowie Prozessdokumentation.

An diesem Punkt liegen alle erfolgsrelevanten Unternehmensvorgänge als detaillierte Prozessketten vor. Es kann nun eine *Zuweisung der Prozessverantwortung* erfolgen. Dabei werden Teilprozesse und Aktivitäten einer oder mehreren Personen durch Festlegung von Stellenaufgaben zugewiesen. Insbesondere scheint es vorteilhaft, die Durchlaufverantwortung für komplette, in sich geschlossene Abläufe durch Stellen oder Teams zusammenzufassen. Eine Möglichkeit dazu besteht in der Schaffung prozessspezifischer Organisationseinheiten. Bei stellen- oder abteilungsübergreifenden Prozessen kann auch ein so genannter *Prozesseigner* als Teil der Sekundärorganisation bestimmt werden.

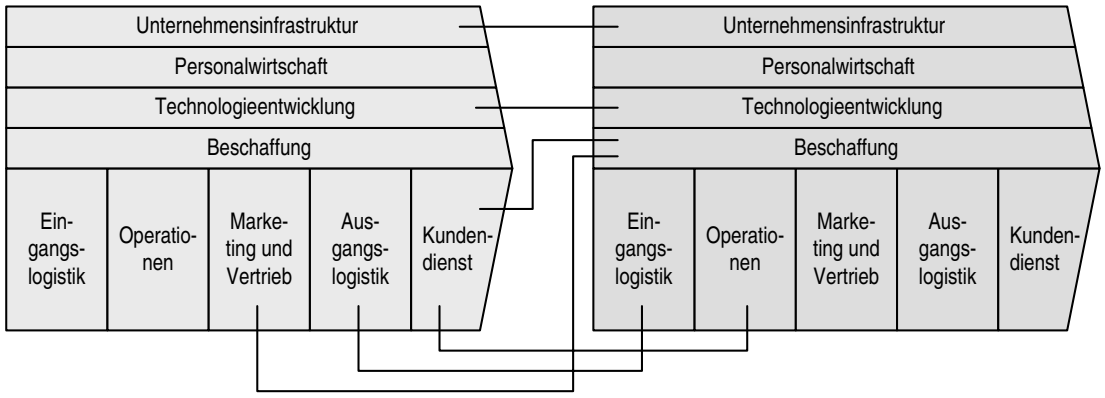

Abbildung 2.16.: Zwischenbetriebliche Verkettung von Geschäftsprozessen

Externe Prozessverkettung Parallel zur internen Gestaltung der Aufbauorganisation erfolgt eine externe Gestaltung der zwischenbetrieblichen *Prozessverkettung*. Hierbei wird den Beziehungen der Organisation zu ihren Umsystemen (z. B. Lieferanten, Transporteure, Kunden) Rechnung getragen. Mit diesen Umsystemen steht das Unternehmen nämlich in einem Leistungsverbund. Abb. 2.16 zeigt diesen Verbund für zwei Unternehmen. Hierbei sind die primären (vertikalen) und sekundären (horizontalen) Prozesse ihrer SGF als Wertschöpfungsketten nach Porter dargestellt.[33] Die aus dem Verbund resultierenden Leistungsbeziehungen müssen nach Porter explizit koordiniert werden. Dabei findet stets ein Informationsaustausch statt. Hier bietet sich der Einsatz organisationsübergreifender IV[34] an.

Der externen Prozessverkettung kommt in jüngster Zeit eine immer größere Bedeutung zu. Dies gilt insbesondere für ihre IT-gestützte Variante. Grund dafür sind substanzielle Veränderungen der traditionellen zwischenbetrieblichen Strukturen: Dabei werden nicht nur die Zulieferketten komplexer. Es findet darüber hinaus eine *Dekomposition* der klassischen Produktion nach prozessorientierten Kernkompetenzen statt. Diese werden wiederum im organisatorischen Rahmen von *Netzwerkorganisationen* integriert. In der Folge ergibt sich eine quantitative und qualitative Steigerung der externen Verkettungen. Diese werden in der nächsten Untersektion eingehender untersucht.

2.3.2. Dienstleistungsunternehmen als virt. Netzwerkorganisationen

Als Reaktion auf den insgesamt stetig steigenden Wettbewerbsdruck sowie auf neue Wettbewerbsfaktoren ist heute immer öfter eine längerfristige, geregelte Form geschäftlicher Beziehungen zwischen autonomen Unternehmen zu beobachten, die mehr

[33] Vgl. [SZ99, S. 54 ff.].

[34] Z. B. nach dem Standard Electronic Data Interchange for Administration, Commerce and Transport (EDIFACT) [UN/93]; eine detaillierte Betrachtung erfolgt in Kapitel 3.

kollaborative als wettbewerbliche Züge trägt [Sie99b]. Dies ist durch zahlreiche Studien empirisch belegt.[35] Auf Grund der außerordentlichen praktischen Relevanz wurden vor allem seit dem letzten Jahrzehnt verstärkt Anstrengungen unternommen, die Vielfalt kollaborativer Beziehungen zu einer fundierten Lehre der *Netzwerkorganisation* zu vereinheitlichen, zu institutionalisieren und letztendlich zu instrumentalisieren. In der Managementforschung wird heute die Netzwerkorganisation (wie auch die Prozessorganisation) neben den klassischen hierarchisch-funktionalen bzw. -divisionalen Organisationsformen als Primärorganisation verstanden [Syd99, S. 281].

Grundsätzlich kann eine enge Beziehung zwischen Netzwerk- und Prozessorganisation festgestellt werden: die Organisation von Netzwerken ist nämlich nicht nur mit dem zuvor beschriebenen Trend zur prozessorientierten Unternehmens- und Dienstleistungsorganisation gut verträglich, sondern kann laut Sydow sogar *„... geradezu als logische Konsequenz der Umsetzung dieses Organisationsprinzips gelten“* [Syd99, S. 279]. Dies ist darauf zurückzuführen, dass die bei Prozessorientierung fokussierten Geschäftsprozesse in aller Regel nicht auf einzelne Unternehmen beschränkt sind.[36] Vielmehr bilden die Geschäftsprozesse eines einzelnen Unternehmens nur Glieder unternehmensübergreifender Wertschöpfungsketten; dies gilt insbesondere für Kernprozesse, teilweise aber auch für deren Unterstützungsprozesse. Die einzelnen Glieder müssen daher verknüpft und deren Beziehungen koordiniert werden.[37] Soll eine Wertschöpfungskette dann in Gänze optimiert werden, so sind alle daran beteiligten Unternehmen einzubeziehen. Diese unternehmensübergreifenden Aspekte der Prozessgestaltung werden in fast allen aktuellen Konzepten der Prozessorganisation hervorgehoben [PF95, S. 24 ff.].

Somit bedingt die Prozessorganisation eine Organisation des Netzwerks beteiligter Unternehmen. Anders herum machen die verschiedenen Formen von Netzwerkorganisationen genau das möglich: sie bieten Koordinationsmechanismen zur Abstimmung autonomer Organisationseinheiten. Diese sind effektiver und effizienter als die rein preisliche Koordination über Märkte oder die hierarchische Koordination monolithischer Organisationen. Im Falle virtueller Unternehmen, die durch extensiven Einsatz von IV-Technik (IT) eine besonders flexible und dynamische Form von Netzwerkorganisationen darstellen, geht das so weit, *„dass hier laufend in Frage gestellt werden kann, welche in- oder externen Einheiten die einzelnen Aktivitäten in einem Wertschöpfungsprozess erbringen“* [MS97, S. 11]. Speziell für die Dienstleistungsproduktion bringt das, wie gezeigt werden wird, besondere Vorteile mit sich.

Im Folgenden werden zunächst die Merkmale, Typen und Aufgaben allgemeiner Netzwerkorganisationen beschrieben. Daran anschließend wird deren Sonderform des virtuellen Unternehmens herausgestellt und die besondere Eignung zur Dienstleistungserstellung hervorgehoben.

[35] Siehe z. B. [WN98].
[36] Vgl. z. B. [Ott96].
[37] Siehe oben und Abb.2.16.

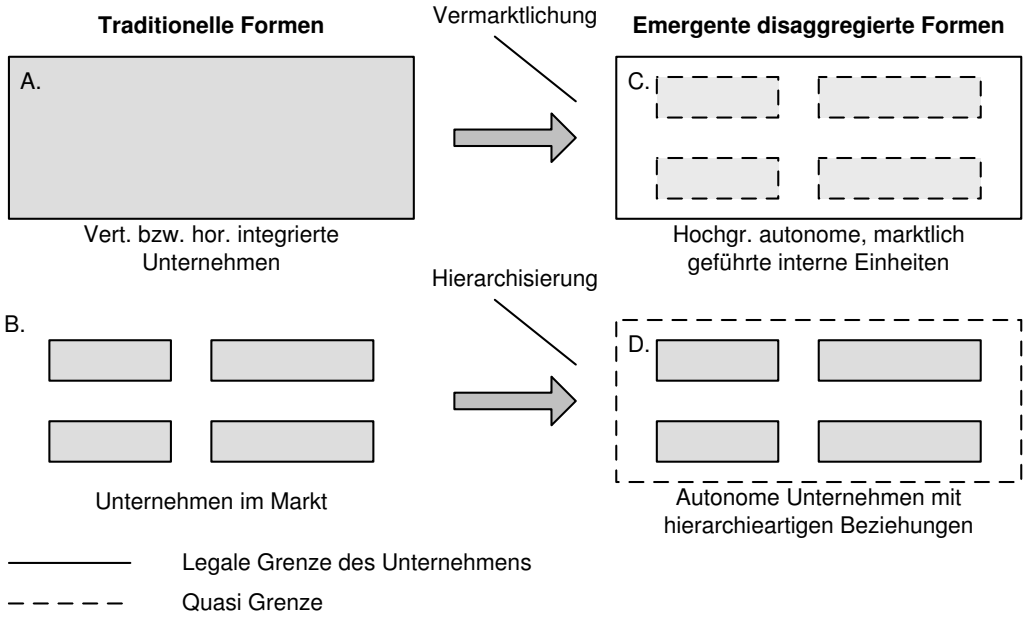

Abbildung 2.17.: Trends zur disaggregierten Organisationsform [Syd99, S. 282]

2.3.2.1. Unternehmen als organisatorische Netzwerke

Unternehmensnetzwerke beschreiben ganz allgemein die koordinierte Zusammenarbeit mehrerer rechtlich selbstständiger und formal unabhängiger Unternehmen.

Der Hintergrund dieser disaggregierten Organisationsform[38] ergibt sich aus zwei diametral entgegengesetzten Tendenzen (vgl. Abb. 2.17): Zum einen gliedern einzelne Unternehmen im Zuge einer Konzentration auf Kernkompetenzen sekundäre Funktionsbereiche als autonome Organisationseinheiten aus (*Quasi-Externalisierung*), zum anderen ziehen Gruppen von Unternehmen in Hinblick auf Synergien ihre Kompetenzen und Ressourcen zusammen (*Quasi-Internalisierung*).

Diese Vernetzungstendenzen führen dann zu äußerst vielfältigen Typen[39] der N-Form: Sie manifestieren sich etwa in verschiedenen Netzwerkorganisationen[40] im engeren Sinn oder Netzwerkkonzepten[41] im weiteren Sinn. Trotz der Vielzahl von Ausprägungen lassen sich einige generische Aussagen treffen. Hierzu soll die oft verwendete Definition von Sydow [Syd92] zugrunde gelegt werden.

[38] Im Allgemeinen als *Netzwerkorganisation (N-Form)* bezeichnet.

[39] Vgl. [Brü99, S. 26 ff.].

[40] Z. B. strategische Allianzen, Hollow Networks, Keiretsu, Management Holdings und virt. Unternehmen.

[41] Z. B. fraktale Unternehmen, Frenchising, Joint Ventures, Outsourcing, Telearbeit und elektronische Märkte.

Definition 7 (Unternehmensnetzwerk) *Ein Unternehmensnetzwerk stellt eine auf die Realisierung von Wettbewerbsvorteilen zielende Organisationsform ökonomischer Aktivitäten dar, die sich durch komplex-reziproke, eher kooperative denn kompetitive und relativ stabile Beziehungen zwischen rechtlich selbstständigen, wirtschaftlich jedoch zumeist abhängigen Unternehmen auszeichnet.*

Konstituierende Merkmale von Unternehmensnetzwerken Den Ausführungen von Siebert in [Sie99b, S. 9 ff.] folgend, besteht das wesentliche Merkmal von Unternehmensnetzwerken in einer gemeinsamen Zielsetzung der Netzwerkpartner. Auf Grund dieses *„Kollektivziels“* ordnen die Teilnehmer ihre Individualziele zum Teil den Interessen des Netzwerks unter. Als Ergebnis entsteht ein relativ breites Spektrum organisatorischer Koordinationsformen zwischen den Extremen des Marktes und der Hierarchie. Letztere wurden in Bezug auf [Coa37] lange als einzige Möglichkeiten der Koordination angesehen. Nachdem hybride Koordinationsmodelle u. a. durch Riordan und Williamson postuliert worden waren [RW85], wurde der Netzwerkbegriff erstmals von Jarillo explizit aufgegriffen [Jar88]. Hiernach sind Unternehmensnetzwerke im Gegensatz zu marktlicher Koordination durch kooperative anstatt rein wettbewerbliche Verhaltensweisen ihrer Teilnehmer gekennzeichnet. Gleichzeitig grenzen sie sich aber auch von hierarchischer Koordination nach Art eines Unternehmens durch marktinduzierte Flexibilität und Einsatzbereitschaft ihrer Teilnehmer ab.

Die Teilnehmer eines Unternehmensnetzwerks stehen also weder im direkten preislichen Wettbewerb eines Marktes, noch sind sie in die Hierarchie einer rechtlichen Unternehmensform eingebunden. Trotzdem kommen gewisse Charakteristika beider Koordinationsextrema zum Tragen: Zum einen zeichnen sich ökonomische Netzwerke im Sinne eines Marktes durch *funktionale Spezialisierung* und *Effizienzdruck* aus. Sie beruhen nämlich in der Regel auf einer intensiven Arbeitsteilung ihrer Teilnehmer. Diese bringen zum Erreichen des Kollektivziels jeweils diejenigen komplementären Ressourcen in die Wertschöpfung des Netzwerks ein, für die sie die größte unternehmensspezifische Kompetenz besitzen. Dabei stehen sie jedoch langfristig gesehen in einem Verdrängungswettbewerb zueinander sowie auch zu außenstehenden Unternehmen. Sie sind aus diesem Grund zu effizienter Leistungserbringung in Bezug auf Preis, Service, Innovation etc. angehalten, um nicht aus dem Netzwerk ausgeschlossen zu werden.

Zum anderen sind Unternehmensnetzwerke durch *Vertrauen* und *Informationsintegration* gekennzeichnet, die typischerweise bei unternehmensinternen Hierarchien vorzufinden sind. Vertrauen basiert auf dem kooperativen Verhalten der Teilnehmer; d. h. deren Verzicht auf opportunistische Handlungen. Deshalb können gewisse Absicherungen von Gefahrenpotenzialen (z. B. Know-how Abfluss) zum Teil entfallen und Wettbewerbsvorteile durch Kostenminderung entstehen. Weiterhin zeichnen sich funktionsfähige Netzwerke durch eine elektronische Daten- und Informationsverknüpfung aus. Dadurch wird auch im Netzwerk ein Informationsstand wie bei integrierten

Chancen	**Risiken**
Steigerung der strateg. Flexibilität	Verlust der Kernkompetenz
Risikoverteilung	Erschwerung strateg. Steuerung
Senkung der Produktionskosten	Einbuße strategischer Autonomie
Senkung der Koordinationskosten	Steigerung der Koordinationskosten
Abschöpfung v. Regelungsarbitrage	Sinkendes Mitarbeiter-Commitment
Interorganisationales Lernen	Unkontrollierter Abfluss von Wissen
Senkung des kapitalbedarfs	Verlust org. Identität

Tabelle 2.2.: Chancen und Risiken von Unternehmensnetzwerken [Syd99, S. 291]

Unternehmen erreicht. Der Fokus liegt auf einer ganzheitlichen Optimierung von Geschäftsprozessen über alle zwischenbetrieblichen Schnittstellen hinweg.

Im Einklang mit ihrer Stellung zwischen Markt und Hierarchie erfolgt die Koordination von Unternehmensnetzwerken durch Steuerungsmechanismen aus beiden Bereichen [Sie99b, S. 22 ff.]. In diesem Sinne kommen sowohl dezentrale Mechanismen preislicher Koordination als auch zentralisierte Planungsmechanismen zum Einsatz. Allerdings führen preisliche Änderungen nicht wie auf Märkten zu einem unmittelbaren Geschäftspartnerwechsel und die Planung erfolgt im Gegensatz zur Hierarchie in gemeinsamer Absprache der Teilnehmer. Da ein gewisses Maß an Führung die Stabilität von Netzwerken erhöht, übernehmen oft einzelne Teilnehmer die Rolle von Netzwerk-Managern, zum Teil als reine Information Broker [MS86], zum Teil aber auch auf Basis einer dominanten Stellung bei der Wertschöpfung.

Chancen und Risiken von Unternehmensnetzwerken Die beschriebenen Eigenschaften implizieren, dass mit dieser Organisationsform gleichsam Chancen als auch Risiken verbunden sind. Tabelle 2.2 fasst die in der Literatur beschriebenen potenziellen Wirkungen zusammen.[42]

Hierbei ist insbesondere zu vermerken, dass Unternehmensnetzwerke zwar das Potenzial besitzen, die Koordinationskosten im Ganzen zu senken, dies aber keine leichte Aufgabe ist. Vielmehr wird die Steuerung von Netzwerken als komplexen, oft polyzentrischen Systemen mit mehreren Steuerungszentren und partieller Selbststeuerung als schwierig beschrieben [SW99]. Dadurch ergibt sich die Gefahr der nur partiellen Systembeherrschung sowie einer möglichen Steigerung der Koordinationskosten. Informationstechnische Hilfsmittel können hier helfen, die Risiken zu verringern und die Chancen umzusetzen.

Management von Unternehmensnetzwerken In jedem Fall bedingt die Realisierung der Chancen und Vermeidung der Risiken von Unternehmensnetzwerken einen Wandel von Rolle und Funktion des Managements [Syd99, S. 294 ff.]. Strategien

[42] Vgl. z. B. [Sie99b, Syd99, Brü99].

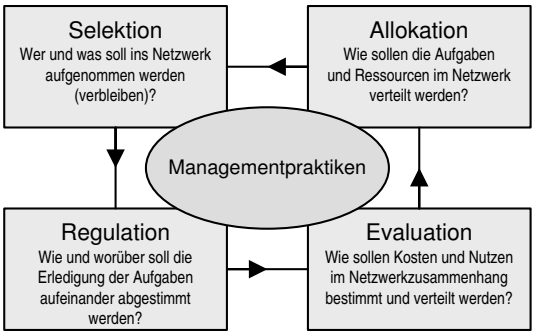

Abbildung 2.18.: Managementfunktionen bei N-Form (nach [Syd99, S. 295])

können dabei nicht mehr alleine für einzelne Unternehmen entwickelt und implementiert werden, sondern müssen auf das Kollektiv ausgerichtet sein. Daneben ergeben sich ganz neue Aufgaben wie die Ein- und Zuordnung der Teilnehmer innerhalb der Netzwerkstruktur zusammen mit der damit einhergehenden Klärung und Regelung der Machtverhältnisse. Allgemein kann das Management von Unternehmensnetzwerken als eine Erweiterung der Funktionsbereiche des innerbetrieblichen Managements (Planung, Organisation, Personal, Kontrolle) um *beziehungsspezifische* Aufgaben angesehen werden.

Eine generische und mittlerweile verbreitete Einteilung netzwerkspezifischer Managementaufgaben wird von Sydow und Windeler vorgeschlagen. Sie unterscheiden vier zentrale Funktionen des Managements interorganisationaler Beziehungen im Allgemeinen und von Netzwerken im Besonderen [SW94] (vgl. Abb. 2.18): (1) *Selektion* von Netzwerkpartnern, (2) *Allokation* von Aufgaben und Ressourcen, (3) *Regulation* von Zusammenarbeit im Netzwerk und (4) *Evaluation* der Teilnehmer, Beziehungen oder des ganzen Netzwerks. Die Selektionsfunktion des Netzwerkmanagements beinhaltet eine genaue Definition der Netzwerkdomäne sowie die Auswahl diesbezüglich geeigneter Teilnehmer. Diese Selektion muss im Laufe der Zeit immer wieder geprüft und ggf. angepasst werden. Die Allokationsfunktion bezieht sich demgegenüber auf die Konfiguration des Netzwerks im Sinne einer Zuteilung von Aufgaben, Ressourcen und Zuständigkeiten an die im Netzwerk befindlichen Unternehmen, wobei deren spezifische Kompetenzen zu berücksichtigen sind. Die (begrenzte) Fähigkeit zur Rekonfiguration ist hierbei entscheidend, da dies gerade die Flexibilität der Organisationsform ausmacht. Die Regulationsfunktion besteht im Wesentlichen aus Entwurf und Forcierung kooperativer Regeln für die Zusammenarbeit von Unternehmen im Netzwerk. Dies betrifft z. B. alle generellen vertraglichen Vereinbarungen oder das Konfliktmanagement, aber auch ein gemeinsames Informationsmanagement mitsamt der Festlegung eines integrierten Führungsinformationssystems (FIS) sowie insbesondere ein gemeinsames Knowledge-Management. Dabei ist zu beachten, dass die Flexibilität des Netzwerks auch hier die ständige Anpassung der Regeln bedingt, was

auch für den wichtigen Faktor der IV zu gelten hat. Die Evaluationsfunktion des Netzwerkmanagements bezieht sich explizit auf den Erfolg des Netzwerks als Ganzes. In diesem Sinne werden das gesamte Netzwerk, einzelne Beziehungen, der Beitrag einzelner Partner sowie auch die Selektions-, Allokations- und Regelungsverfahren selbst untersucht und ausgewertet.

In Ergänzung der sehr allgemein gehaltenen Funktionsbereiche des strategischen Netzwerkmanagements konzentriert sich Brütsch auf eher taktisch/operationale Aufgaben [Brü99, S. 22 f.], deren Gestaltung vor allem der strategischen Regulationsfunktion zugeschrieben werden können. Im Einzelnen führt er den effizienten Austausch von Geschäftsobjekten in zumeist IT-gestützter Form (z. B. EDI), die gemeinsame Bewirtschaftung von Beständen und evtl. Ressourcen in physikalischer und elektronischer Form (z. B. gemeinsame Datenbanken), die Verknüpfung und Optimierung der Geschäftsprozesse zwischen den Teilnehmern insbesondere in Hinblick auf organisatorische und technische Schnittstellen sowie den informationellen Austausch von Know-How an.

Systematisierung von Unternehmensnetzwerken Wie bereits erwähnt, treten konkrete Netzwerkorganisationen in einer großen Anzahl verschiedener Formen auf, die trotz der Gemeinsamkeit der beschriebenen generischen Merkmale sehr unterschiedlicher Natur sein können. Eine Differenzierung von Netzwerktypen ist daher für konkretere Aussagen äußerst wichtig. Dementsprechend wurden zur Systematisierung von Unternehmensnetzwerken eine Reihe verschiedener Klassifikations- und Typologisierungsansätze vorgeschlagen.

Oft werden zunächst nach Integrationsgrad [Brü99, S. 19] bzw. Ausrichtung [AE03, S. 36 ff.] des Netzwerks interne und externe bzw. interorganisationale Netzwerke unterschieden. Interne Netzwerke beschreiben weit verzweigte Geschäftsaktivitäten einzelner Unternehmen. Externe Netzwerke stellen hingegen Konglomerate mehrerer eigenständiger Unternehmen dar. Im Sinne der hier angestellten Überlegungen zu VDU wird die weiterführende Betrachtung sich auf die relevantere Form externer Netzwerke konzentrieren. Auch nach Einschränkung auf die externen Varianten vermerkt Sydow in [Syd99, S. 284] passend, dass *„die Möglichkeiten der Typologisierung von Netzwerken [...] grenzenlos (sind)“*. Aus diesem Grund beschränkt sich die Darstellung auf einige ausgewählte Differenzierungsansätze, die im Rahmen der vorliegenden Arbeit von Interesse sind. Diese Differenzierungsansätze werden in Tabelle 2.3 aufgelistet.

Die Merkmale der Beziehungsstabilität und der führungsmäßigen Steuerungsform stellen wesentliche Dimensionen zur Abgrenzung gängiger Netzwerkorganisationen dar. Sie werden z. B. von Sydow für eine Typologie externer Unternehmensnetzwerke herangezogen (vgl. Abb. 2.19). Im Hinblick auf die fokussierte Problemstellung der vorliegenden Arbeit werden darüber hinaus noch einige Merkmale hinzugenommen: Verschiedene Orte und Polyzentrizitätsgrade ergeben unterschiedliche Varianten in

Merkmal	**Erscheinungsformen**
Stabilität der Mitgliedschaft	- Stabile/Statische Netzwerke - Dynamische Netzwerke
Steuerungsform nach Führungsform	- Hierarchische Netzwerke - Heterarchische Netzwerke
Steuerungsform nach Ort	- Intern gesteuerte Netzwerke - Extern gesteuerte Netzwerke
Grad der Polyzentrizität	- Zentrierte Netzwerke - Dezentrierte Netzwerke
Sektorenzugehörigkeit der Mitglieder	- Industrielle Netzwerke - Dienstleistungsnetzwerke

Tabelle 2.3.: Verschiedene Klassifikationen interorganisationaler Netzwerke (Auswahl einer komplexeren Typologie von Sydow [Syd99, S. 285])

Bezug auf die Netzwerksteuerung. Eine Unterscheidung nach Sektorenzugehörigkeit gliedert zusätzlich die Netzwerkteilnehmer in Dienstleistungs- und Industrieunternehmen auf.

Bzgl. der (zeitlichen) Beziehungsstabilität können statische (bzw. stabile) und dynamische Formen auftreten. Dies wurde z. B. von Snow et al. in [SMC92] untersucht. In statischen Netzwerken ist dabei laut Snow oft eine Ungleichheit teilnehmender Unternehmen zu beobachten. Ein oder mehrere so genannte *fokale Unternehmen* üben dann die Gesamtführung des Netzwerks aus. Derartige Netzwerkorganisationen, die von ihrer Steuerungsform her hierarchisch sind, werden in der Literatur meist als *strategische Netzwerke* bezeichnet [Jar88, Syd92]. Tochterunternehmen und Lieferanten sind hier in hohem Maße durch Besitzverhältnisse, Managementvorgaben oder Dominanz gebunden. Die Struktur ist daher relativ inflexibel. Typische Beispiele solcher Netzwerke finden sich in der Automobilindustrie. Daneben treten sie aber laut Sydow auch immer öfter im Dienstleistungssektor auf [Syd92, S. 19 ff.]. Eine weitere, zum großen Teil statische Netzwerkform sind die so genannten *regionalen Netzwerke*, die sich in der Regel durch räumliche Agglomeration auszeichnen.[43] Sie werden meist durch kleinere gleichberechtigte Unternehmen einer Region gebildet und sind oft durch die potenzielle Realisierung von Größenvorteilen motiviert. Regionale Netzwerke sind idealtypischer weise polyzentrisch und werden heterarchisch durch gleichberechtigte Teilnehmer gesteuert.

Dynamische Netzwerke ändern im Gegensatz zu den statischen Varianten relativ häufig ihre Teilnehmerstruktur. Die Formation richtet sich oft im Sinne einer unternehmensübergreifenden Projektorganisation an wechselnde Ziele und Aufgaben, was allgemein als *Projektnetzwerk* bezeichnet wird. Der ständige Wechsel führt dabei zu hohen Ansprüchen an Unternehmenskultur und Informationstechnik. Bezüglich ihrer Steuerungsform sind solche Projektnetzwerke zumeist hierarchischer Natur (z. B. in der Bau- und Filmindustrie), wobei aber auch heterarchische Varianten

[43] Vgl. z. B. [Syd99, S. 288 f.].

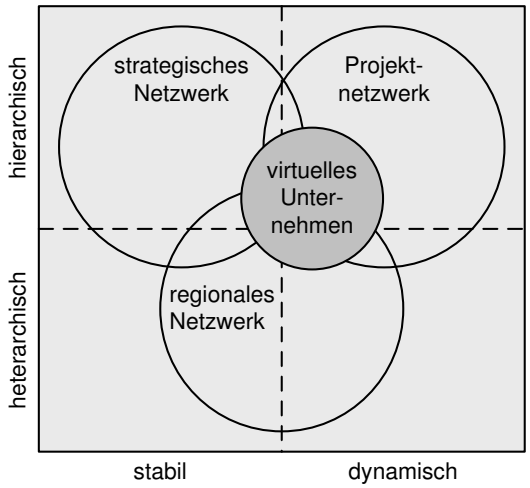

Abbildung 2.19.: Typologie externer Unternehmensnetzwerke [Syd99, S. 287]

denkbar sind. Eine orthogonale Unterteilung dynamischer Netzwerke kann bzgl. ihrer Polyzentrizität erfolgen. In zentrierten dynamischen Unternehmensnetzwerken finden sich oft so genannte *Broker* [MS84]. Ein solcher Vermittler repräsentiert dann das Netzwerk in einheitlicher Weise nach außen. Für spezifische Projekte wählt er die optimalen Teilnehmer aus einem Pool beteiligter Unternehmen aus. Dezentrierte dynamische Netzwerke zeichnen sich hingegen durch eine demokratische Verteilung der Führungsverantwortung und geringe gegenseitige Abhängigkeiten aus. Jedes Unternehmen kann das volle Spektrum der gemeinsam vermarkteten Leistung anbieten und Projekte im Netzwerk leiten.

Generell ermöglicht der Einsatz *interorganisationaler Informationssysteme* immer dynamischere Netzwerkstrukturen. Die rasante technische Entwicklung und fortschreitende Standardisierung kann als eine Triebfeder organisatorischer Unternehmensvernetzung angesehen werden.[44] Dies geht so weit, dass so genannte *virtuelle Unternehmen* entstehen, die lediglich von ihrer Möglichkeit her bestehen. Bei Bedarf vernetzen sich dann reale Unternehmen unter massivem Einsatz von Informationstechnik zu einem temporären Projektnetzwerk.

2.3.2.2. Virt. Unternehmen (VU) zur Dienstleistungsproduktion

Bevor der spezifische Netzwerktyp des virt. Unternehmens näher beschrieben wird, soll ein kurzer Diskurs zum Begriff der Virtualität erfolgen. Dieser wird im Duden ganz allgemein als *„von der Möglichkeit her vorhanden“* definiert. Als „virtuell“ werden in der Regel Varianten bzw. Abbilder eines spezifischen realen Objekts bezeichnet, denen eine (sonst übliche) physikalische Komponente oder Präsenz fehlt, die aber auf

[44] Vgl. z. B. [PRW03].

Basis von Zusatzfunktionalität einen gewissen Zusatznutzen erzeugen. So definiert Scholz in [Sch96] folgende vier Bedingungen für ein virtuelles Objekt:

1. für originales und virt. Objekt sind konstituierende Charakteristika spezifiziert,
2. dem virtuellen Objekt fehlen verschiedene physikalische Attribute,
3. es besitzt stattdessen spezielle Zusatzspezifikationen und
4. führt (gerade durch Wegfall der phys. Attribute) zu erweiterten Nutzeffekten.

Das Konzept der Virtualität lässt sich grundsätzlich in vielen Disziplinen wie Physik, Informatik und auch der Ökonomie anwenden. Bei Letzterer wurden u. a. virt. Produkte, Wertschöpfungsketten, Organisationen (im institutionellen Sinn) und Unternehmen in Betracht gezogen.

Virt. Unternehmen (VU) als Organisationsform wurden in der Literatur seit den Ausführungen von Davidow und Malone [DM92] sehr ausführlich und breit diskutiert.[45] Brütsch untersucht in [Brü99, S. 45 ff.] eine Reihe von Definitionen und analysiert die Organisationsform auf ihre Virtualität im Sinne von Scholz: demnach zeichnen sich VU dadurch aus, dass sie

1. sich als zielbewusst koordinierte Tätigkeiten von mehreren Parteien mit einheitlichem (wirtschaftlichem) Ziel definieren lassen,
2. nicht über die physikalischen Komponenten herkömmlicher Organisationsstruktur, Gebäude, zentraler Stabsfunktionen und Rechtsform verfügen,
3. spezielle Zusatzspezifikationen bzgl. vertrauensbasierter Kooperation, Dezentralisierung von Kompetenzen, integrierter IT/IV und einheitlicher Außenwirkung aufweisen sowie
4. Nutzeffekte in Bezug auf Geschwindigkeit, Flexibilität, Teilung von Infrastruktur, Kosten, Risiko und Marktzugängen, Konfigurationsmöglichkeiten optimierter Geschäftsprozesse und Verkauf kompletter Lösungen realisieren.

Damit ist die Bezeichnung der Virtualität im Sinne von Scholz gerechtfertigt.

Im Weiteren soll die im deutschsprachigen Raum mehrfach aufgegriffene Definition von Arnold in der abgewandelten Form von Mertens als Basis dienen [MGE98, S. 3]:

Definition 8 (virt. Unternehmen) *Ein virt. Unternehmen (VU) ist eine Kooperationsform rechtlich unabhängiger Unternehmen, Institutionen und/oder Einzelpersonen, die eine Leistung auf Basis eines gemeinsamen Geschäftsverhältnisses*

[45] Vgl. z. B. [AFH+95, Brü99, KRR97, MGE98].

erbringen. Die kooperierenden Einheiten beteiligen sich an der Zusammenarbeit vorrangig mit ihren Kernkompetenzen und wirken bei der Leistungserstellung gegenüber Dritten wie ein einheitliches Unternehmen. Dabei wird auf Institutionalisierung zentraler Managementfunktionen zur Gestaltung, Lenkung und Weiterentwicklung des VU weitgehend verzichtet und der notwendige Koordinations- und Abstimmungsbedarf durch geeignete Informations- und Kommunikationssysteme gedeckt. Das VU ist mit einer Mission verbunden und endet mit dieser.

Die Abgrenzung von anderen N-Formen gründet sich im Wesentlichen auf zwei eng zusammenhängende Merkmale:

1. Verzicht auf Institutionalisierung zentraler Managementfunktionen,

2. Rolle der IT als diesbezüglicher „Enabling Factor".

Diese Abgrenzung ist dabei jedoch als äußerst fließend zu bezeichnen. Tatsächlich sind VU nach obiger Definition in der Praxis äußerst selten zu finden. Stattdessen setzen immer mehr Unternehmen einzelne Aspekte davon um. Zur weiteren Veranschaulichung virtueller Unternehmensformen können Stufenmodelle dienen, von denen zwei Varianten dargestellt werden sollen.

Im ersten Modell, das auf Arnold et al. zurückgeht[46], wird die institutionelle Perspektive der obigen Definition aufgegriffen. Es unterscheidet neben einer nicht virtuellen Ausgangsform vier Typen von VU. Abbildung 2.20 veranschaulicht die fünf Stufen (0-4). Die Darstellung zeigt Wertschöpfungsketten der Leistungserstellung an zwei Orten (A,B). Diese Wertschöpfungsketten sind in fünf Teilprozesse (Kästen A-E) untergliedert. Nummern markieren autonome Unternehmen, die jeweils einzelne Teilprozesse im Rahmen des VU übernehmen.

- *Stufe 0* markiert den Grundzustand, der durch Konzentration der Teilprozesse auf Ort und Unternehmen bestimmt ist.

- In *Stufe 1* erfolgt erstmals eine zunächst unternehmensinterne virtuelle Organisation. Unternehmen 1 hält Teilprozesse an nur einem Ort vor. Der komplette Fertigungsprozess ist an den Standorten nur noch durch die Möglichkeit der ortsunabhängigen Vernetzung von Teilprozessen *virtuell* vorhanden.

- Die Auslagerung von Teilprozessen an autonome Partner (2, 3) markiert den Übergang zu *Stufe 2*. Die Partnerunternehmen konzentrieren sich auf die wertschöpfenden Prozessanteile im Sinne ihrer jeweiligen Kernkompetenzen.

- *Stufe 3* bindet dann Kunden und Lieferanten in den Leistungsprozess ein. In [MGE98] wird ab hier stattdessen von Konsumenten und Experten gesprochen.

[46] Vgl. [AFH+95], [MGE98], [Fai98].

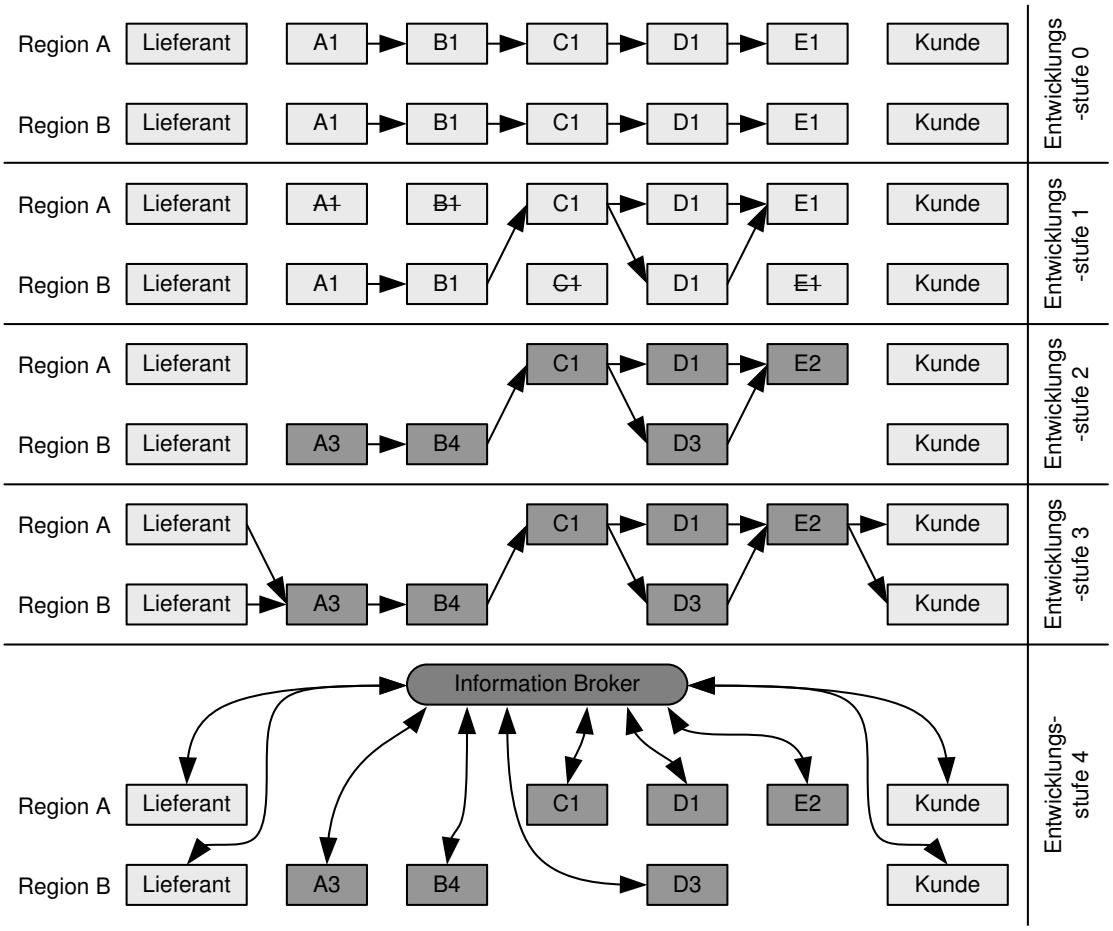

Abbildung 2.20.: Entwicklungsstufen virtueller Unternehmen [AFH+95]

- Am Ende steht in *Stufe 4* die Auflösung eines festgelegten Leistungsprozesses zu Gunsten der voll flexiblen und bedarfsgerechten Koordination durch einen Information Broker.

Venkatraman und Henderson schlagen zur Erklärung von VU ein weiteres Stufenmodell vor, das nicht den institutionalen, sondern den funktionalen Organisationsbegriff in den Mittelpunkt stellt [VH96]. Demnach sind VU durch gewisse Fähigkeiten der virtuellen Organisation gekennzeichnet, die aus dem Einsatz von Informationstechnik hervorgehen. Diese Fähigkeiten lassen sich in drei Bereiche einteilen, die als Vektoren eines Raumes möglicher VU angesehen werden.

Im Einzelnen sind das *Marktinteraktion*, *Kompetenzentwicklung* und *Arbeitskonfiguration*. Für jeden Merkmals-Vektor werden drei Ausprägungen unterschieden, die eine fortschreitende Virtualisierung markieren (vgl. Tab. 2.4). Die Autoren betonen dabei die Abhängigkeit der Vektoren, die sich nur gemeinsam verwirklichen lassen.

	Marktinteraktion	**Kompetenzentwicklung**	**Arbeitskonfiguration**
Stufe 1	Marktzugang	Effiziente Beschaffung	Effiziente Administration
Stufe 2	Interaktive Dienstleistungen	Effektive Kooperation	Verschieb. soz. Bezüge
Stufe 3	Produkt- & Lösungsentwickl.	Kompetenzaufbau	Neudef. soz. Bezüge

Tabelle 2.4.: Drei interdependente Vektoren der Virtualität

Im Folgenden wird ein Überblick der Vektoren gegeben:

- *Marktinteraktion* – Erweiterte Fähigkeiten zur Marktinteraktion beziehen sich auf die Kundeninteraktion im Zuge des Absatzes. Dies beginnt mit der IT-gestützten Integration von Angeboten und Absatzmärkten der Netzwerkunternehmen (Marktzugang). Mit zunehmender Virtualisierung erfolgt eine intensivere Kommunikation zwischen Kunden und Netzwerkunternehmen auf allen Ebenen der Wertschöpfungskette im Sinne einer proaktiven Produktentwicklung (interaktive Dienstleistungen). Schließlich wird der Kunde unmittelbar und aktiv in die Leistungserstellung einbezogen (Produkt- & Lösungsentwicklung).

- *Kompetenzentwicklung* – Kompetenzentwicklung markiert eine parallele Tendenz bei der Lieferanteninteraktion im Zuge der Beschaffung. Der verstärkte Einsatz von IV-Mechanismen erlaubt zunächst die optimierte Auswahl der besten Lieferanten (effiziente Beschaffung). Durch erweiterte Interaktionsmöglichkeiten wird dann die Integration von Kernkompetenzen der Netzwerkunternehmen ermöglicht (effektive Kooperation). Die Entwicklung gipfelt in der (umstrittenen) Vision eines freien Austauschs von Kompetenzen zwischen den Netzwerkunternehmen (Kompetenzaufbau).

- *Arbeitskonfiguration* – Schließlich beschreibt die Dimension der Arbeitskonfiguration den Aufbau von Fähigkeiten zur Organisation des Produktionsprozesses im Sinne optimierter Geschäftsprozesse. Dies beginnt bei der Nutzung von IV zur Effizienzsteigerung einzelner interner Funktionsbereiche (effiziente Administration). Erweiterte Informations- und Interaktionsmöglichkeiten erlauben darüber hinaus die Dezentralisierung von Entscheidungskompetenzen (Verschiebung sozialer Bezüge). Die Weiterentwicklung dieser Selbstständigkeit (auch im Sinne der Rechtsform) führt zu autonomen Organisationseinheiten, die sich zu optimierten Geschäftsprozessen konfigurieren lassen (Neudefinition sozialer Bezüge).

Die funktionale Perspektive von Venkatraman und Henderson lässt Virtualität als orthogonalen Aspekt institutionaler Unternehmensnetzwerke erscheinen, der sich weitgehend unabhängig vom konkreten Netzwerktyp darstellt [Syd99, S. 289]. Dementsprechend können zur Typisierung von VU im Wesentlichen die Klassifikationsmerkmale der generellen N-Form verwendet werden.

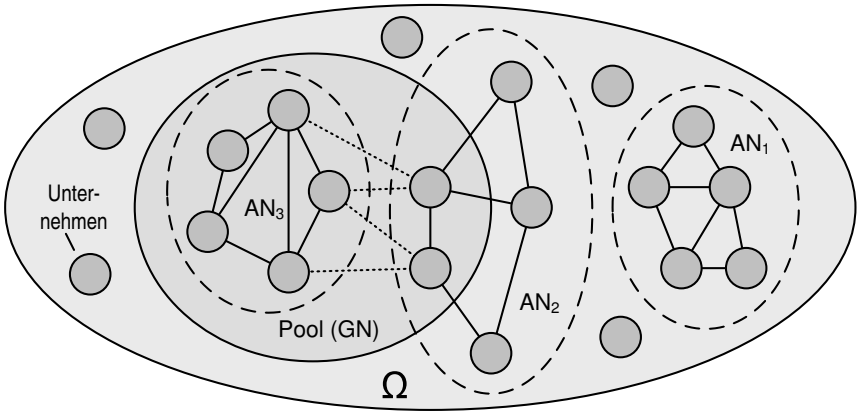

Abbildung 2.21.: Typen von VU nach Dauerhaftigkeit der Beziehungen (nach [CG00])

Oft wird im Zusammenhang mit virt. N-Formen das generelle Merkmal der Beziehungsstabilität herangezogen. Hierbei ist zu bemerken, dass VU zwar Projektnetzwerke darstellen, die für begrenzte Zeiträume gebildet werden, jedoch bedarf die kurzfristige Konfiguration von Projekten einer Vertrauensbasis, die sich nur über längere Zeit bildet. Daher liegen VU meist längerfristige statische Netzwerke zugrunde. Diese bilden einen Pool potenzieller vertrauenswürdiger Projektpartner mit spezifischen Kernkompetenzen. Abbildung 2.21 veranschaulicht dies, wobei Ω die Menge aller Unternehmen und GN einen Pool darstellt. Auf dieser Basis kann dann zwischen drei Typen von VU unterschieden werden: solche, die sich komplett aus Netzwerkunternehmen des Pools zusammensetzen (Typ3, AN_3); solche, die außenstehende Unternehmen zur Kompensation fehlender Kompetenzen oder Kapazitäten hinzuziehen (Typ2, AN_2) sowie solche, hinter denen kein Pool steht (Typ1, AN_1).

Ein weiteres oft herangezogenes Klassifikationsmerkmal betrifft die Steuerungsform und vor allem die Polyzentrizität des VU. Genauer gesagt werden VU mit so genannten *Broker* bzw. *Promoter* oder auch *Leader* herausgestellt [MGE98, S. 12 ff.]. In VU mit Broker/Promoter dienen diese in erster Linie zur homogenen Repräsentation nach außen. Sie umschreiben dabei eine Rolle, deren Aufgaben von genau einem Netzwerkunternehmen (zentralisiert) oder von mehreren (polyzentrisch/dezentralisiert) wahrgenommen werden können. Diese Aufgaben entsprechen im Wesentlichen einer auf die spezielle Situation von VU zugeschnittenen Form des Netzwerkmanagements (s. o.) mit den Funktionsbereichen der Selektion, Allokation, Regulation und Evaluation. Die führungsmäßige Steuerungsform ist dabei als heterarchisch zu bezeichnen. Im hierarchischen Fall, bei dem ein oder mehrere fokale Unternehmen auftreten, sprechen Mertens et al. statt von einer Broker- von einer Leader-Rolle.

In Bezug auf Voraussetzungen virt. Organisationsformen stellt Picot drei Bereiche heraus [PN98]. Demnach sind virt. Organisationen (und VU) als Kombination

aus Mensch (Qualifikation, Motivation), Technik (Hardware, Software, Systeme) und Organisation (Prozesse, Strukturen) zu verstehen. Bei diesen sind auch die Voraussetzungen für das reibungslose Funktionieren zu suchen.

Das virt. Dienstleistungsunternehmen Wie in Untersektion 2.2 ausführlich hinterlegt und schon in der Einleitung dieser Untersektion angesprochen, sind an die Organisation von DLU laut Picot drei wesentliche Anforderungen zu stellen [PN98]:

- Kundenorientierte Konfiguration eines Dienstleistungsprodukts als weitgehend vertrauensbasiertes Zukunftsversprechen
- Unterstützung eines hohen Grades an Immaterialität bei der Leistungserstellung
- Unterstützung eines hohen Grades an Interaktion mit dem Kunden

Wie ebenfalls gezeigt wurde, steht die Bedeutung kundenorientierter Prozessketten bei der Dienstleistungsproduktion im grundsätzlichen Konflikt mit der hierarchisch-funktionalen Organisation der klassischen Sachgüterproduktion, die sich im Sinne von Ressourcenoptimierung auf betriebliche Funktionen konzentriert. Stattdessen werden für die Dienstleistungsproduktion andere Organisationsformen benötigt, die eine direkte, problembezogene Ausrichtung am Dienstleistungsprozess ermöglichen.

Ein erster wesentlicher Schritt in diese Richtung ist die ebenfalls bereits ausführlich dargestellte Prozessorganisation. Diese konzentriert sich im Sinne der Optimierung des Kundennutzens auf den Dienstleistungsprozess. Jedoch kann die Prozessorganisation nicht als ausgesprochen flexibel gelten. Sie ist auch nicht im besonderen Maße zur interaktiven Erstellung immaterieller Leistungen ausgelegt. Fügt man jedoch der Prozessorganisation die übergeordnete Dimension der Netzwerkorganisation hinzu und wendet dort insbesondere noch Prinzipien der Virtualisierung an, so ergibt sich eine Organisationsform, die allen Anforderungen entspricht.

Nach der Organisation von Dienstleistungsprozessen ist als nächste Zwischenstufe die Klasse von Unternehmensnetzwerken zu nennen, die sich durch Einschränkung der Sektorenzugehörigkeit von Netzwerkunternehmen auf den Dienstleistungssektor ergibt. Derartige *Dienstleistungsnetzwerke* werden in [AE03, S. 46] wie folgt definiert:

Definition 9 (Dienstleistungsnetzwerk) *Ein Dienstleistungsnetzwerk bezeichnet die auf die Erbringung einer Dienstleistung ausgerichtete Zusammenarbeit von mehr als zwei rechtlich selbstständigen Parteien, die jedoch zumindest in Bezug auf den Kooperationsbereich wirtschaftlich nicht unabhängig sind. Die Beziehungen zwischen den die Dienstleistung erbringenden Unternehmen gehen dabei über rein marktliche Beziehungen hinaus, d. h. dass sie für eine gewisse Dauer angelegt sind und die Dienstleistung von den Unternehmen nicht nur einmalig erbracht, sondern dauerhaft am Markt angeboten wird. Ebenso findet ein Austausch von Ressourcen zwischen den beteiligten Netzwerkpartnern statt.*

Eine weitere Optimierung lässt sich durch die Virtualisierung von Dienstleistungsnetzwerken erreichen.[47] Zunächst ist der Umstand zu betonen, dass Dienstleistungen eng mit virtuellen Produkten in Verbindung stehen.[48] Aus Kundensicht bieten virtuelle Produkte in besonderer Weise die Möglichkeit zur individuellen Zusammenstellung eines *virtuellen Leistungsnetzes*. Diese Leistungsbündel sind immateriell sowie stark interaktiv und haben daher grundsätzlich Dienstleistungscharakter.[49] Aus Anbietersicht können andersherum Leistungsbündel mit Dienstleistungscharakter auf Grund ihrer Immaterialität relativ leicht virtualisiert werden und bilden dann *virtuelle Leistungsprozesse*. Die Virtualität der Dienstleistungen kommt in der Folge der Virtualisierung von Organisationsformen der DLU zugute. Dies erleichtert nämlich den problemorientierten und bedarfsgerechten Zusammenschluss autonomer Organisationseinheiten des Dienstleistungsnetzwerks. Bei dieser Form der virtuellen Netzwerkorganisation kann man von einem virt. Dienstleistungsunternehmen (VDU) sprechen.

In Bezug auf die oben genannten organisatorischen Anforderungen kann dem VDU zunächst eindeutig die Fähigkeit zur kundenorientierten Konfiguration von Dienstleistungsprodukten attestiert werden. Dies leitet sich direkt aus den Eigenschaften der N-Form im Allgemeinen und VU im Besonderen ab. Unternehmensnetzwerke verfügen in Relation zu den einzelnen Netzwerkunternehmen nicht nur über mehr Kapazität, sondern auch über ein höheres Potenzial an Ressourcen und Kompetenzen, was schon zu erhöhter Dienstleistungsqualität führt. In VU liegen die Kernkompetenzen der Netzwerkunternehmen dann als modulare Geschäftsprozesse vor. Diese können dank der Möglichkeit zur flexiblen Koordination beteiligter Akteure bedarfsgerecht zu Dienstleistungsprozessen kombiniert werden. Darüber hinaus lassen sie sich auch danach noch flexibel in Hinblick auf Quantität und Qualität anpassen. Dem VDU ist es somit möglich, sich ständig am Kundenproblem auszurichten und geforderte Leistungen schnell und flexibel zur Verfügung zu stellen.

Die Flexibilität und die damit einhergehende enorme Dynamik von VU werden – wie im letzten Abschnitt gezeigt – durch den Verzicht auf institutionale Managementfunktionen sowie einen dies kompensierenden massiven IT-Einsatz ermöglicht (was freilich nicht den Verzicht auf Managementfunktionen, sondern deren Anpassung an die veränderte Organisation bedeutet). Die fortschreitende Entwicklung der IV ermöglicht es dabei nicht nur N-Formen zu virtualisieren, sondern kommt auch in direkter Weise der Leistungserstellung im DLU zugute: Die im VU zum Einsatz kommende IT erlaubt ganz allgemein den Austausch leistungsbezogener Daten und Informationen zwischen Netzwerkunternehmen und Kunden, was die immaterielle Leistungserstellung unterstützt – im Extrem bis hin zum virtuellen Produkt (z. B. Tele-Leistungen). In diesem Sinne kann die zweite Anforderung nach Unterstützung

[47] Vgl. zum Folgenden [PN98].
[48] Siehe obige Diskussion zur Virtualisierung.
[49] Vgl. Sektion 2.2.1.2.

eines hohen Grades immaterieller Leistungserstellung als erfüllt gelten. Des Weiteren unterstützt die IT im VU die intensive Interaktion der Netzwerkunternehmen (z. B. mittels E-Mail, Web-Portalen oder auch den später noch eingehend zu betrachtenden Web Services). Dieses Interaktionspotenzial ist freilich nicht auf das Netzwerk beschränkt. Damit ist auch die dritte Anforderung erfüllt; die Unterstützung eines hohen Grades an Interaktion mit dem Kunden.

2.3.3. Koordination virt. Produktionsnetzwerke

Im vorangegangenen Abschnitt wurde die Vorteilhaftigkeit der Organisation von DLU als virtuelle Unternehmensnetzwerke dargestellt. Nun soll untersucht werden, wie sich diese Organisationsform auf das Management der Dienstleistungsproduktion auswirkt.

Unternehmensnetzwerke werden unter dem Blickwinkel der Produktionswirtschaft oft als *Produktionsnetzwerke* bezeichnet. Der erste Teil dieses Abschnitts führt diesen Begriff näher ein und betont besonders dessen virtuelle Variante. Des Weiteren hebt er die Aufgaben des Produktionsmanagements hervor.

Ein wesentlicher Teilaspekt des Produktionsmanagements ist die Koordination der arbeitsteiligen Tätigkeiten. Der zweite Teil dieses Abschnitts geht daher auf die Probleme bei der Regelung der Koordination im VU ein und skizziert einen Lösunsansatz.

2.3.3.1. Einführung virt. Produktionsnetzwerke

Im Folgenden soll sich die Betrachtung noch der Leistungserstellung in VDU widmen. In diesem Sinne ist der Begriff des *virt. Produktionsnetzwerks* einzuführen und auf einige spezifische Aspekte des Produktionsmanagements hinzuweisen. Der allgemeinere Begriff des generellen *Produktionsnetzwerks* bezeichnet eine spezielle Form von Unternehmensnetzwerken, bei dem die Zusammenarbeit der beteiligten Unternehmen primär auf Produktionsprozesse bezogen ist [Sch03, S. 33 ff.]. Die einzelnen Produktionssysteme der Netzwerkunternehmen, die für sich natürlich immer in komplexe Wirkzusammenhänge des natürlichen, sozio-technischen und sozio-ökonomischen Umsystems eingebunden sind, bilden dabei aus produktionstheoretischer Sicht einen gemeinsamen *Produktionsverbund*. Ein Produktionsverbund kennzeichnet eine gewisse Entwicklungsstufe immer weitergehend disaggregierter Produktionssysteme, die von einzelnen Mensch-Maschine-Systemen bis hin zu virtuellen Unternehmen reichen (vgl. Abb. 2.22).

Konkreter kann der (nicht einheitlich definierte) Begriff des Produktionsverbunds im Wesentlichen dadurch charakterisiert werden, dass räumlich getrennte Produktionsstätten (im Folgenden abstrakter als *Produktionseinheiten* bezeichnet) Leistungen auf gleicher Fertigungsstufe austauschen und dabei auf gemeinsame knappe Ressourcen zurückgreifen. Die Produktionseinheiten agieren also nicht unabhängig voneinander.

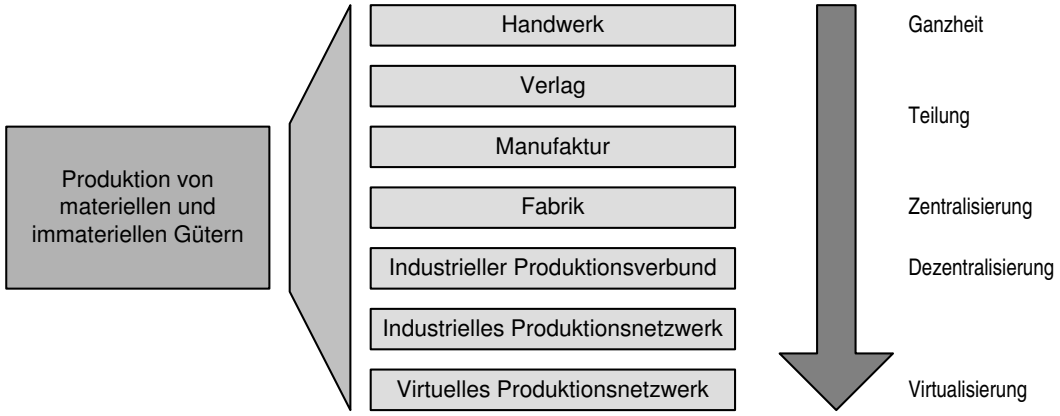

Abbildung 2.22.: Ausprägungen von Produktionssystemen (nach [Sch03, S. 35 ff.])

Sie sind viel mehr in Bezug auf ihre Produktionstechnologien, -prozesse und spezifischen Leistungen im Sinne eines gemeinsamen Produktes verknüpft. Daher sind ihre Material- oder Informationsflüsse über verschiedene Schnittstellen gekoppelt.

In Erweiterung des Produktionsverbunds, der im Allgemeinen als generelles Subsystem eines Unternehmens charakterisiert wird, fokussiert der Begriff des Produktionsnetzwerks die Organisation der Produktionseinheiten in einem Netzwerk. Bei den möglichen N-Formen eines Produktionsnetzwerks gehen die Meinungen auseinander. Wenn ein klassischer Produktionsbegriff im Sinne industrieller Güterproduktion vorliegt, werden meist hierarchisch-statische Netze (d. h. strategische Netzwerke) im Sinne einer vertikalen Integration der Produktion oder hierarchisch-dynamische Netze (d. h. operationale- oder Projektnetzwerke) im Sinne einer horizontalen Integration der Produktion als mögliche Formen genannt [EST00]. Bezieht sich der Produktionsbegriff jedoch auf weitgehend immaterielle Güter wie Leistungsbündel mit hohem Dienstleistungsanteil oder virtualisierte Produkte, so werden auch virtuelle Unternehmen als Produktionsnetzwerke in Betracht gezogen [PN00]. In diesem Fall spricht man auch von virt. Produktionsnetzwerken (VPN).[50]

Produktionsmanagement im VU Durch ihre besonderen, teils produktionstechnisch und teils organisatorisch bedingten Merkmale stellen Produktionsnetzwerke im Allgemeinen und deren virtuelle Form insbesondere auch erweiterte Anforderungen an das *Produktionsmanagement*. Dieses wird dabei klassischerweise in *Produktionsplanung und -steuerung (PPS)* unterteilt. Bei der Betrachtung von Produktionsnetzwerken sind darüber hinaus zwei Ebenen zu unterscheiden:

1. die *Netzwerkebene* mit der zentralen Aufgabe der Abstimmung innerhalb des Netzwerks sowie ggf. zwischen Netzwerk und Auftraggeber,

[50] Vgl. z. B. [CG00].

2. die Ebene der Produktionseinheiten (hier im Folgenden als *Knotenebene* bezeichnet) im Rahmen der Planung und Leistungserstellung.

Die Produktionsplanung kann wiederum grob in die *Produktionsprogrammplanung* und die *Produktionsprozessplanung* unterteilt werden. Während die Produktionsprogrammplanung die Produkte im Sinne von Art, Quantität, Qualität etc. bestimmt, legt die Produktionsprozessplanung deren konkrete Erstellung fest.[51] Auf Netzwerkebene überschneidet sich die Produktionsplanung mit dem Netzwerkmanagement,[52] und zwar vor allem mit Allokation und Regulation. Picot betont dabei, wie sehr die Produktionsplanung von der Netzwerkorganisation, d. h. von der Existenz einer Netzwerkebene profitiert [PN00]: ohne Rücksicht auf Ressourcen oder Kapazitäten kann das Produktionsprogramm komplett in Hinblick auf Kundenanforderungen ausgelegt werden. Daraufhin kann eine optimierte Konstruktion individueller Produktionsprozesse in problemorientierter und bedarfsgerechter Weise jeweils aus den am besten geeignetsten Produktionseinheiten erfolgen (auch als *Enterprise Engineering* oder *Konfigurationsmanagement* bezeichnet). Auf Knotenebene kann sich die Produktionsplanung hingegen weitestgehend auf die entsprechenden Aufgaben der klassischen PPS beschränken.

Ein Beispiel für die Planung von klassisch-industriellen Produktionsnetzwerken ist bei Eversheim zu finden [EST00]. Er schlägt eine fünfstufige Vorgehensweise zur Aufgliederung und Zuteilung der Produktion im Netzwerk vor:

1. Festlegung von Produktionsprogramm und notwendigen Produktionsprozess- bzw. Kapazitätsbedarfen (so genannte Auftragsmodule) sowie Ableitung von Zielen und Restriktionen zur Umsetzung und zum Betrieb des Netzwerks,

2. Grobplanung mit Vorauswahl und Bewertung von Produktionseinheiten anhand ihrer Kapazitätsangebote (so genannte Produktionsmodule),

3. Feinplanung der Logistik im Netzwerk,

4. Feinplanung der Produktion in den Produktionseinheiten und

5. letztendliche Konfiguration auf Netzwerkebene.

Es ist festzuhalten, dass derartig gestaltete Produktionsnetzwerke zu einem relativ hohen Grad formalisiert sind. Sie sind zudem eher längerfristig als flexibel ausgerichtet.

Für virt. Produktionsnetzwerke entwickeln verschiedene Arbeiten Planungsmethoden, die auf marktorientierten Mechanismen der Allokation beruhen. Hier sei der Ansatz von Corsten erwähnt [CG00]: Er schlägt zur Produktionsprogrammplanung auf Netzwerkebene eine Broker-Rolle vor. Der Broker wickelt dann auch im Zuge der

[51] Zur Prozessgestaltung siehe Sektion 2.3.1.2.
[52] Siehe Sektion 2.3.2.1.

Produktionsprozessplanung (genauer: bei der Stellenzuteilung bzw. Konfiguration des Netzwerks) die Auswahl von Produktionseinheiten über einen organisationsinternen Markt ab. Dieser zweite Ansatz zur Produktionsplanung kann als wesentlich flexibler bewertet werden. Er wird – wie so typisch für virtuelle Organisationsformen – erst durch IT-Einsatz möglich: Die Basis bildet eine kooperative Problemlösung im VU mittels MAS (siehe dazu auch [Kir95]). Im Ansatz von Corsten werden durch MAS Auktionen realisiert. Allerdings lässt Corsten dafür wesentliche Fragen der Integration zwischen Produktionseinheiten offen.

Im Folgenden soll die Produktionsplanung, die im Wesentlichen eine betriebswirtschaftliche Fragestellung darstellt, nicht weiter verfolgt und stattdessen die Produktionssteuerung fokussiert werden. Die Produktionssteuerung teilt sich in die *Veranlassung*, *Überwachung* und *Sicherung* der Produktion mit dem Ziel, das Produktionsprogramm durchzusetzen. Die Veranlassung setzt dabei einzelne Leistungsprozesse in Gang. Die Überwachung gleicht den tatsächlichen Zustand der laufenden Produktion mit geplanten Vorgaben ab. Die Sicherung reagiert kompensatorisch auf Störungen.

Eversheim beschreibt die Produktionssteuerung für industrielle Produktionsnetzwerke als Teilaufgabe (neben der Optimierung) innerhalb der *Betriebsphase*, die er als Zeitraum zwischen Veranlassung und Abschluss von Aufträgen definiert [EST00, S. 50 f.]. Da industrielle Serienfertigung unterstellt wird, ist die Planung des Netzwerks zum Zeitpunkt des Auftragseingangs schon abgeschlossen. Der Produktionsprozess ist als Ganzes gestaltet und in Form von Auftragsmodulen an einzelne Produktionseinheiten verteilt worden. Die Produktionssteuerung beinhaltet dann zunächst verschiedene Teilaufgaben der Auftragsveranlassung (im Einzelnen sind das Auftragsfreigabe, Termin-, Kapazitäts-, Verfügbarkeitsfeinplanung und Arbeitsverteilung). Diese sind sowohl auf Netzwerkebene als auch auf Knotenebene durchzuführen: Die Produktionseinheiten steuern ihre wohldefinierten Teilprozesse relativ autonom. Auf der Netzwerkebene verbleiben Steuerungsaufgaben in Bezug auf die übergeordnete Koordination sowie übergeordnete Produktionsprozesse (z. B. Transport- und Lagerprozesse). Zur Erkennung von Optimierungsbedarf des langfristig ausgelegten Netzwerks empfiehlt Eversheim die Implementierung eines übergreifenden Kontroll- und Regelmechanismus in Form eines Regelkreises.

Die Leistungserstellung in virt. Produktionsnetzwerken hat hingegen den Charakter einer Auftragsfertigung. Hier erfolgt die Planung erst nach Eingang eines Auftrags. Bei Corsten beschränkt sich die Produktionssteuerung auf Netzwerkebene dann mangels Betrachtung von Integrationsaspekten zwischen den Produktionseinheiten weitgehend auf die Überwachung [CG00, S. 273 ff.]. Bei der auf Netzwerkebene vorliegenden Grobplanung wird lediglich eine gleichermaßen grobe Initialisierung von Leistungsprozessen der Produktionseinheiten mittels Laufmarken (Token) als übergeordnete Auftragsveranlassung in Betracht gezogen. Dies dient allerdings primär der Rückmeldung von Produktionskennzahlen. Eine Sicherungsfunktion auf

Netzwerkebene wird ebenfalls nur am Rande erwähnt. Auf Knotenebene wird auf die allgemeine Verfahrensweise normaler PPS verwiesen. Die Konzeptionierung der Prozesssteuerung in virt. Produktionsnetzwerken ist hier also insgesamt als knapp zu bezeichnen. Ein Grund dafür ist zweifellos in dem bei virt. Organisationsstrukturen konstituierenden Merkmal fehlender Institutionalisierung zentraler Managementfunktionen zu sehen. Das Beispiel ist symptomatisch für den Umstand, dass unter diesen Prämissen eine dezentrale Koordination bzw. Führung bisher wenig untersucht wurde und daher stattdessen oft Konzepte der Selbstorganisation herangezogen werden, wie im Rest dieses Abschnitts zu zeigen sein wird.

Auch Picot et al. unterscheiden bei der Steuerung von Produktionsnetzwerken verschiedene Fälle: je nach Art der Produktionsprozesse unterscheidet Picot die Steuerung von automatisierten, elektronischen Produktionsprozessen sowie von Produktionsprozessen in Ad-hoc- oder Broker-Netzwerken [PN00]. Im ersten Fall, in dem die Prozesse vollständig vorgegeben und programmiert sind, spricht er von einem „interorganisationalen Fließband“, bei dem die einzelnen Fertigungsstufen automatisch angestoßen werden. Die elektronische Steuerung ist dann für den reibungslosen Ablauf des Produktionsprozesses verantwortlich und es stellen sich darüber hinaus keine spezifischen Anforderungen. Problematischer sieht er hingegen die Steuerung in Ad-hoc- bzw. Broker-Netzwerken. Der Grund ist, dass diese speziellen Formen der Projektnetzwerke sich als Einzelfertigung charakterisieren lassen. Hierbei lässt sich die Produktionssteuerung laut Picot nur bis zu einem gewissen Grad elektronisch unterstützen und erfordert darüber hinaus dezentrale Führungs- und Managementfunktionen (auch als *Telemanagement* bezeichnet), die bislang aber kaum untersucht wurden. Dezentrale Steuerungsmechanismen (oder auch *Management by Wire*) setzen daher oft auf Selbstverwaltung. Aber auch wenn die Teilprozesse der Netzwerkunternehmen (und deren IV) i.B. eines durchgängigen Ablaufs abgestimmt sind und entsprechende Schnittstellen geschaffen wurden wird betont, dass ein Mindestmaß übergeordneter Steuerung und Kontrolle als essenziell angesehen werden muss.

2.3.3.2. Koordination kooperativer Tätigkeiten im VU

Eine in dem hier betrachteten Zusammenhang interessante Untersuchung zur dezentralen Führung geht auf Specht und Kahmann zurück: In ihrer Arbeit untersuchen sie die Regelung kooperativer Tätigkeiten in VU[53] Sie betonen noch mal, dass VU als (nach außen) reale Organisation in Bezug auf ihre interorganisationale Struktur explizit zu planen und zu gestalten sind. Ferner weisen sie darauf hin, dass dabei insbesondere die organisatorische Gestaltung der Koordination ein wesentlicher Faktor ist. Dies liegt daran, dass VU zu Gunsten organisatorischer Flexibilität und Schnelligkeit auf eine Institutionalisierung zentraler Managementfunktionen mit hierarchischer Koordinationsstruktur verzichten. Zur Minimierung der dadurch

[53] Vgl. für das Folgende [SK00].

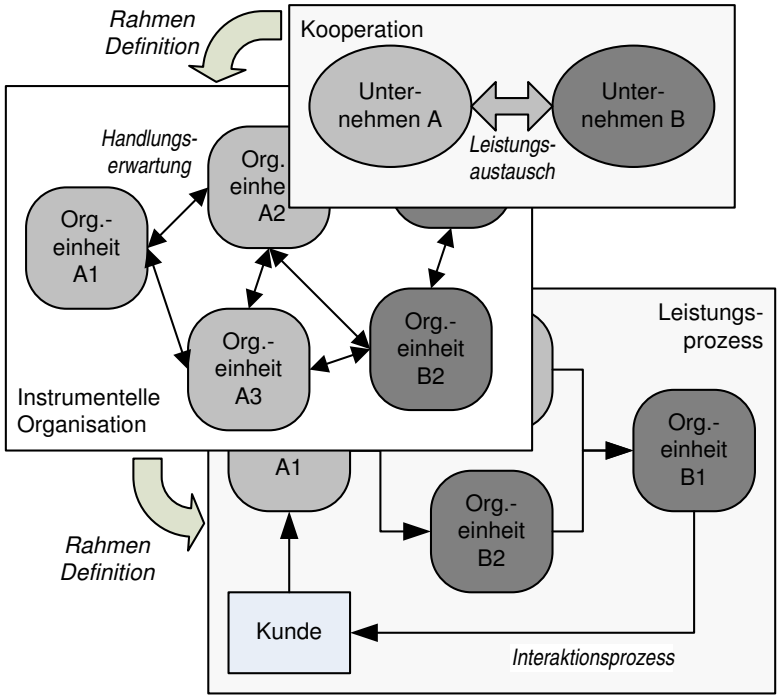

Abbildung 2.23.: Instrumentelle Organisation des virtuellen Unternehmens [SK00]

zusätzlich entstehenden Transaktionskosten ist bei der Planung und Gestaltung von VU neben den eigentlichen Leistungsprozessen insbesondere auch die Regelung der zur Leistungserbringung erforderlichen Interaktionsprozesse zwischen einzelnen organisatorischen Elementen zu bedenken. Im Falle von Produktionsnetzwerken in VDU sind das Interaktionsprozesse zwischen Produktionseinheiten und insbesondere auch den in die Produktion einbezogenen Kunden.

Im Sinne der organisatorischen Gestaltung ist es zweckmäßig, ein VU als Gesamtheit von Leistungsprozessen und einer Menge organisatorischer Regeln zu betrachten. Die organisatorischen Regeln geben dabei einen Rahmen für die Leistungserstellung vor (vgl. Abb. 2.23). Für die generelle Gestaltung der Organisationsstruktur werden nun in der betriebswirtschaftlichen Theorie (z. B. nach Grochla) drei zusammenhängende Freiheitsgrade angenommen:

1. Regelungen der Arbeitsteilung (Spezialisierungsart und -intensität),
2. Regelungen der Koordination (Entscheidungsbefugnisse, Weisungssystem, Koordinationsinstrumente) und
3. Regelungen der Konfiguration (Gliederungstiefe und -breite).

Bei der Anwendung dieser Prinzipien auf VU sind einige Besonderheiten zu beachten. Zunächst kann die Regelung der Arbeitsteilung nicht in beliebiger Weise ausfallen.

Sie muss zum einen die Autonomität der Netzwerkunternehmen berücksichtigen und kann daher nicht beliebig fein bis in Unternehmensinterna hineinreichen. Sie muss zum anderen als Basis der unternehmensübergreifenden Koordination dienen und daher genügend Details enthalten, um deren Planung und Gestaltung zu ermöglichen. Ähnliches gilt für die Regelung der Konfiguration, in deren Rahmen nicht etwa auf einzelne Arbeitsstellen der Netzwerkunternehmen Bezug genommen werden kann, die ja in erster Linie zu deren interner Organisation zu zählen sind.

Noch weitreichende Folgen ergeben sich für die Regelung der Koordination. Hierfür sind generell vier strukturelle Mechanismen möglich:

1. persönliche Weisung,
2. Selbstabstimmung,
3. Programme und
4. Pläne.

Um bei der Regelung der Koordination im VU die Effizienz zu steigern, kommen nun generell zwei Möglichkeiten in Betracht: die Reduktion des Koordinationsbedarfs und/oder die Optimierung der Koordinationsmittel. Ersteres ist zunächst Aufgabe und Ziel der organisatorischen Gestaltung. Letzteres setzt im Anschluss darauf auf und betrifft z. B. eine Verbesserung der Koordinationsprozesse mittels IV, was im Verlauf gerade der vorliegenden Arbeit noch detailliert zu zeigen sein wird. Wie bei herkömmlichen Organisationen kann auch im VU eine Komplexitätsreduktion der koordinativen Gesamtregelung durch Aufgliederung in entkoppelte Organisationseinheiten erreicht werden (vgl. nochmals Abb. 2.23). Diese werden in Bezug auf die klassische Abteilungsbildung auch als *virtuelle Abteilungen* bezeichnet. Anders als bei herkömmlichen Organisationen wird jedoch die Koordination durch Weisung im VU meist als problematisch betrachtet. Stattdessen wird die Koordination durch Selbstabstimmung hervorgehoben. Die für Weisungen notwendigen hierarchischen Beziehungen, die im VU unternehmensübergreifend verlaufen, führen zu hohem Koordinationsbedarf und möglichen Interessenskonflikten der Instanzen. Selbstabstimmung setzt hingegen auf die eigenständige Koordination von Organisationseinheiten. Komplette Autonomie läuft jedoch der Intention des Netzwerks entgegen und so wird meist die Kombination mit übergeordneten Regeln als notwendig erachtet. Pläne und Programme sind weitere Mittel, die den Koordinationsbedarf im VU senken können. Programme geben klare Handlungsanweisungen vor, die keiner weiteren Abstimmung bedürfen. Dies schränkt den Bedarf des unternehmensübergreifenden Informationsaustauschs ein. Der zugrunde liegende Prozess muss zwar vorweg bekannt sein, kann aber konditionale Verzweigungen beinhalten. Ist hingegen kein ausreichendes Wissen vorhanden, lassen sich durch Pläne Vorgehensschemata festlegen. Mittels derer können zu geeignetem Zeitpunkt bedarfsgerecht Programme erstellt werden.

Als Konsequenz schlagen Specht und Kahmann einen Systemansatz vor, der später auch in der vorliegenden Arbeit zur Entwicklung einer IV-Konzeption aufgegriffen wird: statt einer aufgabenorientierten Perspektive (wie bei der Aufgabenanalyse und -synthese oder der Prozessgestaltung) mit resultierender Aufbau- und Ablaufstruktur fokussieren sie eine so genannte *akteursorientierte Sicht* mit dem Resultat einer *Interessens-* und *Interaktionsstruktur*. Dies stellt laut den Autoren *„den interaktionalen Aspekt der arbeitsteiligen Leistungserbringung in den Vordergrund“*. Konkret wird die Modellierung der Interaktionsstruktur durch Programme und Pläne realisiert. Diese bilden exakt die unternehmensübergreifenden Koordinationsprozesse ab. In der Folge bieten sich optimale Voraussetzungen für eine IT-Unterstützung dieser aufwendigen Vorgänge. Daneben wird die Selbstabstimmung durch eine explizite Gestaltung der Interessensstruktur begünstigt. Gegenstand der Betrachtungen sind dabei stets die handlungstragenden Organisationseinheiten (auch als *Akteure* bezeichnet) von Netzwerkunternehmen, Zulieferern und Kunden.

Der akteursorientierte Systemansatz geht mit verschiedenen Konzepten für die drei Regelungsaspekte kooperativer Tätigkeiten einher. Im Sinne der vorliegenden Arbeit ist insbesondere das Koordinationskonzept von Interesse. Hiernach liegt die erste Aufgabe des Netzwerkmanagements in der Festlegung von Szenarien kooperativer Leistungserstellung im Zusammenspiel der Organisationseinheiten. Daraufhin werden der Koordinationsbedarf sowie geeignete Koordinationsmechanismen und -mittel bestimmt. Aus Informationstechnischer Sicht ist die Bestimmung notwendiger Kommunikationskanäle hervorzuheben. Aus organisatorischer Sicht kommen Entscheidungsbefugnisse und Abstimmungsinhalte hinzu. All diese Regelungen erfolgen durch Gestaltung von Programmen und Plänen (vgl. Abb. 2.24). Die Ablaufkoordination der Leistungsprozesse wird in Programmen bestimmt, die zu einer relativ festen Kopplung von Organisationseinheiten führen und nur eine beschränkte Anzahl konditionaler Varianten ermöglichen. Pläne beinhalten dagegen Koordinationsvorschriften indirekter Planungsvorgänge, die zur Gestaltung stark situationsabhängiger Programme dienen. Sie realisieren eine variable Kopplung von Organisationseinheiten. Darüber hinaus kann durch so genannte *Metapläne* sogar noch ein weiterer Indirektionsschritt berücksichtigt werden. Dabei handelt es sich um Koordinationsvorschriften für die Gestaltung von Plänen, woraus lediglich eine lose Kopplung resultiert.

Bei der letztendlichen Realisierung kooperativer Tätigkeiten sind die ausgestalteten Regelungen in konkrete Maßnahmen umzusetzen. Die einzelfallabhängigen Realisierungsmaßnahmen können im Sinne der folgenden Typologie systematisiert werden: Generell können *ideelle* und *materielle* Maßnahmen unterschieden werden, deren Realisierungsprozess entweder eine *Institutionalisierung* oder eine *Implementierung* darstellt. Institutionalisierung betrifft generell die Festschreibung, Sicherung und Stabilisierung organisatorischer Regelungen. Ideelle Institutionalisierung meint die Manifestation von Regelungen, z. B. Verträge und Kooperationsbedingungen. Bei materieller Institutionalisierung werden Produktionsfaktoren eingerichtet: Im VU betrifft dies insbesondere den Aufbau der IT-Infrastruktur. Unter Implementierung

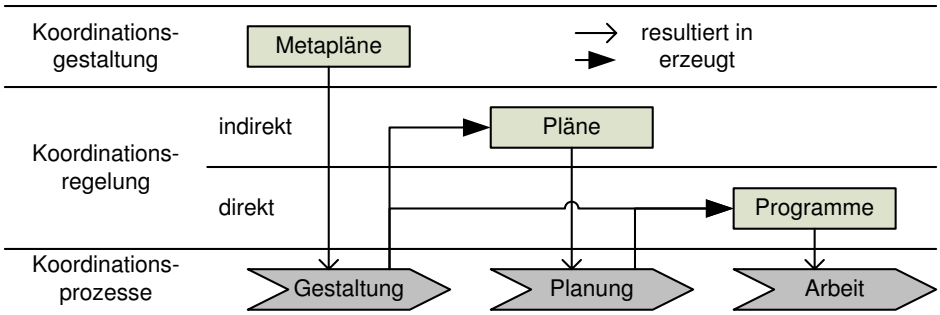

Abbildung 2.24.: Koordinationsebenen in virtuellen Unternehmen [SK00]

wird hingegen die Bekanntgabe, Anweisung und Durchsetzung organisatorischer Regeln verstanden. Im ideellen Fall ist das die Kommunikation von Handlungserwartungen, so z. B. bei Kooperationsprozessen. Schließlich ist unter der materiellen Implementierung die Umsetzung von Erfordernissen bei der Einführung organisatorischer Regelungen zu verstehen; das kann z. B. die Bereitstellung von Ressourcen oder Anpassung von Schnittstellen bedeuten.

Durch seine rationale, in sich geschlossene Form erscheint dieser organisationstheoretische Koordinationsansatz in besonderer Weise tragfähig, eine aufbauende informationstechnische Konzeption zu begründen. Es soll hier jedoch betont werden, dass er aus sozio-ökonomischer Perspektive eher technokratisch anmutet und keinesfalls als ausreichend gelten kann. Deshalb sollen genau solche sozio-ökonomischen Aspekte (z. B. Unterbindung opportunistischer Handlungsweisen, Schaffung einer Vertrauensbasis etc.) in der Folge explizit ausgeschlossen werden.

2.4. Szenario virt. Dienstleistungsproduktion im Logistiksektor

Wie schon in der Einleitung dargestellt wurde, bedient sich die vorliegende Arbeit zur Konkretisierung der ökonomischen Konzepte eines Szenarios aus der (Güter-)Logistik. In der Logistikdomäne spielen Dienstleistungen eine herausragende Rolle.[54] Mittlerweile ist es üblich geworden, verschiedene Arten von Transportleistungen sowie diverse Nebenleistungen zu komplexen Logistik-Systemen zu kombinieren. Daher ist in diesem Bereich auch die Bildung von Unternehmensnetzwerken zu beobachten. Netzwerkunternehmen bringen hier die unterschiedlichsten logistischen Kernkompetenzen ein. Die Organisationsform kommt dabei der Individualität von Dienstleistungsangebot und -nachfrage (in Bezug auf Ort, Zeit, Qualität und Quantität) entgegen. Die Leistungserstellung muss sich nämlich dynamisch und flexibel im Sinne individueller Anforderungen konfigurieren lassen. Daher bieten sich dann besonders auch virtuelle Organisationsformen an.

[54] Vgl. z. B. [Ren92].

Insgesamt stellt sich die Logistikdomäne als äußerst passende Umgebung zur Konkretisierung der bisher diskutierten ökonomischen Konzepte dar. Darüber hinaus bietet die Logistik einen weiteren Vorteil: Ihr wohnt die natürliche Charakteristik als Bindeglied zwischen Unternehmen inne. Aus diesem Grund kann Logistik im weiteren Sinne als Verallgemeinerung kooperativer unternehmensübergreifender Beziehungen gesehen werden [GS00]. In diesem Sinne sind die nachfolgenden Überlegungen, die in der vorliegenden Arbeit auf der Logistik aufbauen, nicht als domänenspezifisch zu verstehen, sondern können und sollen auf andere Bereiche virtueller Dienstleistungsproduktion übertragbar sein.

Das hier dargestellte Szenario dient vor allem zwei Zielen: Zum einen soll die nachfolgende Untersuchung spezifischer IV-basierter Steuerungskonzepte für Interaktionsprozesse der kooperativen Dienstleistungserstellung in VU eingeleitet und durch einen konkreten Anforderungsrahmen begründet werden. Zum anderen soll für die im weiteren Verlauf entwickelten Konzepte und Mechanismen ein Evaluationsrahmen entstehen.

Dazu wird die bis hierher ganzheitliche Darstellung ökonomischer und organisatorischer Konzepte nun wesentlich abgegrenzt. Es wird sowohl eine spezifische Klasse von Dienstleistungen als auch eine spezifische Klasse der Netzwerkorganisation entsprechender DLU fixiert. Für den sich ergebenden spezifischen VDU-Typ können ausreichend konkrete Anforderung zur Steuerung der unternehmensübergreifenden Dienstleistungsprozesse aufgestellt werden. Hierfür wird dann auch ein exemplarischer Geschäftsfall formuliert. Dieser wird später als Anhaltspunkt für die Bewertung der zu entwerfenden IV-Methoden dienen. Er soll den Erfüllungsgrad der Anforderungen erkennbar machen und Anhaltspunkte für die Vorteilhaftigkeit des Verfahrens liefern.

Im Anschluß erfolgt eine kurze Einführung in die Güterlogistik. Dann wird der beispielhaft verwendete „FreightMixer“ Dienst eingeführt und im Sinne der bisherigen Ausführungen eingeordnet. Schließlich wird ein konkreter logistischer Problemfall erörtert.

2.4.1. Dienstleistungsnetzwerke in der Güterlogistik

Unter den logistischen Leistungen versteht man Problemlösungen zusammenhängender Transformations- und Transferprozesse [Ihd91, S. 13]. In diesem Sinne fasst die *Logistik* alle Vorgänge zusammen, die Bewegungs- und Speichervorgänge mit dem Ziel gestalten, die Effizienz eines räumlichen und zeitlichen Ausgleichs zu optimieren sowie einen qualitativen und quantitativen Ausgleich zu schaffen [Pfo71, S. 18 f.]. Logistische Teilziele bestehen in der Überbrückung von Zeit-, Raum-, Quantitäts- und Qualitätsdifferenzen [Sch88, S. 27]. Bestellte Güter sollen zur richtigen Zeit, in der richtigen Menge, zum richtigen Ort, im richtigen Zustand und auf günstigste Weise zum Empfänger transportiert werden. Baumgarten konkretisiert die anfallenden Aufgaben wie folgt:

Definition 10 (Logistik) *... Planung, Steuerung, Durchführung und Kontrolle aller Materialbewegungen [...] mit den dazugehörigen Informationen und den organisatorischen Voraussetzungen vom Ort der Erstellung über den Verbrauch bis zur Entsorgung – mit dem Ziel hoher Effizienz und Wirtschaftlichkeit ...*[55]

Darüber hinaus wird Logistik oft auch noch im wesentlich umfassenderen Sinn verstanden. Der Logistik wird dann die generelle Funktion zur Planung und Steuerung der betrieblichen Abläufe [Las98, S. 14 ff.] (bzw. Geschäftsprozesse) im Sinne von Material-, Daten- und Steuerungsfluss zugeschrieben. Das Logistikmanagement subsumiert bei diesem Begriffsverständnis die klassischen funktionalen Bereiche von Beschaffungs-, Produktions- und Vertriebsmanagement und wird zum *„Operations-“* oder auch *„Value Chain Management“* erhoben [Sch02b, S. 7 f.]. Dies wird auch explizit auf den zwischenbetrieblichen Bereich ausgeweitet. In diesem Sinne definiert Schönsleben ein Logistiknetzwerk als *„Zusammenfassung der Logistiken mehrerer Mithersteller ...“* und folgert, dass ein *„Produktionsnetzwerk [...] als Synonym für ein Logistiknetzwerk betrachtet werden (kann)“* [Sch02b, S. 12]. In der vorliegenden Arbeit soll die Logistik jedoch nicht als Basis eines integralen Managementansatzes für Unternehmensnetzwerke dienen. Sie soll lediglich einen exemplarischen Kontext bilden, in dem ein konkreter Typ von Unternehmensnetzwerken zur Produktion komplexer Logistikdienste im Gütertransport untersucht werden kann. Daher beschränkt sich die nachfolgende Darstellung dieses Abschnitts auf die Güterlogistik.

2.4.1.1. Dienstleistungen in der Logistikkette

Der Gütertransport stellt eine logistische Teilleistung dar. Hier ist die Raumüberbrückung stofflicher Güter mithilfe von Transportmitteln gemeint. Dabei wird eine so genannte *Transportkette* aufgebaut, die laut DIN 30781 eine *„Folge von technisch und organisatorisch verknüpften Vorgängen, bei denen Personen oder Güter von einer Quelle zu einem Ziel bewegt werden“* darstellt. Genauer meint die technische Verknüpfung das Zusammenspiel von Sachmitteln wie Transportmittel oder Lager. Organisatorische Verknüpfung bezieht sich hingegen zum einen auf die Ebene der IT und zum anderen auf die sozio-ökonomische Koordination der involvierten Unternehmen.

Die Gütertransportkette basiert auf der kausalen Beziehung von Gütererzeugung in Unternehmen und Güterverwendung in Haushalten oder anderen Unternehmen, bei der eine räumliche Distanz zu überwinden ist. Im Rahmen der vielfältigen arbeitsteiligen Beziehungen moderner Volkswirtschaften werden ständig verschiedenartige Logistikketten zur Überbrückung von Raumdifferenzen der in Beziehung stehenden Wirtschaftssubjekte aufgebaut. Der freie Warenaustausch in Europa und Just-In-Time Strategien im produzierenden Gewerbe führen dabei zu einer wachsenden Bedeutung umfassender Logistikdienste – insbesondere im Straßenverkehr sowie in

[55] Zitiert in [Ren92, S. 6].

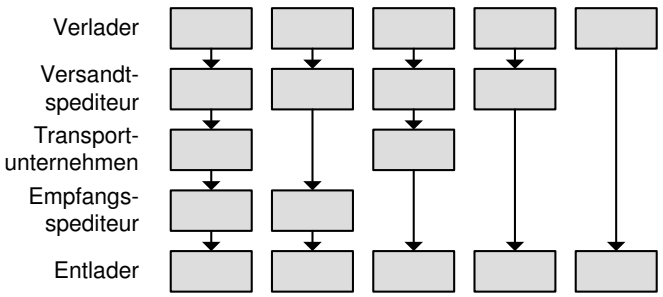

Abbildung 2.25.: Alternative Logistikketten (nach [Mos96, S. 8])

der Kombination verschiedener Transportmittel (so genannte *intermodale Dienste*) [BTGN+98].

Der Güterlogistik werden heute außer den Hauptleistungen noch verschiedene Nebenleistungen im Rahmen oder am Rande der Logistikkette hinzugerechnet. Zu den Hauptleistungen werden meist Transport, Umschlag und Lagerei (TUL) gezählt. Als typische Nebenleistungen werden u. a. z. B. Bestellwesen, Schadensregulierung, Verpackung, Abrechnung und Retouren genannt [Syd92, S. 36 f.] [Mos96, S. 7]. Stabenau führt zudem Beratungsfunktionen (Marktanalyse, Kosten- Leistungsvergleiche) und so genannte „Servicefunktionen“ (Informations-, Produkt- und Finanzdienstleistungen) an [Sta87].

Da sich Unternehmen (vor allem in Unternehmensnetzwerken) heute auf ihre Kernkompetenzen konzentrieren müssen, wird der unternehmensübergreifende Gütertransport oft von spezialisierten (Logistik-)Unternehmen durchgeführt [Ren92, S. 1]. Deren Kernkompetenzen liegen gerade in den verschiedenen logistischen Leistungen. Logistikunternehmen sind dabei als DLU zu klassifizieren: ihre logistischen Leistungen sind als überwiegend immaterielle Transformations- und Transferprozesse gegeben, die auf einem Leistungspotenzial (z. B. Transportkapazität) basieren, sich bei Nachfrage am Kunden ausrichten und diesen als externen Produktionsfaktor interaktiv einbeziehen (z. B. Warenübernahme), wobei am Ende nur ihre Wirkung in Form einer dauerhaften (im Wesentlichen örtlichen) Zustandsänderung des Kunden oder seiner Objekte erhalten bleibt. Konkreter können diese Logistikdienstleistungen wie folgt definiert werden [Ren92, S. 20]:

Definition 11 (Logistikdienstleistung) *Der Begriff der Logistikdienstleistungen beschreibt Tätigkeiten der wirtschaftlichen und effizienten Planung, Steuerung, Durchführung und Kontrolle aller Material-, Waren- und Informationsflüsse entlang der logistischen Kette, die zur Erfüllung der Kundenanforderungen notwendig sind und durch Unternehmen mit dem primären Unternehmenszweck der Dienstleistungserstellung erbracht werden.*

Die zahlreichen Logistikdienstleistungen im Gütertransport werden meist nicht alle von einem einzelnen Logistikunternehmen erbracht. In der Logistikkette arbeiten

vielmehr verschiedene Akteure Hand in Hand. Klassischerweise werden verschiedene funktionale Rollen unterschieden. In Anlehnung an [Mos96, S. 11 ff.] sollen hier Verlader, Spediteur, Transporteur und Entlader unterschieden werden, die in unterschiedlichen Konstellationen zusammenwirken können (vgl. Abb. 2.25). Die Rollen und ihre Beziehungen werden im Folgenden kurz dargestellt:

- *Ver-/Entlader* sind die Pole, zwischen denen die logistischen Ausgleichprozesse stattfinden. Verlader fragen ein Bündel logistischer Leistungen nach, um Güter an ihre Kunden zu liefern. Sie stellen Warensendungen zusammen und erteilen entweder Aufträge an Speditionen (Fälle 1-4) oder übernehmen selbst logistische Leistungen (Fall 5). Sie stehen dabei in einer Informationsbeziehung nicht nur zu Speditionen, sondern auch Entladern. Letztere sind nämlich Nachfrager und Ziel von Warensendungen. Sowohl Verlader als auch Entlader fordern transparente, kontrollierbare Logistikdienstleistungen mit durchgängiger aktueller Information über den Zustand der Logistikkette.

- *Spediteure* treten im Wesentlichen als Vermittler von Transportleistungen zwischen Verladern und Transportunternehmen auf. Dazu planen, organisieren und steuern sie den Warenfluss sowie den damit verbundenen Informationsfluss. Spediteure vermitteln aber auch noch weitere logistische Haupt- und Nebenleistungen oder übernehmen diese selbst. Bei eigenen Transportmitteln spricht man z. B. von Spediteuren im Selbsteintritt; oft sind auch eigene zentrale Lager vorhanden. So genannte Versandspediteure übernehmen den Güterempfang vom Verlader und organisieren deren Transport zum so genannten Empfangsspediteur, der seinerseits die Auslieferung an den Entlader übernimmt. Die logistische Leistungsfähigkeit wird dabei physikalisch (z. B. Wirkungsbereich, Lieferfrequenz, Qualität) und informatorisch (Fähigkeiten in Bezug auf elektr. Tracing/Tracking, Dokumentenaustausch, Warenerfassung etc.) bewertet. Auf Grund ihrer zentralen Stellung innerhalb der Logistikkette unterhalten Speditionen vielfältige Informationsbeziehungen. Zu deren Unterstützung sind in der Regel organisatorische und zum Teil auch (informations-)technische Schnittstellen gegeben.

- *Transportunternehmen* umfassen u. a. Bahnen, Reedereien und Frachtführer. Sie übernehmen Transportleistungen zur Überbrückung von Raumdifferenzen der vom Verlader versandten Güter. Dabei handeln sie im Auftrag von Speditionen, von denen sie Informationen über Waren, Orte und Zeiten erhalten. Die Raumüberbrückung kann durch ein oder mehrere Transportmittel erfolgen.

2.4.1.2. Logistische Informationsflüsse

In der Güterlogistik zeigt sich besonders deutlich die Bedeutung der Information für die Leistungserstellung. Aus betriebswirtschaftlicher Sicht dienen Informationen

(also zweckorientiertes Wissen) der Koordination wertschöpfender Aktivitäten und gehen daher als Einsatzgüter in den Wertschöpfungsprozess ein, d. h. es handelt sich dabei um Produktionsfaktoren [PM93]. Informationen haben dabei erheblichen Einfluss auf die betriebliche Leistungserstellung: Sie sind generell mit Kosten zur Erhebung, Übermittlung, Verarbeitung und Speicherung verbunden und bergen deshalb ein Potenzial zur Rationalisierung. Die Informationsqualität wirkt sich zudem auf die Effektivität und Effizienz betrieblicher Planung und Steuerung aus. Dabei ist besonders zu beachten, dass der Nutzen von Informationen situations- und kontextabhängig ist (z. B. in Bezug auf Zeit). Schließlich stellen Informationen selbst zum Teil wesentliche Leistungen dar, die aus dem Wertschöpfungsprozess als Output hervorgehen. Dies kommt besonders dann zum Tragen, wenn Leistungen selbst immateriell sind oder wiederum in einen Produktionsprozess eingehen. Z. B. determinieren Informationen oftmals die Qualität von Dienstleistungen.

Der Gütertransport ist durch die diversen Haupt- und Nebenleistungen entlang der Logistikkette geprägt. Jede dieser Logistikdienstleistungen ist durch einen eigenen Dienstleistungsprozess gegeben, wobei die einzelnen Prozesse eng verflochtene Kettenglieder darstellen. Informationen haben hier die Aufgabe, die Prozesse zu koordinieren und zu unterstützen [Pfo97]. Dementsprechend lassen sich logistische Informationen zunächst nach ihrem Zweck klassifizieren, einzelne Prozesse zu unterstützen oder die Prozesskette zu koordinieren. Im Gütertransport sind unterstützende Informationen z. B. durch *Handelsinformationen*[56] und *Transportinformationen*[57] gegeben. Handelsinformationen werden in der Regel zwischen Ver- und Entlader ausgetauscht. Transportinformationen sind hingegen für praktisch alle Logistikdienstleistungen entlang der Logistikkette relevant. Darüber hinaus kommt den Transportinformationen aber vor allem auch eine koordinierende Funktion zu. Durch sie werden nämlich die Logistikdienstleistungen aufeinander abgestimmt.

Entlang der Logistikkette kommt es neben dem eigentlichen Warenfluss des Gütertransports also zu verschiedenen damit verbundenen Informationsflüssen. Der Fluss der Handelsinformationen verläuft primär zwischen Verlader und Entlader. Die Transportinformationen hingegen, fließen die gesamte Logistikkette entlang. Durch den Einsatz von IT ist es möglich geworden, den Informationsfluss entlang der Logistikkette vom Warenfluss zu trennen. Transportinformationen fließen damit nicht mehr nur synchron zu den Waren. Darüber hinaus können sie nun den Waren vorauseilen, was eine wesentlich bessere Planung aller logistischen Teildienstleistungen erlaubt. Schließlich sind noch Rückflüsse sowohl von Handels- als auch von Transportinformationen zu beachten. Sie liefern Statusinformationen und erlauben die Nachverfolgung und Kontrolle der Transporte. Abb. 2.26 zeigt einen Überblick aller hier fokussierten Informations- und Warenflüsse der Logistikkette im Zusammenhang.

[56] Z. B. Warenwerte, Qualitätsdaten, Zahlungsinformationen, Herkunftsnachweis, Lieferscheine, Rechnungen.

[57] Z. B. Quell- und Zieldaten, Maße und Gewichte, (Be-)Handlungsdaten, Leitwege und Verkehrsträgerinformationen, Gefahrenhinweise.

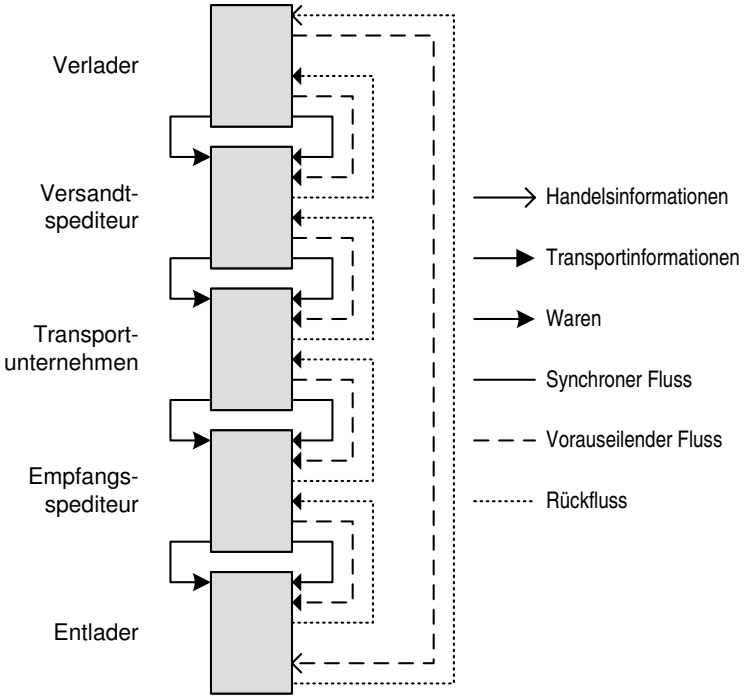

Abbildung 2.26.: Informations- und Warenflüsse in der Logistikkette

Die Beherrschung der Informationsflüsse stellt für Logistikunternehmen eine wesentliche Kompetenz dar. Die administrativen, planerischen und dispositiven Aspekte dieser Aufgabe werden mit dem Begriff der *Informationslogistik* erfasst [Sch02b, S. 42 f.].

Die Informationslogistik sieht sich im Hinblick auf wachsende logistische Leistungsanforderungen mit neuen Herausforderungen konfrontiert [Pfo97]. Durch den freien Warenaustausch auf offenen Märkten und durch zunehmend disaggregierte Organisationsformen produzierender Unternehmen nimmt der Bedarf an logistischen Leistungen zu. Diese Zunahme ist dabei nicht nur quantitativer Natur; vor allem in qualitativer Hinsicht müssen logistische Leistungen immer präziser erbracht werden und nehmen zudem im Funktionsumfang zu. Transparente Logistikdienstleistungen setzen daher eine immer leistungsfähigere und vermehrt unternehmensübergreifende Informationslogistik voraus.

2.4.1.3. Eine virtuelle Logistikkette

Um die zunehmenden Leistungsanforderungen bei hohem Wettbewerbsdruck erfüllen zu können, sind Logistikunternehmen selbst auf adäquate disaggregierte Organisationsformen angewiesen. Diese fordern von der Informationslogistik ganz neue Qualitäten zur Koordination kooperativer Tätigkeiten in bisher nicht gekannter

Dynamik und Flexibilität. Dies wird in diesem Abschnitt anhand *virtueller Transportnetzwerke* nach Zäpfel und Wasner veranschaulicht.[58]

Kontext der Betrachtung sind Speditionssammelgutverkehre, bei denen mittelständische Speditionen viele Klein- und Stückgüter zusammenfassen, um sie als größere Gesamtladung effizienter zu transportieren. In diesem Sektor besteht durch die Deregulierung der EU-Transportmärkte ein hoher Konkurrenz- und Wettbewerbsdruck für europäische Unternehmen. Um hier bestehen zu können, müssen kleine und mittelständische Speditionen ihr Angebot optimieren. Sie gehen dabei zwei Wege: Zum einen erweitern sie ihr Dienstleistungsangebot durch logistische Nebenleistungen, zum anderen optimieren sie die logistischen Hauptleistungen durch Disaggregation ihrer Organisationsstruktur.

Die logistische Hauptleistung misst sich u. a. an Wirkungsbereich und Lieferfrequenz. Diese Größen werden vornehmlich von dem Transportnetzwerk einer Spedition determiniert. Zur Optimierung der Hauptleistung müssen Organisationsstruktur und Transportnetzwerk gemeinsam umstrukturiert werden. Das kann durch so genannte *Hub-and-Spoke-Systeme* geschehen (vgl. Abb. 2.27).

Bei Hub-and-Spoke-Netzwerken bilden die Transportverbindungen im Gegensatz zu Rasternetzwerken keinen voll vermaschten Graphen. Im Mittelpunkt steht ein zentraler Sammelknoten (Hub), der über jeweils eine leistungsstarke Transportverbindung mit verschiedenen Speditionsterminals (Spokes) verbunden ist. Die Speditionsterminals unterhalten zur Abdeckung eines Teilgebiets Beziehungen zu lokalen Frachtführern. Diese sammeln Klein- und Stückgüter in so genannten Nebenläufen bei den Verladern ein und bringen sie zu den zuständigen Terminals. Von den Terminals gehen die konsolidierten Gütertransporte im so genannte Hauptlauf zum Hub, wo sie sortiert werden und dann zu den Terminals zurück laufen. Dort werden sie von den Frachtführern an die Entlader ausgeliefert. Eine exemplarische Lieferung ist in Abb. 2.27 durch gepunktete Kanten dargestellt.

Neben verschiedenen logistischen Vorteilen[59] liefern Hub-and-Spoke-Transportnetzwerke insbesondere die Voraussetzung zur Bildung von Unternehmensnetzwerken. Mehr noch bieten sie dabei sogar Potenzial für virtuelle Unternehmen. In erster Linie besteht für kleine und mittlere Speditionen die Möglichkeit zur Realisierung von Größenvorteilen und Erweiterung ihres Wirkungsbereichs durch Zusammenschluss im Hub-and-Spoke-System. Sie bleiben dabei jedoch selbstständig und konzentrieren sich weiterhin auf ihre Kernkompetenz der lokalen Distribution in den Nebenläufen. Der Hauptlauf wird dementsprechend durch einen eigenständigen Betreiber des Hubs wiederum als Kernkompetenz erbracht. Der Hub kann darüber hinaus unter Umständen die Rolle der fokalen Unternehmung einnehmen.

Bei der Planung des Hub-and-Spoke-Systems ist die Effektivität des Netzwerks in Bezug auf Wirkungsbereich (z. B. national oder EU), Kapazität und Lieferfrequenz

[58] Vgl. dazu [ZW00].
[59] Siehe hierzu [ZW00].

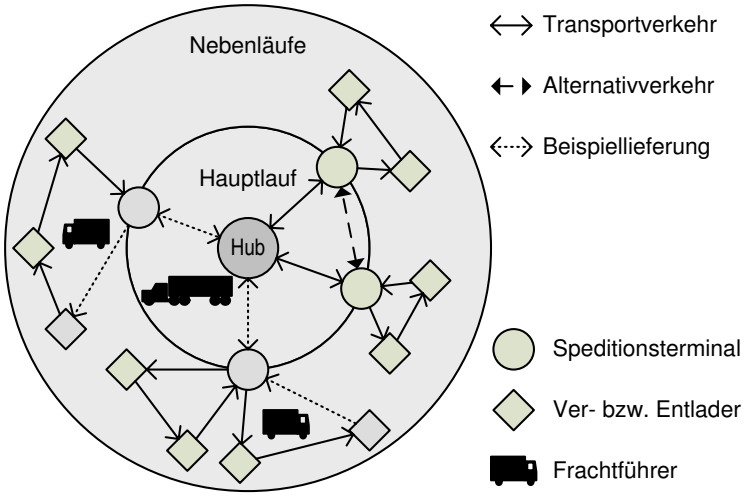

Abbildung 2.27.: Struktur eines Hub-and-Spoke-Systems (nach [ZW00])

(meist 24/48 Stunden-Verkehre im nationalen/EU Bereich) zunächst durch eine strategische Planung (vor allem Dimensionierung des Hub) grundsätzlich sicherzustellen. Die Effizienz kann darüber hinaus maßgeblich auf operationaler Ebene beeinflusst werden. Bei der Planung einzelner Haupt- und Nebenläufe können durch das Aufbrechen der Hub-and-Spoke-Struktur mittels alternativer Verbindungen (z. B. Direktverkehre zwischen den Terminals, in Abb. 2.27 durch eine gestrichelte Kante dargestellt) erhebliche Einsparungen erzielt werden (siehe hierzu wiederum [ZW00]). Dies stellt jedoch hohe Anforderungen an die Flexibilität der Organisation und insbesondere an die Informationslogistik. Eine auftragsinduzierte Umstrukturierung des Hauptlaufes bedingt nämlich zweierlei: zum einen die kurzfristige Disponierbarkeit von Ressourcen (Transportmittel, Lagerkapazität, Personal) und zum anderen die Möglichkeit zur dynamischen Anpassung der Informationsflüsse.

Innerhalb des Hub-and-Spoke-Systems kann eine weitergehende Disaggregation auf Ebene der Spediteure stattfinden. Diese bilden dann durch Auslagerung von Logistikdienstleistungen ein *„Netz im Netzwerk“* [Syd92, S. 36 f.]: z. B. werden oft die eigentlichen Transportaktivitäten an kleinere Frachtführer abgegeben. Auch logistische Nebenleistungen werden oft ausgegliedert. Dies ist besonders dann der Fall, wenn die Dienstleistungen deutlich außerhalb der Kernkompetenz der Spedition liegen, z. B. bei Finanzdienstleistungen oder Transportversicherungen. Als Konsequenz müssen die Speditionen komplexe Kooperationen koordinieren. Besonders dann, wenn diese Mikro-Unternehmensnetzwerke wiederum als virtuelle Unternehmen ausgelegt werden sollen, müssen die beteiligten Logistikdienstleister über entsprechende IV-Mechanismen verfügen, die vor allem auch eine dynamische und flexible Informationslogistik erlauben.

2.4.2. Das FreightMixer-Szenario

Das in der vorliegenden Arbeit verwendete Szenario beschreibt ein VDU der Güterlogistik. Das Konzept basiert auf dem *„FreightMixer“*-Szenario der Hewlett-Packard (HP) Laboratories [HP00]. Dieses Szenario hat den Vorteil, dass es sich sehr gut als Grundlage für die weiteren Überlegungen der vorliegenden Arbeit anpassen lässt. Darüber hinaus wurde seine Relevanz in einer Impact-Studie mit verschiedenen britischen Unternehmen gezeigt [CL01]. Es beschreibt einen fiktiven Markt für Logistikdienstleistungen, der maßgeblich von IV-Mechanismen zur automatisierten Geschäftsabwicklung zwischen Dienstleistern und Kunden geprägt ist. Hierbei wird zunächst die Beziehung von Logistikunternehmen zu Endkunden (das ist im HP-Szenario die industrielle Produktionswirtschaft) beeinflusst. Die Betrachtung fokussiert jedoch die Beziehungen verschiedener Logistikdienstleister bei der gemeinsamen Produktion von Logistikdienstleistungen [PDVM01].

Logistikdienstleister kommen in dem Szenario über eine elektronisch gestützte Form von Marktplätzen, den *E-Markets*, zusammen. E-Markets sind nicht öffentlich; sie sind nur Mitgliedern zugänglich und bilden somit schon ein Unternehmensnetzwerk mit Vertrauensbasis. Das Netzwerkmanagement wird hierbei durch einen sogenannten *Market-Maker* erbracht. Dieser übernimmt auch die Funktionen eines Brokers. Neben der Selektion der Mitglieder steuert der Market-Maker in erster Linie die Allokation über elektronische Kataloge und Auktionen.

Als Logistikdienstleister treten zunächst die bekannten Unternehmensgruppen des Gütertransports im engeren Sinne (Spediteure, Transportunternehmen) sowie Logistikunternehmen im weiteren Sinne (z. B. Transportversicherer) auf. Der Kern dieser logistischen Dienstleistungen bleibt im klassischen Sinn erhalten. Sie werden jedoch zusätzlich auf IV-Ebene abgebildet und liegen dort parallel als *„E-Services“* vor. Dies meint ganz allg., dass für die Dienstleistungen informationstechnische Pendants vorliegen. Aus Sicht traditioneller DLU sind hiermit elektronische Beschreibungen und Schnittstellen verbunden, die eine erhebliche Steigerung der Integrationsfähigkeit bewirken. Dienstleistungsprozesse können dadurch ad hoc initiiert und ihre Interaktionen automatisiert werden.

Die hohe Integrationsfähigkeit einzelner Dienstleistungen im Vertrauenskontext des Marktplatzes dient als Basis für die Konfiguration von VDU. Aus deren Sicht bilden E-Services die eigentlichen Dienstleistungen. Hiermit ist nicht nur die Fähigkeit verbunden, Dienstleistungen des VDU zu beschreiben und zugänglich zu machen, sondern vor allem den Dienstleistungsprozess elektronisch abzubilden. Dadurch ist es möglich, die E-Services beteiligter DLU automatisch zu koordinieren und Dienstleistungen des VDU virtuell zu produzieren. Die Kernorganisation erfolgt dabei im Sinne einer *„Hollow Organisation“* [Brü99, S. 28 f.], die als Vermittler zwischen den sich verändernden Bedürfnissen des Marktes und den begrenzten Kapazitäten der Anbieter agiert, ohne selbst Produktionsmittel zu unterhalten. Es liegen also virtuelle

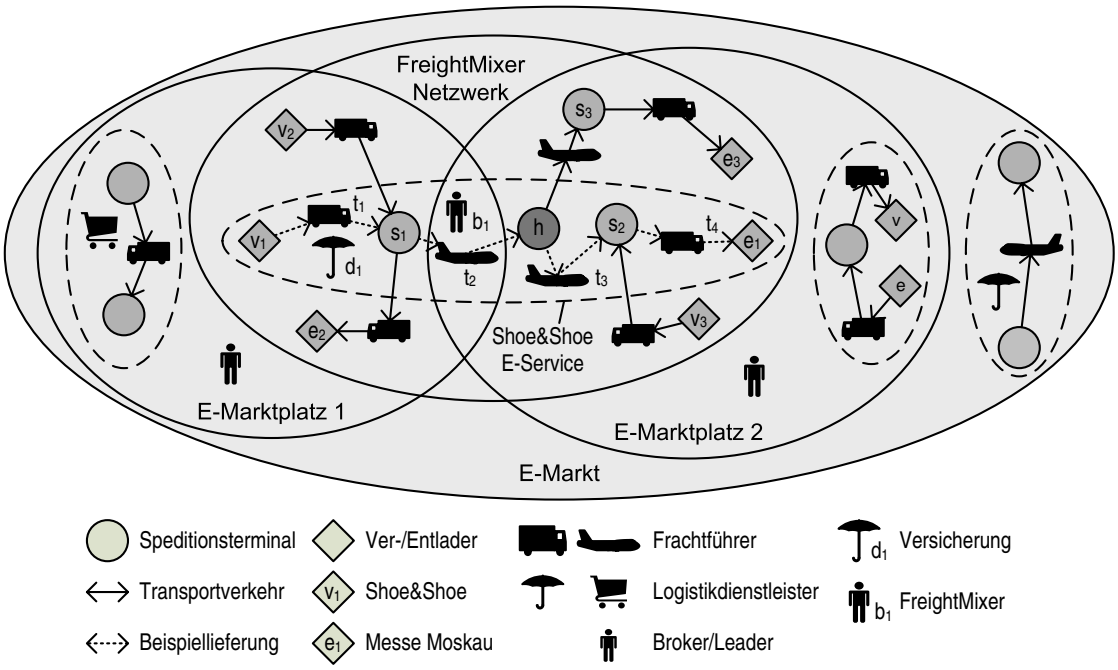

Abbildung 2.28.: Szenario und Geschäftsfall

Unternehmen der höchsten Entwicklungsstufe[60] vor, bei denen eine vollkommen flexible und bedarfsgerechte Koordination von Netzwerkunternehmen durch einen Information Broker erfolgt. Diese Organisationsform wird im Szenario fokussiert und durch ein fiktives Unternehmen „FreightMixer“ veranschaulicht, dass die Rolle des Brokers einnimmt. Im Szenario wird ferner angenommen, dass FreightMixer auf mehreren E-Markets aktiv ist. Damit begründet FreightMixer ein Netzwerk orthogonal zu den Marktplätzen. Abb. 2.28 fasst dies schematisch zusammen.

Die Netzwerkunternehmen, die in dem durch FreightMixer geführten Unternehmensnetzwerk zusammenkommen, produzieren logistische Dienstleistungen. Sie bilden also ein Produktionsnetzwerk, das in Bezug auf die organisatorische Gestaltung virtuellen Charakter aufweist. In Bezug auf die Produktionsplanung ist das HP-Szenario dabei sehr generell. Die strategische Planungsebene wird vollkommen offen gelassen. Die operationale Planungsebene basiert auf der Annahme eines Rasternetzwerks für Gütertransporte. Bei einzelnen Lieferungen erfolgt die Planung mittels multipler Auktionen alternativer individueller Verkehre auf den Marktplätzen [PBB+01]. Abbildung 2.29 zeigt die Planung einer Lieferung, bei der multiple Auktionen für verschiedene alternative Routen durchgeführt werden. Da diese Art der Planung ein offenes Feld aktueller Forschung darstellt und sich die Planung in Rasternetzwerken generell als problematisch darstellt [ZW00], bietet sich dadurch keine ausreichende Argumentati-

[60] Siehe S. 53.

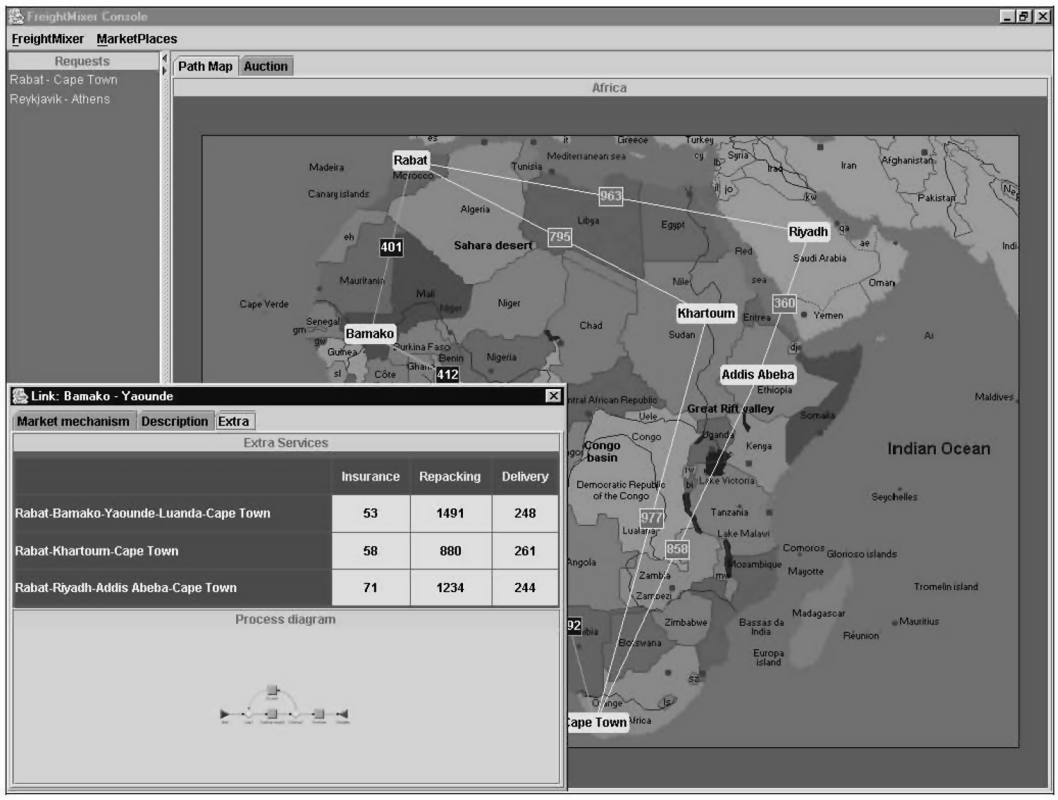

Abbildung 2.29.: Screenshot des FreightMixer Demo-Systems [PW03a]

onsgrundlage für die weiteren Ausführungen der vorliegenden Arbeit. Aus diesem Grund ist eine leichte Abwandlung in Form einer Konkretisierung vorzunehmen.

Bei der Spezialisierung des ursprünglichen HP-Szenarios wird statt dem allgemeinen Rasternetzwerk ein Hub-and-Spoke-Netzwerk angenommen. In diesem Sinne wird unterstellt, dass FreightMixer die strategische Planung eines virtuellen Hub-and-Spoke-Systems vornimmt, also innerhalb des FreightMixer-Netzwerks bestimmte Dienstleister für Funktionen des Hub-and-Spoke-Systems vorgesehen werden (also vor allem Hubs, Spokes und die dazwischenliegenden Transportverbindungen). Dabei sind verschiedene mehr oder weniger dynamische Varianten mit multiplen Transportnetzwerken und variablen Dienstleistern für deren einzelne Funktionen vorstellbar. An dieser Stelle soll aber zunächst vorausgesetzt werden, dass genau ein Hub-and-Spoke-System geplant wird und dessen Funktionen mit längerfristigem Planungshorizont an DLU vergeben werden. Für diesen Fall liegen fundierte Planungsmethoden sowohl für die strategische als auch die operationale Planung vor. Dabei besteht eine klar eingeschränkte Flexibilität, die darin besteht, dass statt eines Hauptlaufs unter gewissen Umständen Direktverkehre zwischen den Terminals eingesetzt werden. Abbildung 2.28 zeigt die Klassifikation von Dienstleistern im Sinne eines Hub-and-

Spoke-Systems durch die Beschriftung: Speditionsterminals mit Spoke-Funktion sind durch ein s und solche mit Hub-Funktion durch ein h gekennzeichnet.

2.4.2.1. Analyse des FreightMixer-Szenarios

Nachdem das generelle Szenario im vorangegangenen Abschnitt eingeführt wurde, soll nun in diesem Abschnitt eine nähere Untersuchung des spezifischen FreightMixer-Netzwerks erfolgen. Dabei liegt der Fokus zum einen auf der Ableitung von Eigenschaften des organisatorischen Unternehmensnetzwerks. Zum anderen werden die Eigenschaften der in diesem Umfeld vollzogenen Dienstleistungsproduktion herausgearbeitet.

FreightMixer als virtuelles Unternehmensnetzwerk FreightMixer begründet ein Unternehmensnetzwerk logistischer DLU: Die Netzwerkunternehmen sind auf einen Teil der arbeitsteiligen Produktion logistischer Dienstleistungen funktional spezialisiert. Sie stehen dabei in Verdrängungswettbewerb zueinander und müssen unter marktähnlichen Bedingungen gegeneinander in Auktionen konkurrieren. Trotzdem teilen sie eine Vertrauensbasis die sich zum einen auf der Mitgliedschaft in mindestenz einem gesicherten[61] Marktplatz gründet. Zum anderen garantiert FreightMixer als Leader des Netzwerks das nicht-opportunistische Verhalten der Teilnehmer. Ganz besonders ist die Informationsintegration der Netzwerkunternehmen hervorzuheben. IV-Mechanismen erlauben die Integration von Planung und Steuerung der Produktion und minimieren dabei die Institutionalisierung von Management-Funktionen. Zusammengefasst sind damit alle konstituierenden Merkmale eines Unternehmensnetzwerks vorhanden. Konkret zeigt dieses Netzwerk die folgenden Merkmale:

- Das Netzwerk ist *dynamisch*. DLU können aus dem Netzwerk herausfallen oder über verschiedene Marktplätze eingegliedert werden. Die Dynamik ist dabei im Hauptlauf des Hub-and-Spoke-Systems geringer. In den Nebenläufen herrscht eine größere Fluktuation (z. B. bzgl. kleineren Frachtführern).

- Das Netzwerk ist *zentriert*. FreightMixer repräsentiert das Netzwerk als fokale Unternehmung nach außen. FreightMixer bietet auch als einziges Netzwerkunternehmen dessen Leistung auf den Marktplätzen an. Die an der Dienstleistungsproduktion beteiligten Unternehmen sind für Kunden vertragsrechtlich transparent.

- Das Netzwerk wird durch *interne hierarchische Führung* gesteuert. FreightMixer übernimmt als Broker und Leader das Netzwerkmanagement, d. h. die Selektion der Mitglieder sowie die Allokation, Regulation und Evaluation von Aufgaben im Zuge eines Auftrags. Darunter fällt auch die Netzwerkebene von Planung und Steuerung der Dienstleistungsproduktion.

[61] Alle Teilnehmer des Marktplatzes werden z. B. in Bezug auf Kreditwürdigkeit evaluiert.

- Das Netzwerk ist ein *Dienstleistungsnetzwerk*. Bei den im Netzwerk angebotenen und produzierten Leistungen handelt es sich ausschließlich um logistische Haupt- und Nebenleistungen. Alle Netzwerkunternehmen sind Logistikdienstleister.

Das FreightMixer-Netzwerk zeigt darüber hinaus die Charakterzüge virtueller Unternehmen: Eine Institutionalisierung zentraler Managementaufgaben im Netzwerk ist durch den Einsatz von IV-Mechanismen minimiert. Das Netzwerkmanagement lässt sich dadurch von einem reinen Information Broker bewerkstelligen. Insofern liegt nach der institutionellen Sicht des Entwicklungsstufenmodells von Arnold (siehe S. 53 f.) ein voll entwickeltes VU vor. Ähnlich deutlich fällt auch die Einordnung in das Venkatraman-Henderson-Modell aus, die sich auf die IT-basierten Organisationsfähigkeiten von FreightMixer gründet. E-Markets bieten hier mit Katalogen und Auktionen die Mittel zur strategischen Selektion optimaler Dienstleister für das Hub-and-Spoke-System. Ferner erlauben sie die Vermarktung der virtuellen Leistungen des Netzwerks durch dynamische und bedarfsgenaue Allokation der Aufgaben aktuell nachgefragter Transportaufträge. E-Services bieten darüber hinaus mit einer technischen Abbildung von Dienstleistungsprozessen und Schnittstellen die Mittel zur Regulation und Evaluation der interaktiven Erstellung von bis zu diesem Punkt lediglich geplanten Leistungen. Insgesamt ermöglichen diese Mittel die Marktinteraktion mittels interaktiver Dienstleistungen, die Kompetenzentwicklung mittels effektiver Kooperation und die Arbeitskonfiguration mittels Neudefinition sozialer Bezüge.

Produktion von FreightMixer-Dienstleistungen FreightMixer bietet als VDU bedarfsgerechte Bündel logistischer Haupt- und Nebenleistungen an. Diese Angebote stellen auf die Kunden abgestimmte virtuelle Leistungsnetze dar. Sie korrespondieren mit virtuellen Leistungsprozessen, die sich innerhalb des FreightMixer-Netzwerks flexibel und dynamisch aus den einzelnen realen Leistungsprozessen der Netzwerkunternehmen konfigurieren lassen. Bei den Netzwerkunternehmen sind Logistikdienstleister und FreightMixer zu unterscheiden. Logistikdienstleister erbringen die einzelnen logistischen Teilleistungen. FreightMixer erbringt als Dienstleister das Netzwerkmanagement. Es können also die drei Formen der *Netzwerkgesamtdienstleistung*, der *Netzwerkteildienstleistung* und *Netzwerkmanagementdienstleistung* unterschieden werden. Deren Eigenschaften sind in Tabelle 2.5 zusammengefasst.

Ein wesentliches Merkmal des Szenarios liegt in der dispositiven Natur der Netzwerkmanagementdienstleistung. Dies macht es möglich, dass FreightMixer objektbezogene sowie maschinen- und personalintensive Logistikdienstleistungen ohne entsprechende eigene Elementarfaktoren produziert. Möglich wird das wiederum durch die standardisierte Form der Leistungserbringung auf Basis von IT. Durch die virtuelle Netzwerkorganisation können aber trotz dieser Standardisierung von Netzwerkteil- und -managementdienstleistungen hoch individuelle Netzwerkgesamtdienstleistungen erbracht werden.

Merkmal	Netzwerkgesamt-dienstleistung	Netzwerkteil-dienstleistung	Netzwerk-Management-Dienstleistung
Einsatzfaktoren	objektbezogen	objektbezogen	dispositiv
Externer Faktor	Kunde	FreightMixer (Kunde)	Dienstleister
Integrationsgrad	direkt abhängig	direkt abhängig	direkt abhängig
Kontaktzwang	gebunden	gebunden	gebunden
Leistungs-verwertung	direkt, konsumtiv	direkt & indirekt, konsumtiv	indirekt, konsumtiv & investiv
Individualität	individuell	standardisiert	standardisiert

Tabelle 2.5.: Merkmale der Dienstleistungen im FreightMixer-Szenario

Alle Dienstleistungen sind von ihren externen Produktionsfaktoren (also denjenigen, für die und mit denen die Leistungen erbracht werden) direkt abhängig und unterliegen einem Kontaktzwang mit diesen. Darüber hinaus ist hervorzuheben, dass Netzwerkteildienstleistungen dadurch, dass sie in der Netzwerkgesamtdienstleistung aufgehen, zum Teil auch vom Kunden abhängig sind und Kontaktzwang unterliegen. Dies spiegelt sich auch in der Leistungsverwertung wider: Netzwerkteildienstleistungen stehen in direkter (z. B. Versandspedition) oder indirekter (z. B. Hub) Beziehung zum Konsumenten. Während Netzwerkgesamt- und -teildienstleistungen eher als konsumtiv gelten können, hat die Netzwerkmanagementdienstleistung auch klar investive Anteile (z. B. strategische Planung des Hub-and-Spoke-Systems).

Die Produktion der Netzwerkgesamtdienstleistung erfolgt primär durch die logistischen Leistungsprozesse der Netzwerkteildienstleistungen. Die Hauptaufgabe von FreightMixer besteht in der Regelung der Koordination dieser kooperativen Tätigkeiten. Das FreightMixer-Netzwerk ist in diesem Sinne als virtuelles Produktionsnetzwerk zu sehen. FreightMixer selbst obliegt die Produktionsplanung und Steuerung auf Netzwerkebene. Die strategische Produktionsplanung betrifft in erster Linie das Hub-and-Spoke-System. Dieses gibt einen Rahmen für das Produktionsprogramm und die Produktionsprozesse vor. Konkret sind die notwendigen logistischen Dienstleistungen (Hub, Spokes, Transportverbindungen, Versicherung etc.) zu identifizieren und entsprechende Dienstleister langfristig in das FreightMixer-Netzwerk einzubinden. Im Zuge dessen werden auch die einzelnen Leistungsprozesse festgelegt, deren Koordinationsbedarf ermittelt sowie entsprechende koordinative Regelungen gestaltet. Dabei spielen insbesondere Programme zur Ablaufkoordination eine große Rolle. In der operativen Produktionsplanung wird dann das tatsächliche „Produktionsprogramm“ direkt in Abhängigkeit von den Kundenaufträgen bestimmt. Im Rahmen des bestehenden Hub-and-Spoke-Systems wird für jeden Auftrag der virtuelle Leistungsprozess angepasst. Dabei können in gewissen Grenzen und geregelt durch Pläne Programmänderungen nötig werden, z. B. bei Direktverkehren oder individuellen Nebenleistungen. Die Konfiguration des Netzes erfolgt danach mittels Auktionen der resultierenden Teilaufgaben unter den Netzwerkunternehmen. Zur

Steuerung der Produktion werden schließlich die virtuellen Leistungsprozesse als zusammengesetzter Dienstleistungsprozess eines E-Service repräsentiert. FreightMixer kann dann die Koordination der individuellen Leistungsprozesse eines spezifischen Auftrags automatisieren. Das schließt ganz allgemein die Freigabe, Kontrolle und Sicherung der Produktionsprozesse ein. Im speziellen Fall von Gütertransporten kommt die Informationslogistik hinzu. Die Koordination einzelner Leistungsprozesse erfolgt durch Selbstabstimmung der Netzwerkunternehmen.

2.4.2.2. Ein exemplarischer Geschäftsfall

Im Folgenden wird die Arbeitsweise von FreightMixer anhand eines exemplarischen Geschäftsfalls veranschaulicht. Dieser konkrete Fall wird später auch zur Evaluation der entwickelten Methoden herangezogen. Hintergrund ist die Abwicklung einer Transportdienstleistung von FreightMixer für den Schuhproduzenten Shoe&Shoe.

Das Beispiel basiert auf einer fiktive Situation aus dem Marketing. Hierbei soll angenommen werden, dass der Schuhproduzent Shoe&Shoe aus Bristol während einer Messe in Moskau einige Prototypen der neuen Kollektion für ein Vertriebsgespräch benötigt. Diese Schuhe befindet sich jedoch noch in Bristol. Damit das Gespräch zustande kommen kann, müssen sie vor Ende der Messe nach Moskau transportiert werden. Shoe&Shoe sucht daher eine möglichst schnelle, sichere und günstige Transportmöglichkeit.

Die Lösung des Transportproblems stellt hohe Anforderungen an die logistische Leistung. Für die Hauptleistung wird eine grenzüberschreitende intermodale Transportkette benötigt, für die bei großer Entfernung nur ein knappes Zeitfenster besteht. Als Nebenleistung fordert Shoe&Shoe eine spezielle Versicherung für die außergewöhnlich wertvolle Fracht. Zudem spielt auch in diesem Fall der Preis eine Rolle.

Shoe&Shoe spezifiziert die Anforderungen als Anfrage auf dem elektronischen Marktplatz E-Move, zu dem auch FreightMixer gehört. Die benötigte Dienstleistung kann hier auf Grund ihrer Individualität nur schwer von allgemeinen Großanbietern bereitgestellt werden. FreightMixer betreibt hingegen nicht nur ein leistungsfähiges Hub-and-Spoke-System, mit dem die Hauptleistung günstig abgedeckt werden kann. Im FreightMixer-Netzwerk befinden sich zudem auch Versicherungen, die passende Policen für Spezialtransporte anbieten. FreightMixer plant nun ein individuelles Leistungsbündel, das die Transportleistung des Hub-and-Spoke-Systems mit der Versicherung kombiniert. Für dieses Leistungsbündel ermittelt FreightMixer dann mittels der Auktionsmechanismen von E-Move die günstigsten Anbieter im eigenen Netzwerk und macht Shoe&Shoe ein Angebot. Da der Preis stimmt, nimmt Shoe&Shoe an. Abbildung 2.28 zeigt die individuelle Konfiguration des FreightMixer-Netzwerks für den Shoe&Shoe Dienst: sie umfasst die logistische Kette (gepunktete Kanten) zwischen Shoe&Shoe (v_1) und der Messe Moskau (e_1), auf der passende Transportunternehmen ($t_1 - t_4$), Speditionsterminals (s_1, s_2) und die Versicherung (d_1) angeordnet sind.

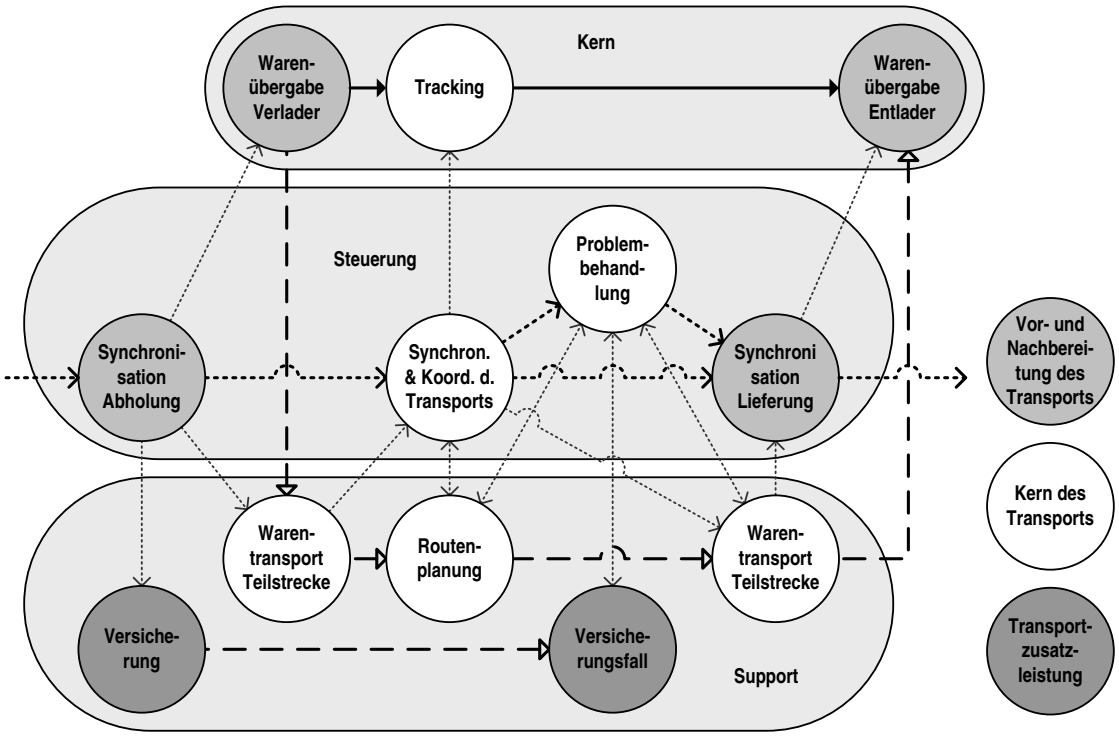

Abbildung 2.30.: Grobstrukturierung der FreightMixer Kernleistungsprozesse

Das spezifische Angebot basiert auf einem virtuellen Leistungsprozess, der bislang nicht real existiert. Dieses Leistungsbündel kann überhaupt nur deshalb angeboten werden, da alle Teilleistungen in Form von E-Services vorliegen. Nach Bestätigung des Angebots durch Shoe&Shoe realisiert FreightMixer den virtuellen Leistungsprozess ad hoc durch Komposition als zusammengesetzten E-Service, der die Koordination der einzelnen Teile regelt. Dazu kommen sowohl Programme als auch Pläne zum Einsatz. Die Koordination des Hub-and-Spoke-Systems erfolgt durch ein Programm, das den standardisierten Interaktionsprozess zwischen Verlader, Speditionsterminals, Transportunternehmen und Entlader regelt. Bevor FreightMixer jedoch das Programm starten kann, muss es so angepasst werden, dass die zusätzliche Transportversicherung einbezogen wird. FreightMixer geht dazu nach einem Plan vor, der eine sichere und konsistente Anpassung garantiert.

Das individuelle Programm komplettiert die Prozessarchitektur der Dienstleistung. Im Sinne der in Sektion 2.3.1.2 vorgestellten Prozessstrukturierung bildet es den Steuerprozess. Die Leistungsprozesse der beteiligten Netzwerkunternehmen bilden die Support-Prozesse. Der Kernleistungsprozess des exemplarischen Dienstes ist in Abb 2.30 als Vorwärtsprozess in Grobstruktur gezeigt.

Wenn das individuelle Programm vorliegt, nimmt FreightMixer dessen Konfiguration vor. Dabei werden die im Programm enthaltenen Rollen durch die aktuellen

Akteure besetzt. Das sind in diesem Fall Shoe&Shoe, die logistischen Dienstleister des Hub-and-Spoke-Systems sowie die Versicherung. Mit dem konfigurierten Programm übernimmt FreightMixer jetzt die übergeordnete Produktionssteuerung. Dies beinhaltet die koordinierte Freigabe einzelner Leistungsprozesse, die Kontrolle des Ablaufs sowie die Sicherung des Ergebnisses. Auf diese Weise synchronisiert FreightMixer z. B. die Verladung der Schuhe mit dem Transport zum Speditionsterminal und regelt die Übergabe zum nachfolgenden Transport im Hauptlauf des Hub-and-Spoke-Systems. Dabei kontrolliert FreightMixer den ordnungsgemäßen Ablauf der Teiltransporte und leitet ggf. kompensatorische Maßnahmen ein, falls ein Problem auftritt. Das Programm erlaubt ferner einen gewissen Grad an Flexibilität. So kann z. B. je nach aktueller Auftragslage der Speditionsterminals ggf. eine Direktverbindung gewählt werden.

Während FreightMixer die Produktionssteuerung auf Netzwerkebene übernimmt, laufen die einzelnen Leistungsprozesse in Selbstabstimmung der jeweiligen Akteure ab. So regelt z. B. Shoe&Shoe die Verpackung und Verladung seiner Ware eigenständig und auch der Frachtführer koordiniert seine LKW-Flotte autonom. Die aus der Abhängigkeit und dem Kontaktzwang der einzelnen Teildienstleistungen resultierenden Interaktionsbeziehungen zwischen den Leistungsprozessen stellen dabei kein Problem dar. Alle Teilnehmer bieten ihre Dienstleistungen nämlich als E-Services an. Und diese können durch beliebig komplexe Interaktionsprozesse verknüpft werden. Dadurch erhält Shoe&Shoe z. B. automatisch die Uhrzeit der Abholung vom Frachtführer und letzterer umgekehrt eine Bestätigung von Shoe&Shoe. Die mit den einzelnen Dienstleistungen einhergehenden Interaktionsprozesse wurden von FreightMixer im Voraus als elektronischer Dienstleistungsprozess des zusammengesetzten E-Service gestaltet. Sie können hier nun ad hoc zur Anwendung kommen.

Die gemeinsame IV-Infrastruktur der Netzwerkunternehmen, auf deren Basis für alle Dienstleistungen elektronische Schnittstellen vorliegen, erlaubt es FreightMixer, die Produktionssteuerung auf der Netzwerkebene weitgehend zu automatisieren. Auf diese Weise kann FreightMixer Shoe&Shoe nicht nur effektiv, sondern auch äußerst effizient bedienen. Neben dieser werden nämlich parallel noch viele weitere Dienstleistungen produziert. Gleichzeitig kommt FreightMixer mit einem Minimum an Produktionsmitteln aus. Die günstige Kostenstruktur der Produktion ist die Basis für FreightMixers Wettbewerbsfähigkeit.

2.5. Zusammenfassung

In den vorangegangenen Sektionen dieses Kapitels wurden die ökonomischen Grundlagen der vorliegenden Arbeit gelegt. Die Ausführungen setzten am fundamentalen Dienstleistungsbegriff an. Sie stellten sowohl dessen immateriellen Prozesscharakter sowie dessen inhärente Interaktivität heraus. Es wurde dann deutlich gemacht, wie sich diese konstituierenden Eigenschaften auf die Produktion von Dienstleistungen

auswirken. Ferner wurde herausgestellt, welche Faktoren dabei die Produktivität determinieren. Es zeigte sich, dass der Schlüssel zur effizienten Dienstleistungsproduktion in der Optimierung von Interaktionsprozessen liegt. Dies kann besonders gut durch IV-Mechanismen erreicht werden.

Um einen Ansatzpunkt für diese IV-Mechanismen zu finden, wurde die Organisation der Dienstleistungsproduktion untersucht. Dabei wurde zunächst der Wandel der instrumentalen und funktionalen Organisation von der Funktions- zur Prozessorientierung nachvollzogen. Diese Entwicklung war für die Produktion von Dienstleistungen entscheidend: bestehen diese doch im Wesentlichen gerade nur aus einem Leistungsprozess. Daher wurden auch der Prozessbegriff sowie die Prozessgestaltung grundlegend eingeführt und auf Dienstleistungsprozesse ausgeweitet.

Die prozessorientierte Organisation unterstützt die Dienstleistungsproduktion aus organisationstheoretischer Sicht jedoch noch nicht optimal: Sie ist zu unflexibel. Daher wurde die Betrachtung auf organisatorische Netzwerke ausgeweitet. Das Netzwerkprinzip nimmt den Gedanken der Prozessorientierung auf und stellt den Prozess in den Mittelpunkt. Es bricht aber die monolithische Struktur der institutionellen Organisation auf und macht sie dadurch flexibel. Das sind optimale Voraussetzungen für Dienstleistungen, die sich im Sinne ihrer Interaktivität permanent am Kunden ausrichten müssen.

In Fortsetzung dieses Gedankens wurden dann virt. Unternehmen vorgestellt. Dabei wurde untersucht, wie (institutionale) Netzwerkorganisationen (instrumental und funktional) zu organisieren sind, um ihre Flexibilität zu maximieren. Es stellte sich heraus, dass dazu gerade IV-Mechanismen maßgeblich sind. Diese kommen darüber hinaus noch dem Interaktionsbedarf von Dienstleistungsprozessen zugute. Mit dem VDU wurde die optimale Organisationsform gefunden, die wesentliche Grundlagen zum späteren strukturellen Entwurf eines informationstechnischen Dienstleistungsmodells liefert.

Auf Basis der gefundenen organisatorischen Grundstruktur kehrte die Betrachtung dann zu den Methoden des Produktionsmanagements zurück. Diese wurden im Rahmen so genannter Produktionsnetzwerke untersucht. Insbesondere wurde ein generelles Koordinationsprinzip zur Umsetzung von Managementaufgaben in virtuellen Unternehmen vorgestellt. Zusammengenommen war dadurch ein konzeptioneller Rahmen zur methodischen Unterstützung der Dienstleistungsproduktion gegeben.

Die Betrachtungen fokussierten dann die Konkretisierung der theoretischen Konzepte in einem Szenario. Um dieses zu untermauern, wurden zunächst die notwendigen Grundlagen der Güterlogistik knapp dargestellt. Dabei wurde besonders auf die organisatorischen Aspekte von Prozessen und Netzwerken eingegangen. Es zeigte sich dabei, dass dieser Bereich auf ganz natürliche Weise von den vernetzten Prozessen logistischer Ketten bestimmt wird. Daneben wurde auch ein fundiertes Konzept zur virtuellen Organisation eines Transportnetzes vorgestellt.

Schließlich wurde das Szenario eines konkreten logistischen DLU vorgestellt. Anhand dessen sollen im Verlauf der nachfolgenden Arbeit konkrete Voraussetzungen für eine IV-Infrastruktur abgeleitet werden. Das durch eine Impact-Studie abgesicherte Beispiel der Hewlett-Packard Labs wurde dazu basierend auf dem zuvor dargestellten Konzept virtueller Transportnetze abgewandelt. Dies erlaubte eine detaillierte Untersuchung der ökonomischen Eigenschaften der Organisation. Zudem wurde das Szenario durch einen konkreten Geschäftsvorfall illustriert und dadurch ein Testfall für die spätere Evaluation der Arbeit geschaffen.

3. Service Oriented Computing (SOC) zur Prozessintegration im VDU

3.1. Zielsetzung

Kapitel 3 bildet den zweiten Grundlagenteil der vorliegenden Arbeit. Dessen Ziel besteht darin, fundamentale Techniken einzuführen, auf deren Basis eine Unterstützung der im vorherigen Kapitel dargestellten Produktion von Dienstleistungen in virtuellen Unternehmensnetzwerken – den so genannten virt. Dienstleistungsunternehmen (VDU) – mittels Informationsverarbeitung realisiert werden kann. Dies soll mit dem Hintergrund der angestrebten Virtualisierung von Dienstleistungen durch E-Services erfolgen, d. h. dass IT-basierte Unterstützungsmöglichkeiten zur Koordination der (Geschäftsprozesse zur) Produktionssteuerung im Mittelpunkt stehen.

Das Kapitel soll zunächst in Sektion 3.2 ein generelles Bild der Informationsverarbeitung im VDU vermitteln. Hierbei liegt das Ziel zunächst in der Festlegung einer eindeutigen Terminologie betrieblicher IV als Grundlage der weiteren Betrachtungen. Des Weiteren soll das Problemfeld der betrieblichen IV-Integration in geeigneter Weise strukturiert werden, so dass ein Verständnis für die Situation in heterogenen IV-Landschaften entsteht. Aufbauend auf diesen Erkenntnissen sollen letztendlich die Anforderungen von virt. Dienstleistungsunternehmen analysiert und daraus spezifische Bereiche integrativer Infrastruktur- und Anwendungsfunktionen abgeleitet werden.

In Bezug auf die Fragestellung der vorliegenden Arbeit sollen dann in Sektion 3.3 problemspezifische Funktionsbereiche der Informationsverarbeitung im VDU vertieft werden. Das Ziel besteht darin, drei Technikbereiche aus dem Kontext der integrativen Infrastruktur vorzustellen, die wesentlichen Anteil an dem E-Service-Rahmenwerk dieser Arbeit haben werden. Dies schließt sowohl ein konzeptionelles Rahmenwerk für ganzheitliche IV-Lösungen im betrieblichen Kontext als auch spezifische Techniken zur Interoperation und prozessorientierten Koordination kooperierender Anwendungssysteme ein.

Schließlich soll Sektion 3.4 die konkrete Technikfamilie des Service Oriented Computing (SOC) einführen. Das Ziel liegt zum einen darin, die anvisierten Funktions- bzw-Technikbereiche einer VDU-Infrastruktur im Hinblick auf das Service-Paradigma zu konkretisieren. Zum anderen sollen mit den konkreten Standards und Spezifikationen des SOC die Mittel für eine letztendlich angestrebte Implementierung aufgezeigt werden.

3.2. Einsatz integrierter Informationsverarbeitung (IV) im VDU

VU basieren grundsätzlich auf einer Steigerung der organisatorischen Flexibilität. Dies wird im Wesentlichen durch einen Verzicht auf die Institutionalisierung zentraler Managementaufgaben erreicht. Das wiederum kann nur durch den massiven Einsatz von IT und innovativen IV-Mechanismen gelingen. Grundsätzlich stellt sich die Frage, welche strategischen Ziele mit dem Einsatz von IV verfolgt werden und welche konkreten Anforderungs- und Aufgabenbereiche sich aus den Eigenschaften von VDU an die IV ableiten lassen. Des Weiteren ist zu klären, welche IV-Systeme dann im VDU zum Einsatz kommen sollen und wie sie sowohl untereinander als auch in die Strukturen und Abläufe des Unternehmens integriert werden können. Durch Beantwortung dieser Fragen sollen die folgenden Sektionen die Diskussion virt. Dienstleistungsunternehmen aus informationstechnischer Perspektive fortsetzen. Die Zielsetzung der Betrachtungen besteht hier vor allem darin, die spätere Abgrenzung und Einbettung der angestrebten E-Service-Technologie zu ermöglichen. Aus diesem Grund soll zum einen eine knappe Einführung betrieblicher IV gegeben werden und zum anderen eine Analyse ihres Einsatzes im VDU erfolgen.

In diesem Sinne legt Abschnitt 3.2.1 zunächst einige grundlegende Konzepte und Begriffe integrierter Informationsverarbeitung im Unternehmen fest. Dabei werden die verschiedenen Systembegriffe der betrieblichen IV voneinander abgegrenzt. Zugleich wird die Notwendigkeit einer (System-)Integration hervorgehoben und in Bezug auf ihre Dimensionen dargestellt. Auf dieser Basis analysiert Abschnitt 3.2.2 Ziele, Anforderungen und Aufgaben für IV im VDU und diskutiert zwei ihrer wesentlichen Aspekte: generische Mechanismen einer integrativen Infrastruktur sowie fachliche Funktionen zur Unterstützung der Kooperation im VDU.

3.2.1. Integrierte Informationsverarbeitung im Unternehmen

Der folgende Abschnitt führt einige fundamentale Begriffe und Konzepte der betrieblichen Informationsverarbeitung ein, die zum Verständnis der restlichen Arbeit benötigt werden. Zunächst erfolgt dazu eine Abgrenzung verschiedener Systembegriffe, die zum einen die unternehmerische und zum anderen die technische Sicht betrieblicher IV-Mechanismen betonen. Danach wird die besondere Bedeutung einer *integrierten* IV hervorgehoben und in diesem Sinne der Integrationsvorgang von (technischen) IT-Systemen mit seinen verschiedenen Ebenen dargestellt.

3.2.1.1. Systemsichten betrieblicher Informationsverarbeitung

Aus der Perspektive der Wirtschaftsinformatik wird unter Informationsverarbeitung (IV) im engeren Sinne die Verarbeitung von (betrieblichen) Informationen durch Algorithmen, Verfahren oder Methoden zu neuen Informationen verstanden.[1] In

[1] Vgl. z. B. [Sch00, S. 45 ff.].

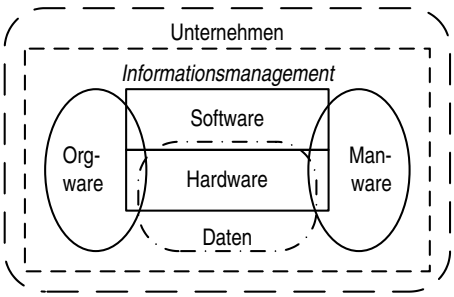

Abbildung 3.1.: Komponenten eines IV-Systems [Sch00, S. 49]

der Regel wird der IV-Begriff jedoch im weiteren Sinne verwendet: als elektronisch gestützte Verarbeitung von Informationen mitsamt Eingabe, Bearbeitung über mathematische, logische und andere Operationen, Ausgabe, Übertragung und Speicherung. Zur Erfassung einer solchen Sicht im Unternehmen sind verschiedene Perspektiven erforderlich, die durch verschiedene Systembegriffe repräsentiert werden. Die wichtigsten davon sind Informations- und Anwendungssysteme.

Informationssysteme Die betriebliche Informationsverarbeitung erfordert zunächst eine Betrachtung ganzheitlicher Mensch/Aufgabe/Technik-Systeme [Hei97, S. 861 f.], die als *Informationssysteme (IS)* bezeichnet werden (vgl. Abb. 3.1). IS leisten die Beschaffung, Verarbeitung, Übertragung, Speicherung und Bereitstellung der Informationen. Der Schwerpunkt liegt dabei meist entweder auf der Verarbeitung (Eingabe-Verarbeitung-Ausgabe (EVA)-Prinzip) oder der Kommunikation von Informationen. IS umfassen diesbezüglich vielfältige Aspekte: grundlegende Bestandteile sind Hardware, Software und Daten; die beiden letztgenannten werden bei fachlichem Bezug auch als *Anwendungssystem* oder kurz *Anwendung* bezeichnet. Daneben fließt die Organisation der Einsatzumgebung sowie des IS selbst ein *(Orgware)*. Schließlich sind auch die menschlichen Akteure einzubeziehen *(Manware)*. Im Unternehmen können mehrere IS existieren, die in ihrer Gesamtheit die Informationsinfrastruktur [Hei97, S. 862] bilden.

All diese Aspekte müssen durch ein *Informationsmanagement* berücksichtigt werden [MBK+01, S. 197 ff.]. Dieses umfasst organisatorische und dispositive Aufgaben, die sich in Planung, Durchsetzung und Kontrolle der IV gliedern lassen. Zu den wichtigsten Aufgaben des Informationsmanagements gehört die strategische Planung. Hierbei ist durch eine *IV-Strategie* festzulegen, wie die Unternehmensstrategie durch IV unterstützt werden soll und welche Potenziale eine IV-Strategie dem Unternehmen eröffnet.

Die IV-Strategie ist dann in einer *Architektur der Informationsinfrastruktur (kurz IV-Architektur)*[2] zu konkretisieren. Eine weitere Unterteilung kann u. a. in Bezug

[2]Mit dem Architekturbegriff wird in der Regel – unter Anspielung auf die Architekturlehre

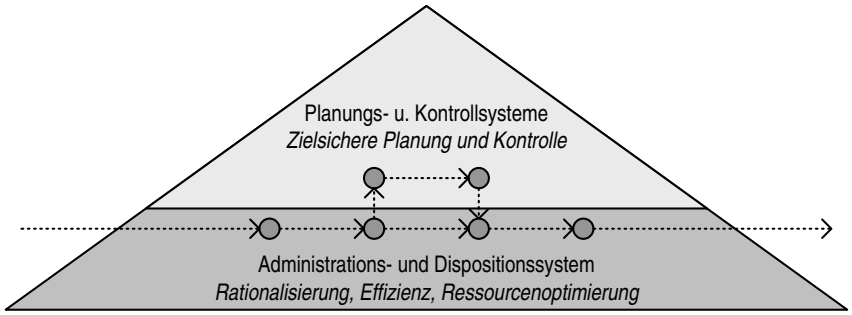

Abbildung 3.2.: Integrierte AS der Organisationspyramide (nach [MBK+01, S. 4])

auf *Informations-* und *Technologiearchitektur* erfolgen. Die Informationsarchitektur spezifiziert die betriebliche Informationsnachfrage in Bezug auf fachliche Daten, Funktionen und Prozesse. Dies erfolgt in der Regel durch deren Modellierung in entsprechenden *Daten-* und *Anwendungsarchitekturen*. Die Technologiearchitektur (kurz IT-Architektur) spezifiziert schließlich das mit der Nachfrage abzugleichende Informationsangebot in Form von *Mikro-* und *Systemarchitektur*. Die Mikroarchitektur beschreibt Anwendungssoftware und Datenstrukturen. Die Systemarchitektur erfasst die technische Infrastruktur mitsamt Hardware und Systemsoftware.

Anwendungssysteme Software und Daten für ein konkretes betriebliches Anwendungsgebiet werden als *Anwendungssystem (AS)* im engeren Sinne bezeichnet [SH01]. Anwendungssysteme sind also die „automatisierten“ Komponenten eines IS, die mit Bezug auf die Anwendungsarchitektur spezifisch für die Zwecke eines Unternehmens(typs) und eines Anwendungsgebiets geschaffen wurden [Mer97, S. 46]. Teil bzw. Kern des AS ist die *Anwendungssoftware* (im Folgenden kurz als *Anwendung* bezeichnet). Sie grenzt sich durch ihren konkreten Anwendungsbezug von der *System-* und *systemnahen Software* ab.

Im betriebswirtschaftlich-organisatorischen Anwendungskontext kann man nach dem Verwendungszweck verschiedene Kategorien von AS unterscheiden (vgl. auch Abb. 3.2):

- *Operative Systeme* – Die erste Kategorie beinhaltet zunächst *Administrationssysteme*, die der Rationalisierung vorhandener und strukturierter Abläufe

im Bauwesen – ein Bauplan im Sinne eines Modells aller relevanten Komponenten und Beziehungen eines Systems aus verschiedenen Blickwinkeln mitsamt den Konstruktionsregeln zur Erstellung des Bauplans verstanden [Sin97, S. 875]. In Bezug auf den Architekturbegriff im IV-Kontext existiert jedoch keine einheitliche Terminologie [Wal98]. So wird z. B. die hier gemeinte Architektur aller Informationssysteme u. a. auch (zum Teil widersprüchlich) als Informations- oder IS-Architektur bezeichnet. Die verschiedenen Architekturbegriffe werden daher jeweils vor ihrer Verwendung kurz eingeführt und in diesem Sinne konsistent weiterbenutzt.

dienen. Ferner gehören dazu *Dispositionssysteme*, die Entscheidungsfindungsprozesse automatisieren. Gemeinsam bilden sie die Kategorie der operativen Systeme. Entsprechende AS finden zumeist auf den unteren Ebenen der betrieblichen Hierarchie Anwendung.

- *PuK-Systeme* – Die zweite Kategorie umfasst zunächst *Planungssysteme*, die zur manuellen Unterstützung unstrukturierter Vorgänge (z. B. durch Informationsbereitstellung) dienen. Sie sind oft eng mit *Kontrollsystemen* verbunden. Diese weisen auf spezifische Situationen hin und leiten Handlungsempfehlungen ab. Zusammen werden sie zur Kategorie der Planungs- und Kontrollsysteme (PuK) zusammengefasst. Solche AS kommen im Wesentlichen auf den höheren Managementebenen zum Einsatz.

3.2.1.2. Integration betrieblicher IT-Systeme

In einem Unternehmen werden heute in der Regel vielfältige und verschiedenartige operative und PuK-Systeme verwendet. Diese stehen dabei nicht isoliert nebeneinander. Sie sollen vielmehr die ganzheitlichen Abläufe und Zusammenhänge eines Unternehmens widerspiegeln und dabei den künstlichen Grenzen der arbeitsteiligen Aufbauorganisation entgegenwirken. Dies ist das Ziel der *integrierten Informationsverarbeitung*.[3] Hierbei stehen die AS durch Informations- und Kontrollflüsse in Verbindung (vgl. noch mal Abb. 3.2). Der Informationsfluss sichert die konsistente Verbreitung elektronischer Daten ohne Medienbrüche und verhindert mehrfache Eingaben. Der Kontrollfluss bildet unternehmerische Geschäftsprozesse ab und sichert deren Umsetzung im laufenden Betrieb.

Aus dem gleichen Grund zeigt sich der Entwurf betrieblicher IT-Architekturen meist als Anpassungsprozess vorhandener Strukturen an neue Anforderungen. Dabei sind die meisten Anwendungssysteme, Infrastrukturmechanismen und Systemressourcen bereits vorhanden. Der Entwurfsvorgang erfolgt somit größtenteils *Bottom-Up*[4]. Dabei werden die vorhandenen sowie ggf. neu hinzukommenden Systeme und Ressourcen im Sinne der veränderten Anforderungen integriert [CMA03, S. 617 f.].

Der Vorgang der betrieblichen Systemintegration wird sich für die integrierte Informationsverarbeitung im virt. Dienstleistungsunternehmen noch als wesentliche Herausforderung erweisen. Gleichzeitig kann die in der vorliegenden Arbeit angestrebte Prozesssteuerung der Dienstleistungsproduktion als funktionale Verfeinerung aufgefasst werden. Im Folgenden wird der Integrationsvorgang daher noch etwas genauer eingeführt und in seinen verschiedenen Dimensionen beleuchtet.

[3] Vgl. z. B. [MBK+01, Sch98, Mer00].

[4] *Bottom-Up* bezeichnet eine Strategie der Systementwicklung, die von vorhandenen Elementen ausgeht und diese im Sinne gegebener Anforderungen (wieder-)verwendet, im Gegensatz zur *Top-Down* Strategie, bei der aus den Anforderungen spezifische Elemente abgeleitet werden (vgl. z. B. [Mer97, S. 441]).

Mit dem Integrationsbegriff ist meist a) ein Prozess, durch den Individuen niederer Ordnung zusammenkommen, um Individuen höherer Ordnung zu bilden oder b) die Komplettierung des Integrationssubjekts zu einem Ganzen gemeint [CMA03, S. 618 f.]. Entsprechend kann Integration im Kontext betrieblicher IV ganz allgemein als Prozess angesehen werden, der z. B. Hardware, Rechenleistung, Speicherplatz, Datenbestände, System- oder Anwendungssoftware etc. (d. h. Ressourcen) miteinander vereint und so zu (Sub-)Systemen oder Ressourcen höherer Ordnung führt. Die Elemente stehen dabei generell in einer komplementären und interaktiven Relation. In der vorliegenden Arbeit ist damit vor allem die Integration von Anwendungssystemen gemeint, die unterschiedliche Aktivitäten eines übergeordneten Geschäftsprozesses einbringen und nach dessen Vorgaben interagieren.

Potenziale und Hindernisse Für Unternehmen ergeben sich durch die Fähigkeit zur Integration von IT-Systemen eine Reihe positiver Eigenschaften für die Organisation. Für die folgenden Aspekte kann in diesem Sinne eine direkte Wirkung festgestellt werden:

- *Flexibilität:* Durch die generelle Möglichkeit zur (Re-)Integration beliebiger Systeme lässt sich die IT-Architektur an Veränderungen der Informationsarchitektur anpassen. In diesem Sinne bleibt die Organisationsstruktur flexibel.
- *Agilität:* (Re-)Integration erlaubt auch die schnelle Reaktion auf unvorhergesehene (unter Umständen unvorhersehbare) Ereignisse.
- *Effizienz:* Die Integration von AS erhöht durch Automation von Daten- und Kontrollfluss die Effizienz entsprechender IS. Zudem sind die Organisationseinheiten durch Systemintegration besser ver- bzw. eingebunden und können sich stärker einbringen.
- *Qualität:* Integration steigert die Qualität organisatorischer Vorgänge. Z. B. bewirkt die Integration eine Erhöhung ihrer Geschwindigkeit. Zudem entfallen potenzielle Fehler bei Datenübertragung und Prozessverlauf.

Insgesamt wirkt die Systemintegration also auf verschiedene Wettbewerbsfaktoren (z. B. Geschwindigkeit, Flexibilität, Qualität), die für moderne Unternehmen auf globalisierten, schnelllebigen Märkten von entscheidender Bedeutung sind. Auf der anderen Seite sehen sich Unternehmen jedoch verschiedenartigen Hindernissen gegenübergestellt:

- *Heterogenität:* Die Komponenten sind oft stark heterogen: z. B. wenn die Komponenten auf unterschiedlichen Abstraktionsebenen operieren, sich in Dimension und Wirkungsbereich unterscheiden, unterschiedliche systemtechnischen Unterstützungsfunktionen nutzen, sich in unterschiedlichen Lebenszyklusphasen befinden, sich in den Schnittstellenmechanismen unterscheiden oder auf unterschiedlichen Paradigmen, Ontologien und Metamodellen aufsetzen.

- *Verteilung:* Die Systeme sind meist physisch über Abteilungen, Niederlassungen oder auch Unternehmen verteilt. Zudem können sie logisch z. B. auf verschiedene Computer verteilt sein.
- *Autonomie:* Die Komponenten sind oft etablierte Systeme, die nicht auf ein Zusammenwirken im globalen Kontext ausgelegt wurden. Sie arbeiten daher sehr autonom. Dies erschwert die Durchsetzung einer globalen Kontrolle.
- *Technologische Evolution:* Durch die rapide Weiterentwicklung von Technologien verkürzt sich der Produktlebens- und -wechselzyklus. Dadurch wird die Integration immer neuer Versionen von Komponenten zum kontinuierlichen Vorgang. Es kommt hinzu, dass dann oft auch Komponenten verschiedener technischer Generationen integriert werden müssen.
- *Interdisziplinärer Bezug:* Bei der Integration müssen Faktoren vieler verschiedener Perspektiven und fachlicher Bezüge berücksichtigt werden, die sich durch technische Evolution noch ausweiten.

Diese und weitere Hürden sind der Grund dafür, dass bis heute keine pauschalen Lösungen existieren und Systemintegration nach wie vor ein Gebiet intensiver Forschung darstellt. Generell wird jedoch der IV hierbei die Rolle einer *integrativen Infrastruktur* zugeschrieben, die eine Vermittlerrolle zwischen den verschiedenartigen Komponenten einnimmt. Leitbild ist dabei ein „Unternehmens-Betriebssystem“ (Enterprise Operation System[5]), das Komponenten im Plug-and-Play-Verfahren in den Betrieb aufnimmt.

Perspektiven und Ebenen Im Detail betrachtet zeigt sich die Systemintegration äußerst facettenreich. Zunächst kann die Integration aus verschiedenen Perspektiven betrachtet werden; die wichtigsten davon sind im Folgenden gelistet:[6]

- *Funktionen/Prozesse:* Bei *Funktionsintegration* werden die sich ergänzenden Funktionen fokussiert. *Prozessintegration* betont dabei den dahinterstehenden Geschäftsprozess.
- *Informationen: Informations-* bzw. *Datenintegration* legt den Schwerpunkt auf die Art der ausgetauschten Informationen.
- *Methoden: Methodenintegration* betrachtet die Vereinheitlichung von (fachlichen) Methoden (z. B. Kalkulationsmethoden).
- *Interoperabilität und Kommunikation:* Eine weitere Perspektive betrifft Kommunikationsverbindungen und Interoperabilitätsmechanismen.

[5] Vgl. z. B. [AMI93].
[6] Vgl. z. B. [MBK+01].

- *Koordination:* Ein wichtiger Aspekt der Integration liegt in der Koordination der Subjekte im Sinne eines gewünschten Gesamtverhaltens.

Ausgehend von der funktionalen Perspektive kann man bei der Systemintegration noch verschiedene Ebenen unterscheiden. Eine derartige Gliederung kann zunächst nach *Integrationsrichtung* erfolgen: Werden operative Systeme entlang der Wertschöpfungskette verbunden, spricht man von horizontaler Integration. Die Anbindung von PuK-Systemen an operative Systeme wird hingegen als vertikale Integration bezeichnet. Damit verbunden ist die *Integrationsreichweite*. Innerhalb einer monolithischen Organisation spricht man von innerbetrieblicher Integration. Ein Unternehmensnetzwerk erfordert hingegen zwischenbetriebliche Integration.

Die *Integrationswirkung* kann dann je nach Charakteristik der dahinterstehenden Prozesskette in einer Halb- oder Vollautomation bestehen. Vollautomation ist das Idealziel der Rationalisierung. Sie lässt sich aber für betriebswirtschaftliche Prozesse oft nur schwer erreichen. Bei Halbautomation werden manuelle und automatische Anteile von Mensch und Maschine kombiniert. Dabei kann entweder der Mensch oder die Maschine die aktive Steuerung des Kontrollflusses übernehmen und dabei die Vorgänge des Prozesses anstoßen.

Vorgehensmodell und Infrastruktur Durch seine Vielschichtigkeit und die Vielzahl an Hindernissen stellt sich der Integrationsvorgang als komplexe Aufgabe dar. Dessen Bewältigung erfordert Methoden, die stets zu einem großen Teil individuell auf die spezifischen Komponenten und Technologiegenerationen abgestimmt sein müssen. Trotzdem existieren Ansätze für generische Vorgehensmodelle. Camarinha-Matos et al. schlagen hierfür ein *generisches Vorgehensmodell* vor [CMA03], das im Folgenden kurz dargestellt werden soll:

1. *Referenzmodell nutzen:* Systemintegration ist Teil eines komplexen Umfelds eng verwobener technischer und betrieblicher Vorgänge. Als Orientierungshilfe ist die Nutzung von Referenzmodellen[7] als idealtypischen Vorgaben zur Einordnung und Abgleich der Integrationsvorgänge ratsam.

2. *System modellieren:* Zur systematischen Planung konkreter Integrationsmaßnahmen und als Kommunikationsgrundlage der beteiligten Akteure sollten Komponenten und Systeme in ihrem Ausgangs- und Zielzustand modelliert werden. Methoden und Werkzeuge können im Einzelfall variieren, sie sollten aber möglichst eng mit dem ganzheitlichen Integrationsprozess abgestimmt sein.

3. *Komponenteninteraktion harmonisieren:* Ein grundlegender Schritt besteht in der Befähigung der zu integrierenden Komponenten zur wechselseitigen

[7] Ein grundlegendes Beispiel wird in Sektion 3.3.1.2 mit dem GERAM Enterprise Architecture Framework gezeigt.

Interaktion; dazu müssen sie ggf. angepasst und harmonisiert werden. Das betrifft zum einen die Schaffung gemeinsamer Schnittstellen(-mechanismen) und Protokolle zum Zugriff auf die Komponentenfunktionen (z. B. durch Umhüllung mit so genannten Wrappern). Zum Teil müssen die beteiligten Systeme aber auch zunächst in sinnvolle Komponenten zerlegt werden.

4. *Informationen und Wissen verknüpfen:* Komplementär zur Harmonisierung der Komponenteninteraktion sind vielfach die durch den Datenaustausch übermittelten Informationen in semantischer Hinsicht aufeinander abzubilden. Die Nutzung von Standards kann hierbei eine Hilfestellung geben, ist aber in der Regel nicht vollständig möglich und führt trotzdem zu Mehrdeutigkeiten.

5. *Koordinationsebene etablieren:* Schließlich besteht ein wesentlicher Schritt darin, die Vorgänge der zu integrierenden Komponenten im Sinne einer gemeinsamen Zielsetzung aufeinander abzustimmen. Dies ist ein integrationsspezifischer Aspekt, der nicht durch die Komponenten selbst abgedeckt werden kann. An dieser Stelle ist deshalb eine zusätzliche Ebene einzuführen, auf der die separaten Aktionen der Komponenten durch geeignete Mittel orchestriert werden. Man kann in diesem Zusammenhang von einem Betriebs- oder Ausführungssystem für das integrierte System sprechen.

Zur Anwendung dieses generischen Vorgehensmodells bedarf es einer Konkretisierung im Sinne möglichst standardisierter Instrumente und Techniken; also z. B. Referenz- und Metamodelle wie GERAM und UML mit entsprechenden Werkzeugen, abgestimmter Interoperabilitätsplattformen wie CORBA, Mechanismen zur Vorgangsunterstützung wie WfMC-RM sowie aufsetzender Interaktionsprotokolle und Ontologien wie EDIFACT.[8]

Das Ziel einer derartigen Zusammenstellung liegt in der Schaffung einer *Infrastruktur der Systemintegration*. Die Hauptaufgabe einer solchen Infrastruktur kann dahingehend identifiziert werden geeignete Bedingungen herzustellen, um eine grundsätzliche Interoperation von Komponenten zu ermöglichen. Darüber hinaus sollten generische Funktionen enthalten sein, um die Kooperation der Komponenten in spezifische Integrationsszenarien zu unterstützen. Diese werden u. a. von unterschiedlichen Integrationsebenen (vor allem in Bezug auf Integrationsreichweite) bestimmt. Hierbei unterscheidet man in der Regel Infrastruktur zur inner- und zwischenbetrieblichen Integration.[9]

[8] Generalised Enterprise Reference Architecture And Methodology (GERAM) [IFI03], Unified Modelling Language (UML) [Sie03], Common Object Request Broker Architecture (CORBA) [OMG04], Workflow Management Coalition Referenzmodell (WfMC-RM) [Hol95] und Electronic Data Interchange for Administration, Commerce and Transport (EDIFACT) [UN/93] stellen hier eine beliebige Auswahl exemplarischer Techniken dar, die kategorisch den Schritten des Vorgehensmodells entsprechen.

[9] In diesem Zusammenhang wird meist von Enterprise Application Integration (EAI) und B2B-Integration (B2Bi) gesprochen; in Sektion 3.3.2 wird dies noch eingehender betrachtet.

Einige in der vorliegenden Arbeit notwendigen und angewandten Integrationsinstrumente solcher Infrastrukturen werden in der übernächsten Sektion 3.3 eingeführt. Zuvor erfolgt jedoch in Sektion 3.2.2 eine Analyse integrierter Informationsverarbeitung im virt. Dienstleistungsunternehmen. Die hierbei identifizierten Anforderungen beeinflussen nämlich nicht nur die spätere Gestaltung von E-Services als spezifische Integrationstechnik, sondern auch den nachfolgenden Blickwinkel auf konkrete Techniken.

3.2.2. Integrierte IV im virt. Dienstleistungsunternehmen

In Bezug auf die Zielsetzung der vorliegenden Arbeit ist primär die betriebliche IV in VDU von Interesse. Für den spezifischen Fall des VDU existiert jedoch bislang kaum diesbezügliches Basiswissen. Die Charakteristik integrierter IV im VDU soll daher auf entsprechende Erkenntnisse für virt. Unternehmen und Dienstleistungsunternehmen zurückgeführt werden.

Das Organisationsprinzip des virt. Unternehmens prägt als Basis des VDU dessen grundlegende IV-Charakteristik. Dies lässt sich u. a. an der IV-Strategie erkennen, die im Falle von VU diejenige von DLU in entscheidenden Teilen subsumiert. Für eine konkretere Architektur ergeben sich dann komplementäre Anforderungen aus beiden Bereichen. Hierfür wird in Sektion 3.2.2.1 ein homogener Satz an Kategorien identifiziert.

Die Anforderungen resultieren in der Notwendigkeit einer umfassenden integrativen Infrastruktur, die es autonomen Teilnehmern eines VDU erlaubt, in agiler Weise flexible Informationsstrukturen aufzubauen und zur Kollaboration zu nutzen. Eine solche Infrastruktur kann im generellen Fall virtueller Organisationsstrukturen logisch in zwei Teile strukturiert werden [CMA03, S. 662 .f]: Zum einen obliegt einer (horizontalen) Basisinfrastruktur die Aufgabe Interoperabilität herzustellen sowie das Teilen und den Austausch von Daten mitsamt der Koordination zu ermöglichen. Zum anderen kommt fachlich basierten Funktionen die Aufgabe zu, das Unternehmen in allen Phasen des Lebenszyklus zu unterstützen. Die beiden Teile werden in den Sektionen 3.2.2.2 und 3.2.2.3 behandelt.

3.2.2.1. Zielsetzung und Anforderungen für den IV-Einsatz

Die Untersuchung der IV im VDU startet bei deren Zielsetzung. In ihrer allgemeinsten Form geht diese als IV-Strategie aus den Unternehmenszielen hervor. Mertens unterscheidet hierbei drei Kategorien von IV-Strategien, die im Unternehmen idealtypisch in konsekutiven Phasen durchlaufen werden [Mer00] (vgl. Tab. 3.1). Der Zweck des IV-Einsatzes liegt dabei in den ersten beiden Kategorien in der Unterstützung klassischer Unternehmensstrategien: Optimierung unternehmensinterner Kernkompetenzen und Beeinflussung unternehmensexterner Beziehungen (z. B. Marktposition, Wettbewerbsvorteile). Spätestens ab der dritten Kategorie nimmt die IV dann durch

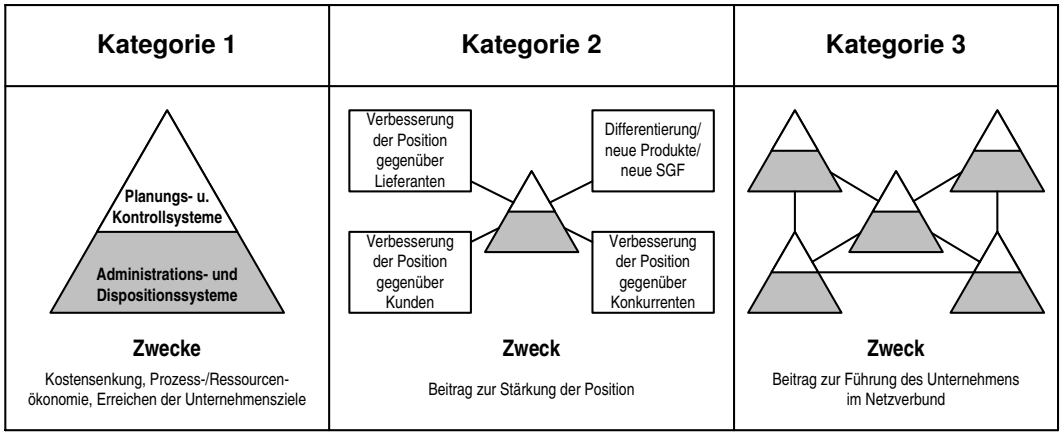

Tabelle 3.1.: IV-Strategien (nach [MBK+01, S. 198])

Befähigung zur Führung des Unternehmens im Netzwerk auch Einfluss auf die Unternehmensstrategie.

Im DLU stehen die Merkmale von Dienstleistungen und ihre Konsequenzen für Beschaffung, Produktion und Absatz im Vordergrund. Aus dieser Perspektive liegt der Zweck des IV-Einsatzes in der Unterstützung von Dienstleistungstransaktionen [MBK+01, S. 129]. Dies schließt vor allem Rationalisierungen bei der Bildung des Potenzials zur Erbringung von Dienstleistungen und Optimierung von deren Wirkung, aber auch die Interaktion mit Kunden und Lieferanten ein. In Bezug auf VU wird hingegen die Organisationsform betont. Bei dieser Sichtweise liegt der Zweck des IV-Einsatzes schwerpunktmäßig in einer Unterstützung der Unternehmensführung im Netzwerk [MGE98, S. 67]. Die Flexibilität und Dynamik dieser spezifischen N-Form wird hierdurch überhaupt erst möglich: eine IV-Unterstützung des VU-Lebenszyklus muss nämlich die autonomen Netzwerkunternehmen zur dezentral gesteuerten Kollaboration bei Formation, Durchführung und Auflösung von in sich geschlossenen Missionen befähigen. Es kann festgestellt werden, dass beide Strategien wesentliche Anteile in der 3. Kategorie aufweisen. Dabei bezieht sich diese Strategie im VU in umfassender Weise auf den ganzen Lebenszyklus, während bei DLU in diesem Sinne vor allem operative Aspekte des DLP eine Rolle spielen. Insgesamt ergibt sich für die IV-Strategie im VDU ein Schwerpunkt in der Unterstützung der zwischenbetrieblichen Integration.

Die strategische Kategorie gibt allerdings zunächst nur eine sehr generelle Zielrichtung vor. In Bezug auf die spezifische IV in VDU ist nun vor allem die Realisierung der Strategie durch konkrete Infrastruktur von Interesse. Um sich dieser zu nähern, ist eine Betrachtung der spezifischen Anforderungen von fachlicher Seite erforderlich. Diese werden im Folgenden zunächst für VU/DLU einzeln analysiert und schließlich unifiziert.

Anforderungen an die IV im virt. Unternehmen IV-Strategien zur Führung von VU-Netzwerken resultieren in hohen Anforderungen an IT-Architekturen und -Systeme der Netzwerkunternehmen. Aus der fachlichen Perspektive lässt sich ein genereller Satz von Anforderungen aus den konstituierenden VU-Merkmalen[10] ableiten[11]:

1. *Autonome Kooperationspartner* – Als erstes Merkmal ist festzustellen, dass VU „... eine Kooperationsform rechtlich unabhängiger Unternehmen (sind), die eine Leistung auf Basis eines gemeinsamen Geschäftsverständnisses erbringen". Kooperation und gemeinsames Geschäftsverständnis im Unternehmensnetzwerk beruhen u. a. auf Teilung und Austausch von Informationen. Dazu müssen IT-Systeme aller Netzwerkunternehmen im Sinne einer gemeinsamen Kommunikationsinfrastruktur vernetzt und interoperabel sein. Die Fluktuation autonomer Teilnehmer macht Sicherheitsmechanismen und Standardkonformität notwendig.

2. *Kompetenzformation („Breeding")* – Das zweite Merkmal besagt, dass „die kooperierenden Organisationseinheiten des VU [...] sich an der horizontalen oder vertikalen Zusammenarbeit vorrangig mit ihren Kernkompetenzen (beteiligen)". Die Planung/Entwicklung von Missionen (Breeding) bedingt fachliche Unterstützungsfunktionen zur Identifikation/Vermittlung von Kompetenzen/Partnern (Brokerage) [CMA03, S. 670 ff.]. Dazu gehört auch eine Unterstützung in Bezug auf das Management der zugrunde liegenden Pools/Cluster.

3. *Homogene Außenwirkung* – Ein drittes Merkmal ist, dass „die kooperierenden Einheiten [...] bei der Leistungserstellung gegenüber Dritten wie ein einheitliches Unternehmen (wirken)". Diese einheitliche Wirkung erfordert u. a. entsprechend einheitliche Kunden-Schnittstellen für integrierte AS verschiedener Partner.

4. *Selbstabstimmung* – Das vierte Merkmal ist entscheidend für die Differenzierung des VU unter den N-Formen: „Im VU wird auf die Institutionalisierung zentraler Managementfunktionen zur Gestaltung, Lenkung und Weiterentwicklung des VU weitgehend verzichtet und der notwendige Koordinations- und Abstimmungsbedarf durch geeignete IS gedeckt". Dieses Merkmal führt zur Forderung nach generischen und fachlichen Unterstützungsfunktionen zur Kooperation von Netzwerkunternehmen in Selbstabstimmung. Generische Funktionen umfassen Teilung und Austausch von Informationen/Wissen ergänzt um Mechanismen zur Koordination von und Kollaboration zwischen Netzwerkunternehmen. Auf fachlicher Ebene ist der VU-Lebenszyklus zu unterstützen [CM05, S. 93 ff.]. Hierzu zählen initial im Wesentlichen Breeding (s. o.), operational zwischenbe-

[10] Zu den Merkmalen vgl. Def. 8 u. Sektion 2.3.2.2, S. 51.

[11] Die Analyse basiert auf [MGE98, S. 68]; für die vorliegende Arbeit wurden jedoch abstraktere Anforderungskategorien hergeleitet.

triebliche Geschäftsprozesse sowie branchenspezifische Funktionen und final u. a. Management von After-Sales-Sevices (s. u.).

5. *Missionscharakter* – Das fünftes Merkmal betrifft schließlich die zeitliche Komponente: „Das VU ist mit einer Mission verbunden und endet mit dieser". Der regelmäßige Wechsel zeitlich begrenzter Missionen bedingt die Flexibilität von Kooperationsstrukturen und die Agilität entsprechender Integrationsmethoden auf allen Ebenen. Die Auflösung von VU nach Missionen erfordert spezifische Funktionen für das Management von After-Sales-Service, z. B. bei Support und Gewährleistung.

Aus der Analyse gehen eine Reihe von Forderungen nach Qualitäten einer integrativen Infrastruktur hervor. Diese stellen die wesentliche Vorgabe für Anforderungen von VDU dar und sind nun durch eine entsprechende Analyse von DLU zu ergänzen.

Anforderungen an die IV im Dienstleistungsunternehmen Die konstituierenden Eigenschaften von Dienstleistungen und Dienstleistungsproduktion führen zu spezifischen Anforderungen an die Informationsverarbeitung im DLU. Im Mittelpunkt steht dabei der immaterielle und interaktive Kernleistungsprozess, der von DLU in mehrstufigen Verfahren Kundenindividuell produziert bzw. erbracht wird[12]. Einzelne Anforderungen können aus der Dienstleistungsdefinition[13] abgeleitet werden:

1. *Potenzialvermarktung* – Aus der potenzialorientierten Sicht („Dienstleistungen sind selbstständige, marktfähige Leistungen, die mit Bereitstellung und/oder dem Einsatz von Leistungsfähigkeiten verbunden sind.") folgt die Anforderung nach fachlicher Unterstützung zur Vermarktung und Vorkombination. Die Vermarktung von Dienstleistungen erfordert Integration mit dem Kunden durch E-Commerce-Mechanismen, die als elektronische Märkte für Dienstleistungen fungieren können.[14] Dies setzt Vernetzung des Front Office voraus. Die Vorkombination zeigt sich als autonomer Produktionsprozess, der weitgehend durch Standard-IV (z. B. PPS-Systeme) abgedeckt werden kann.

2. *Interaktiver Leistungsprozess* – Die prozessorientierte Sicht („Interne und externe Faktoren werden im Rahmen des Leistungserstellungsprozesses kombiniert.") betont die Interaktivität des Kernleistungsprozesses. Das betrifft zum einen die Interaktion mit dem Kunden. Zum anderen müssen auch Lieferanten und Partnerunternehmen simultan einbezogen werden. Hierfür sind generische und fachliche Unterstützungsfunktionen zur Kooperation der Akteure erforderlich. Besonders die zwischenbetriebliche Kooperation im Back Office braucht generische Mechanismen für Informationsaustausch, Koordination und verschiedene

[12] Vgl. Sektion 2.2.
[13] Vgl. Def. 1, S. 14.
[14] Vgl. z. B. Übersicht in [MTL99].

Formen der Kollaboration. Des Weiteren müssen hierfür das Management zwischenbetrieblicher DLP und oftmals branchenspezifische Funktionen unterstützt werden. Grundvoraussetzung ist zwischenbetriebliche Vernetzung und Interoperabilität.

3. *Individuelle Wirkung* – Aus der ergebnisorientierten Sicht („Die Faktorkombination des Dienstleistungsanbieters wird mit dem Ziel eingesetzt, an den externen Faktoren – Menschen oder deren Objekten – nutzenstiftende Wirkungen zu erzielen.“) zeigt sich die Immaterialität und Individualität von Dienstleistungen. Immaterialität hebt die Information als wesentlichen Produktionsfaktor hervor und eröffnet erhebliche Potenziale für IV. Immaterialität erschwert aber auch das Verständnis abstrakter Leistungen/Prozesse und stellt hohe Anforderungen an integrierte Kundenschnittstellen. Individualität der Leistung bedingt Flexibilität von Kooperationsstrukturen inkl. Dienstleistungsprozess und Agilität der Integrationsmethoden. Zudem bedingt Kundenwechsel Standardkonformität und Sicherheitsmechanismen im Front Office.

Die Anforderungen lassen erkennen, dass sich die IV im DLU nicht so sehr durch typische Anwendungssysteme differenziert (diese sind eher fall- oder branchenspezifisch). Vielmehr stehen wie beim VU Integrationsaspekte im Vordergrund und viele der schon dort identifizierten Anforderungen finden sich wieder.

Kategorien von Infrastrukturanforderungen im VDU Auf Basis der Einzelbetrachtungen können die Anforderungen an eine integrative Infrastruktur für VDU nun durch Unifikation und Aggregation angenähert werden. Tabelle 3.2 fasst zunächst die Teilergebnisse zusammen. Die Tabelle zeigt noch mal die einzelnen VDU-Merkmale und ordnet ihnen die oben gefundenen Anforderungen in vereinheitlichter Form zu.

Die Anforderungen beziehen sich auf verschiedene Aspekte der Infrastruktur. Entsprechend sind sie in Kategorien eingeteilt. Zunächst können Anforderungen identifiziert werden, die sich an eine generische (horizontale) Basisinfrastruktur richten. Sie betreffen Funktionen und Eigenschaften in Bezug auf die allgemeine Kommunikation und Kooperation von Akteuren innerhalb und außerhalb des Netzwerks zur Dienstleistungserbringung. Davon unterscheiden sich Anforderungen an fachspezifische (vertikale) Unterstützungsfunktionen des VU-Lebens- bzw. Dienstleistungszyklus, die sich grob bzgl. Einleitung, Betrieb und Auflösung strukturieren lassen. Daneben beziehen sich einige komplementäre Anforderungen auf die gesamte Infrastruktur.

3.2.2.2. Mechanismen einer VDU-Basisinfrastruktur

Zur zwischenbetrieblichen Integration von VDU wird zuallererst eine fundamentale Basisinfrastruktur benötigt. Diese ermöglicht die grundlegende Kommunikation

	Merkmale / Anforderungskategorien	Infrastruktur zur zwischenbetrieblichen Integration im VDU												
		Horizontale Basisinfrastruktur						Vert. Fachfunktionen im VU/DLP-Zyklus						
		Kommunikation			Kooperation			Einleitung	Betrieb		Auflösung			
		Vernetzung	*Interoperabilität / Systemintegration*	*Sicherheit*	*Info- & Wissensteilung*	*Koordination*	*Kollaboration*	*Breeding Umgebung / Elektr. Dienstmarkt*	*Verteilte Geschäftsprozesse*	*Branchenfunktionen*	*Management von After Sales Services*	*Standardkonformität*	*Agilität*	*Flexibilität*
VU	*Aotonome Koop.-Partner*	x	x	x								x		
VU	*Kompetenzformation*							x						
VU	*Homogene Außenwirkung*		x											
VU	*Selbstabstimmung*				x	x	x	x	x	x	x			
VU	*Missionscharakter*										x		x	x
DLU	*Potentialvermarktung*	x						x						
DLU	*Interakt. Leistungsprozess*		x		x	x	x		x	x				
DLU	*Individuelle Wirkung*		x	x							x	x	x	x

Tabelle 3.2.: Anforderungen an die VDU-Infrastruktur

von IT-Systemen der Netzwerkunternehmen und hilft ihnen bei der Kooperation untereinander sowie mit Kunden und Lieferanten.

Kommunikationsinfrastruktur Kommunikation basiert zuallererst auf Vernetzung der Partnerunternehmen. Die Infrastruktur muss dazu nach Reinhart [RM00] vier Anforderungen erfüllen:

1. Gewährleistung des flächendeckenden Zugriffs für Netzwerkunternehmen
2. Vermeidung hoher Kosten bei Anschluss/Betrieb des Kommunikationsmediums
3. Vermeidung von Spezialwissen bei Nutzung des Kommunikationsmediums
4. Gewährleistung von Kommunikationssicherheit

Diese Bedingungen werden heute im Wesentlichen durch das Internet[15] erfüllt [RM00, Sie99a, Fis00], das meist als Basis der Kommunikation im VU und DLU angesehen wird.

[15]Der Begriff des Internets geht auf eine facettenreiche historische Entwicklung rund um die Vernetzung von IT-Systemen auf Basis des TCP/IP-Referenzmodells zurück, die mit der Entwicklung des ARPANET ihren Ausgang nahm (vgl. z. B. [Tan86, S. 44 ff.]). In der vorliegenden Arbeit werden damit zunächst grundsätzlich die unteren Schichten (1-4) von Open Systems Interconnect (OSI) abstrahiert [Ker92]; d. h., es wird eine Transportschicht zur Verbindung von Systemen vorausgesetzt. Darüber hinaus wird – soweit nicht explizit ausgeschlossen – eine Menge von damit in Verbindung stehenden (de facto) Standards umschrieben, die zu einer Reihe von weithin akzeptierten Fähigkeiten und Eigenschaften auf fachlicher Ebene des Anwenders führen. Dazu gehören u. a. Daten und Informationsaustausch (z. B. FTP, Hypertext), Individual- und Gruppenkommunikation (z. B. E-Mail, News), fundamentale Sicherheitsmechanismen (z. B. Verschlüsselung und Signatur mit Public Key-Verfahren).

In DLU ist die Kommunikationsinfrastruktur in verschiedene Bereiche geteilt. Hier werden nämlich Anwendungssysteme im Sinne des Dienstleistungsprozesses strukturiert, der einen Kernprozess mit Kundenkontakt (Front Office Funktionen) sowie einen Unterstützungsprozess ohne Kundenkontakt (Back Office Funktionen) beinhaltet. Diese Struktur wirkt sich auf die Integrationsreichweite bzw. Vernetzung der AS mit verschiedenen Akteuren aus. Dabei sind die Systeme des DLU je nach Aufgabe zum Teil intern, zum Teil mit Kunden und zum Teil mit Partnern/Zulieferern in geeigneter Weise zu vernetzen.[16]

Auf Basis der Vernetzung muss die Infrastruktur für bestimmte Formen der Kooperationsunterstützung eine offene verteilte Systemumgebung zur Interoperabilität, Interaktion und Integration von Anwendungssystemen über die grenzen einzelner Unternehmen bereitstellen. Dazu wurde für VU z. B. Middleware[17] vorgeschlagen [Fis00, S. 444 f.]. Deren Aufgabe ist es, eine Interoperabilitätsplattform für die potenziell heterogenen Systeme verschiedener Netzwerkunternehmen bereitzustellen. Dabei sollen sich die Systeme beim Aufbau einer Kooperationsbeziehung möglichst nach dem „Plug and Play"-Prinzip flexibel und agil koppeln lassen [AFH+95, S.13]. Hierfür wurden in der Literatur u. a. die Paradigmen der Komponenten [MGE98, S. 76], Business Objects [Fis00, S. 430 f.] und Grid Ressourcen [FK99, S. 9] diskutiert.

Aktuell werden derartige Mechanismen auch verstärkt im Zusammenhang mit verschiedenen Paradigmen der Kooperationsunterstützung wie z. B. Agenten und Services untersucht. Deren Darstellung erfolgt im nächsten Abschnitt.

Kooperationsunterstützung Unter Voraussetzung einer grundlegenden Kommunikationsinfrastruktur unterscheiden frühe Ansätze drei fundamentale Kategorien von IT-Architekturen zur Kooperation im VU[18]: Basiskommunikation (bzw. „Applikations-Kommunikation"), Datenverteilung (bzw. „Daten-Sharing") und Anwendungsverteilung (bzw. „Applikations-Sharing") (vgl. Abb.3.3). Das Differenzierungsmerkmal liegt hierbei in der Form der Kooperationsunterstützung bei verschiedenen Stufen der Systemintegration.

Die einfachste Form der Integration ist die Basiskommunikation. In diesem Fall dienen Anwendungssysteme zum manuellen Austausch arbeitsbegleitender Daten- und Informationen. Bei der Datenverteilung liegt eine gemeinsame Datenbasis vor, die VU-weit genutzt wird (z. B. durch ein verteiltes Datenbankmanagementsystem (DBMS)). Dabei sind die Integrität und Sicherheit der Daten zu gewährleisten und Zugriffsrechte zu berücksichtigen. Bei Anwendungsverteilung erfolgt schließlich eine gemeinsame Nutzung einheitlicher, verteilter Anwendungssysteme „... im Sinne von Groupware" [AFH+95]. Dies können interaktiv genutzte Anwendungssysteme sein. Hierbei besteht aber auch die Möglichkeit zur durchgängigen Automation kooperativer

[16] Vgl. Abb. 3.4.

[17] Eine Vertiefung dieser Techniken erfolgt in Sektion 3.3.2.

[18] Vgl. [AFH+95], [Fai98, S. 52 ff.], [MGE98, S. 78 ff.], [Sie99a].

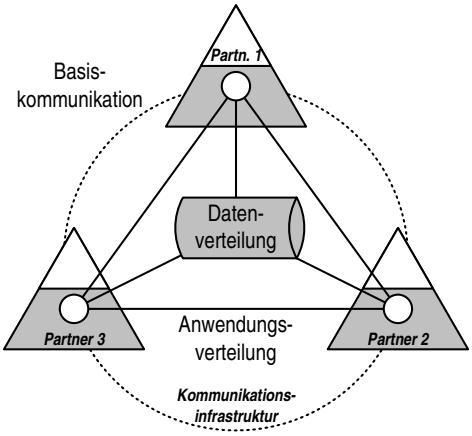

Abbildung 3.3.: Architekturen zur Kooperationsunterstützung im VU (nach [AFH+95])

Vorgänge. Auf Grund der temporären Kooperationen ist dann allerdings die schnelle und effiziente Entwicklung automatisierter Lösungen von entscheidender Bedeutung [MGE98, S. 81 ff.].

Im Zusammenhang mit diesen Architekturkategorien nennt Faisst drei sich zum Teil überschneidende Gruppen generischer Unterstützungsfunktionen zur Kooperation im virt. Unternehmen [Fai98, S. 54 ff.]:

- *Kommunikationsunterstützung:* Befähigung menschlicher Akteure zur Kommunikation in verschiedenen synchronen/asynchronen, bilateralen/multilateralen Formen, z. B. durch E-Mail, Newsgroups, Videokonferenzen.

- *Koordinationsunterstützung:* Befähigung der Netzwerkunternehmen zur Regelung ihrer kooperativen Vorgänge mit wechselseitigem Abstimmungsbedarf, z. B. durch Vorgangsmanagement mit Hilfe von Projekt-Management-Systemen oder Workflow-Management-Systemen.

- *Kollaborationsunterstützung*[19]*:* Befähigung von menschlichen Akteuren zur gemeinsamen unstrukturierten Arbeit durch (1) gemeinsame Informationsräume auf Basis verteilter Hypertext- oder Datenbankmanagementsysteme sowie (2) Workgroup Computing, z. B. in Form von Planungs-, Entscheidungs- und Sitzungsunterstützungssystemen oder Gruppeneditoren.

Ganz ähnliche Funktionsgruppen werden auch im Zusammenhang mit DLU genannt, wo vor allem die Durchführung des Dienstleistungsprozesses große Ansprüche an die Kooperationsunterstützung der Basisinfrastruktur stellt. Die Kooperation

[19]Die ursprüngliche Bezeichnung „Kooperationsunterstützung“ wurde auf Grund der bisherigen Begriffswahl angepasst.

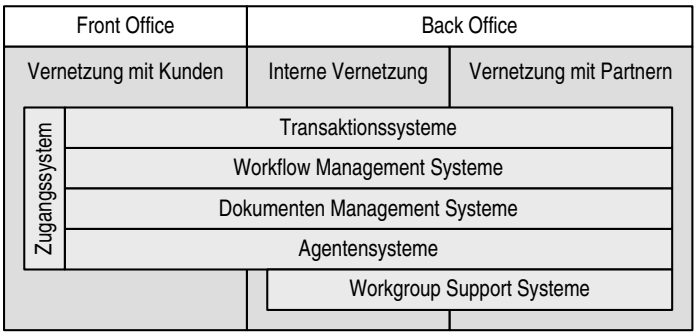

Abbildung 3.4.: Basisinfrastruktur im DLU (nach [MBK+01, S. 130])

findet hier in Form individueller Interaktionen zwischen DLU, Kunde und Partnerunternehmen/Lieferanten statt.

Zur Unterstützung wurden Koordinations- und Kollaborationsfunktionen wie Transaktionssysteme, Workflow-Management-Systeme (WfMS), Dokumentenmanagementsysteme (DMS) oder Workgroup Support Systeme (WSS), aber auch Agentensysteme vorgeschlagen. Auf Kundenseite kommen dabei oft spezifische Zugangssysteme, z. B. in Form von WWW-Portalen oder Kiosk-Systemen, zum Einsatz [MBK+01, S. 128 ff.]. Abbildung 3.4 zeigt Einbettung und Reichweite der Kooperationsunterstützung im Zusammenhang mit der Kommunikationsinfrastruktur.

Erweiterte Paradigmen Über die basalen Architekturkategorien hinaus werden in der aktuellen Literatur drei zum Teil komplementäre Paradigmen für Basisinfrastrukturen in virt. Unternehmen diskutiert[20]:

- *Schichten/Transaktionen* – Hierbei wird auf den Schichten der innerbetrieblichen IT-Architekturen einzelner Netzwerkunternehmen eine zusätzliche *Kooperationsschicht* aufgesetzt. Die Kooperation zwischen VU-Partnern wird dann in Form von Transaktionen über diese Schichten abgewickelt. Im Mittelpunkt der hier angebotenen Funktionalität steht der Austausch technischer Daten auf Basis netzwerkweiter Standards. Des Weiteren ist die Integration mit vorhandenen AS auf Basis von Interoperabilitätsplattformen/Middleware (z. B. verteilte Objektsysteme) von großer Bedeutung. In diesem Zusammenhang werden in der Regel auch Koordinationsmechanismen berücksichtigt, die oft auf Workflow-Varianten basieren. Zudem sind in der Kooperationsschicht auch Mechanismen für ein verteiltes Management von Informationen und VU-spezifische Sicherheitsrichtlinien zu finden.

[20]Die hier beschriebenen Paradigmen wurden durch eine Studie internationaler Projekte identifiziert, die im Kontext des Europäischen IST VOSTER Projekts durchgeführt wurde [CM05, S. 85 ff.].

- *Agenten* – Bei Agenten-basierten Infrastrukturen werden Netzwerkunternehmen als Agenten und VU als Multiagentensystem im Sinne einer KI-Sicht repräsentiert[21]. Dieses Vorgehen wird durch vielfältige Parallelen zwischen MAS und VU motiviert. Hierbei wirken sich jedoch die Implikationen des (Software-technischen) Agentenparadigmas stark auf die Art der Kooperationsmechanismen aus. In diesem Sinne stehen marktorientierte Verhandlungs- und Vertragsmechanismen im Vordergrund, die sich wiederum in erster Linie auf die Initialisierungsphase im VU-Lebenszyklus anwenden lassen. Anforderungen im Sinne des VU-Betriebs wie Koordination, Kontrolle und Sicherung von verteilten Kooperationsvorgängen oder Informationsaustausch/-teilung werden hingegen bislang nur rudimentär unterstützt.

- *Services* – Bei Service-basierten Infrastrukturen werden die (Kern-)Kompetenzen und das damit einhergehende Kooperationsverhalten von Netzwerkunternehmen durch eine Menge von Services repräsentiert. Services bieten dazu die Möglichkeit zur klaren Spezifikation von Inhalten (syntaktisch und semantisch). Zudem ist der Zugriff darauf über standardisierte Schnittstellen möglich, die unabhängig von der dahinter stehenden Realisierung sind. Eine Registratur *(Registry)* verbreitet Dienstangebote der Netzwerkunternehmen im Pool/Cluster. Nachfrager *(Requestors)* können hier passende Angebote zur Initialisierung von VU finden und werden zum Erbringer *(Provider)* vermittelt. Hier erhalten sie auch die eindeutige Referenz eines Service-Proxy, über den sie auf den Service in der Betriebsphase des VU zugreifen können. Ferner sieht das Service-Paradigma die Koordination und Komposition verschiedener „atomarer“ Services auf Basis von Prozessen vor, die konzeptionell zu (verteilten) Geschäftsprozessen konform gehen.

Obwohl mit den einzelnen Paradigmen recht unterschiedliche Perspektiven auf die VU-Kooperation einhergehen, schließen sie sich nicht gegenseitig aus. So können etwa Serviceaufrufe auf Basis von Transaktionen einer Kooperationsschicht realisiert sein oder durch Agenten hervorgerufen werden. Die Paradigmenwahl wird dabei in erster Linie von der Anwendungsanalyse beeinflusst. Konkret scheint sie sogar von der Branche abhängig zu sein. Camarinha-Matos et al. weisen in diesem Zusammenhang darauf hin, dass es besonders vorteilhaft sei, Infrastrukturen auf Basis von Schichten/Transaktionen in der Industrie einzusetzen und solche auf Basis von Services im Dienstleistungssektor [CMA03, S. 668].

Der letzte Hinweis deckt sich vollkommen mit den Ansätzen der vorliegenden Arbeit. In diesem Sinne werden entsprechende Techniken des Service Oriented Computing in Sektion 3.4 noch ausführlich dargestellt.

[21] Agenten sind hierbei autonome Systeme mit flexiblem und proaktivem Verhalten, die im Kontext eines MAS interagieren und dabei im Sinne einer gemeinsamen verteilten Problemlösungsstrategie handeln.

Die Rolle von Standards Die Integration wechselnder Netzwerkunternehmen führt dazu, dass die zu integrierenden Architekturen und Systeme nicht a priori bekannt sind. Unter diesen Bedingungen ist Standardkonformität u. a. in den Bereichen der Kommunikation, des Datenaustauschs und der Anwendungen unumgänglich [MGE98, S. 69].

Eine grundlegende Basis bilden hier z. B. Referenzmodelle für Architekturen und Systeme wie z. B. OSI, ODP[22] oder GERAM der ISO. Hier werden Ansätze für die generellen Aspekte und Vorgehensweisen bei Entwurf und Realisierung von Kommunikationsinfrastruktur und verteilten Systemen im ganzheitlichen Kontext des Unternehmens (Netzwerks) betrachtet. Des Weiteren sind u. a. Standards für Middleware (z. B. CORBA), Austauschformate (z. B. EDIFACT), formale Spezifikationssprachen (z. B. STEP[23]), Branchen (z. B. RosettaNet), Normrahmen (z. B. CALS[24]), WfMS (z. B. WfMC-Referenzmodell), betriebswirtschaftliche Anwendungslogik und Ablauforganisation gebräuchlich.

Generelle Impulse für die Standardisierung im VU/VDU kommen aus dem Bereich des B2B-Integration (B2Bi), der sich mit Fragestellungen im Zusammenhang allgemeiner zwischenbetrieblicher Integration befasst. Ein Überblick folgt in Sektion 3.3.2.4.

3.2.2.3. Unterstützungsfunktionen im VDU-Lebenszyklus

Das Zusammenwirken verschiedener Unternehmen im VDU erfordert neben generischen auch verschiedene fachliche Kooperationsmechanismen. Als Bezugsrahmen wird meist der Lebenszyklus von VU bzw. DLP herangezogen. Anhand deren Phasen erfolgt dann die Strukturierung der verschiedenen Funktionen. Insgesamt ist dabei festzustellen, dass die Grenze zwischen Funktionen der horizontalen und vertikalen Infrastruktur fließend ist und eine Zuordnung zum Teil von der Perspektive der Betrachtung abhängt.

Unterstützungsfunktionen im VU-Lebenszyklus VU bilden sich temporär aus strategischen Unternehmensnetzwerken (VU-Typ3)[25] oder ad hoc am Markt (VU-Typ1,2)[25], um spezifische Missionen zu erfüllen. Sie enden nach deren Abschluss. Dieser zyklische Vorgang wird auch als *VU-Lebenszyklus* bezeichnet [Fai98]. Faisst unterscheidet hier Identifikations-, Anbahnungs-, Vereinbarungs-, operative und Auflösungsphase. Fachliche Unterstützungsfunktionen bzw. entsprechende operative und PuK-Systeme können nun in zwei Gruppen unterteilt werden: Die erste Gruppe unterstützt spezifische Phasen des VU-Lebenszyklus. Die zweite Gruppe kommt unabhängig vom Lebenszyklus zum Einsatz. Abbildung 3.5 zeigt einige Beispiele.[26]

[22] Open Distributed Processing (ODP) [IJ95].
[23] Standard for the Exchange of Product Model Data (STEP), ISO 10303.
[24] Computer Aided Acquisition and Logistics Support (CALS), siehe [MGE98, S. 74].
[25] Siehe Sektion 2.3.2.2, S. 56.
[26] Siehe hierzu [Fai98, S. 64 ff.], [MGE98, S. 93 ff.], [AFH+95], [Mer99, S. 86 ff.].

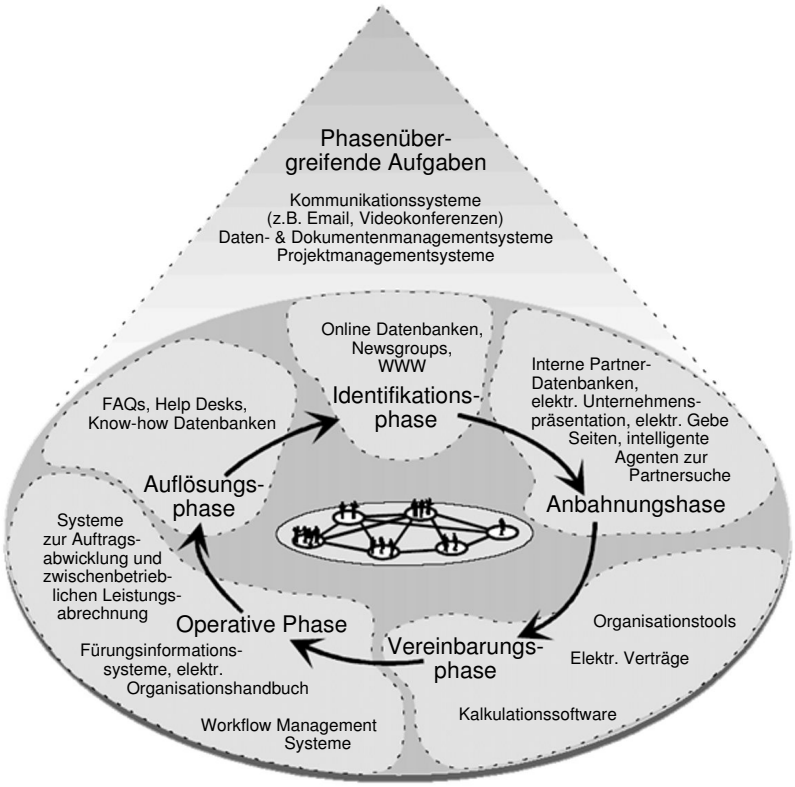

Abbildung 3.5.: Unterstützungsfunktionen im VU-Lebenszyklus [MF97, S. 125]

Der Lebenszyklus beginnt mit der *Selektion*[27] von externen Unternehmen für das strategische Unternehmensnetzwerk (VU-Typ3) oder für die Mission (VU-Typ1,2). Bei deren Identifikation kann vor allem das World Wide Web (WWW) mit Schnittstellen zu (branchen-)spezifischen Community Support Systemen (CSS), öffentliche Unternehmens-DBs, Foren etc. hilfreich sein [Sie99a]. Bei der Allokation von Netzwerkunternehmen für eine Mission (VU-Typ3) kann auf ähnliche, aber umfassendere Identifikationsmechanismen innerhalb des strategischen Netzwerks zurückgegriffen werden. Zusätzlich können hier auch intelligente Agenten zur spezifischen Unterstützung der einzelnen Akteure verwendet werden [Kir95]. In Verbindung mit Auktionsmechanismen können MAS zur Realisierung interner Märkte dienen [CG00].

Der nachfolgende Aufgabenbereich der *Regulation* muss im VU besonders effizient realisiert werden. Während der Vereinbarungsphase können Planungssysteme die Gestaltung erleichtern[28]. Administrations- und Dispositionssysteme helfen dann, die Planung zügig umzusetzen[29]. Die operative Phase umfasst im Wesentlichen

[27] Vgl. Aufgabenbereiche des Netzwerkmanagements in Sektion 2.3.2.1, S. 47.
[28] Z. B. Kalkulations-Software zur zwischenbetrieblichen Leistungsverrechnung.
[29] Z. B. elektr. Verhandlungsmechanismen und Verträge.

Entwicklung und Produktion von Gütern und/oder Dienstleistungen. Hier kommen bei den Netzwerkunternehmen AS zum Einsatz, die die verteilte Operation im VU berücksichtigen[30].

Die *Evaluation* kann im Rahmen integrierter FIS durch normalisierte Kontrollsysteme unterstützt werden. In der Auflösungsphase herrschen schließlich rechtliche und administrative Aufgaben vor. Hierbei können die Netzwerkunternehmen u. a. durch spezifische Wissensmanagementsysteme unterstützt werden. Diese stellen Informationen über frühere Abläufe und Verfahren zur Verfügung und konservieren neues Wissen für zukünftige Lebenszyklen.

Zu den phasenübergreifenden AS gehören zunächst einige komplementäre operationale Systeme. Daneben sind hier einige Funktionen von PuK-Systemen einzuordnen. Zu den operativen Systemen gehören z. B. WWW-basierte Präsentationssysteme und Schnittstellen. Diese werden von allen Netzwerkunternehmen und in allen Phasen genutzt, um eine einheitliche Darstellung des VU gegenüber Kunden zu erreichen. Eine weitere wichtige Klasse von Anwendungssystemen dient dem Wissensmanagement. Dazu gehören z. B. elektronische Organisationshandbücher (ELO) sowie Wissens-DBs und -verteilungssysteme. Bei den PuK-Systemen sind FIS zu nennen. Diese leisten phasenspezifische Informationsbereitstellung und Entscheidungsunterstützung. Dazu gehören aber auch VU-weite Kommunikationsmechanismen, die phasenübergreifend zum Einsatz kommen. Hierzu gehören z. B. Message Handling und Konferenzsysteme.

Unterstützungsfunktionen im Dienstleistungslebenszyklus Im DLU kommt der IV die Aufgabe zu, den Lebenszyklus von Dienstleistungen zu unterstützen. Dafür lässt sich aus der potenzial-, prozess- und ergebnisorientierten Dienstleistungssicht ein Vorgangsmodell mit *Vorkontakt-*, *Kontakt-* und *Nachkontaktphase* der Akteure ableiten[31] [Hen92, S. 21]. Bodendorf nimmt noch eine Verfeinerung dieses Grundmodells vor, deren Phasen in Abbildung 3.6 gezeigt sind [Bod99].

Entsprechende Unterstützungsfunktionen können nun aus den Phasen des Dienstleistungslebenszyklus abgeleitet werden: In der Vorkontaktphase müssen zunächst der Bedarf geplant und die Leistungsbereitschaft hergestellt werden. Dabei werden z. B. spezifische PPS- oder auch Yield Management Systeme (YMS) genutzt. Zum anderen erfolgt die Information und Beratung der Kunden auf Basis des Marketings, was fließend in einen flexiblen Vereinbarungsteil der Kontaktphase übergeht. Hierbei gehen die Vorgänge zum großen Teil mit dem E-Commerce konform [Mer02, S. 30 f.]. Entsprechende Anwendungssysteme im Front Office umfassen dann z. B. Präsentations-, Auskunfts- und Beratungssysteme über das Internet [Alp96, S.174 ff.].

[30] Z. B. PPS-Systeme mit spezifischen Funktionen zur verteilten Produktion, vgl. Sektion2.3.3.1.

[31] Dieser Ablauf wird in der Wirtschaftsinformatik zum Teil als „Dienstleistungsprozess“ bezeichnet [MBK+01, S. 129]. Dies wird in der vorliegenden Arbeit jedoch nicht aufgegriffen, vgl. Def. 5, S. 35.

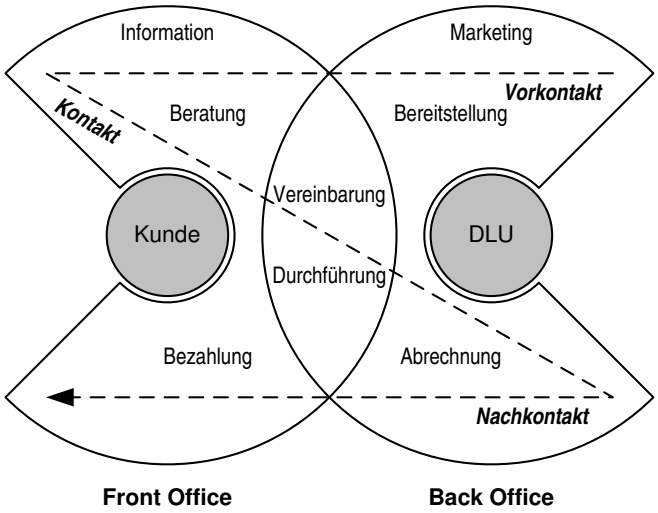

Abbildung 3.6.: Phasen des Dienstleistungslebenszyklus (nach [MBK+01, S. 129])

Im Back Office können Anwendungssysteme das Marketing z. B. durch Customer Relationship Management unterstützen.

Nach Abschluss der vertraglichen Vereinbarungen beginnt in der Kontaktphase die Durchführung des Kernleistungsprozesses. Während der Kunde hierbei z. B. durch Transaktions- oder Agentensysteme eingebunden wird, kommen dazu im Back Office u. a. spezifische WfMS, WSS und DMS zum Einsatz. Dies ist u. a. davon abhängig, ob die Dienstleistung nur im Front bzw. Back Office oder in beiden durchgeführt wird. Die spezifischen Anwendungssysteme zur fachlichen Unterstützung können dann je nach Typ der Dienstleistung sehr unterschiedlich ausfallen. Beispiele dafür sind Patientenmanagementsysteme aus dem medizinischen, Kreditprüfungssysteme aus dem finanziellen oder Routenplanungssysteme aus dem logistischen Bereich.

Schließlich ist in der Nachkontaktphase die Abrechnung und Bezahlung von Dienstleistungen zu unterstützen. Da die Bewertung von Dienstleistungen generell ein betriebswirtschaftlich ungelöstes Problem darstellt[32], wird hier meist auf pauschale Abrechnungsmethoden nach Festpreis (produktbezogener Ansatz) oder Aufwand (prozessbezogener Ansatz) ausgewichen. Zur Zahlung können dann elektronische Zahlungsmechanismen nach dem Vorbild des E-Commerce genutzt werden [Mer02, S. 459 ff.].

[32] Vgl. 2.2.2.1.

3.3. Ausgewählte Instrumente der IV-Infrastruktur im VDU

Wie in der vorangegangenen Sektion gezeigt wurde, stellen VDU umfassende Ansprüche an die zwischenbetriebliche Integration auf technischer und anwendungsbezogener Ebene. Eine entsprechende Infrastruktur muss temporäre zwischenbetriebliche Kooperationen von Unternehmensnetzwerken in ihren verschiedenen Phasen wirkungsvoll unterstützen. Dies erfordert u. a. eine flexible Kommunikation und Koordination von IT-Systemen beteiligter Netzwerkunternehmen. Damit können verteilte Anwendungssysteme zum Zweck der Kollaborationen zwischen den Netzwerkunternehmen in dynamischer Weise realisiert werden. Die folgenden Sektionen stellen verschiedene Instrumente dar, die zur Lösung einer derartigen Integrationsaufgabe dienen können.

Die Auswahl der hier berücksichtigten Instrumente/Techniken basiert auf den Zielen der vorliegenden Arbeit: Es wird eine Technologie angestrebt, bei der eine Virtualisierung von Dienstleistungen durch Abbildung auf E-Services erfolgt. E-Services sollen den Dienstleistungsprozess virtuell nachvollziehen und die im Verlauf anfallenden Aufgaben verschiedener Netzwerkunternehmen eines VDU koordinieren. Es handelt sich dabei also im Sinne der vorangegangenen Ausführungen generell um eine Integrationstechnik. Da diese einen generischen Mechanismus unterstützt, ist sie in die horizontale Basisinfrastruktur von VU einzuordnen. Sie setzt dabei auf den Interoperabilitätsmechanismen einer Kommunikationsinfrastruktur auf und fällt selbst in den Bereich der Kooperationsunterstützung. Der Schwerpunkt liegt auf der *Koordination von Komponenten eines integrierten Produktionssystems für Dienstleistungen*. Hierbei soll vor allem im Bereich der Erfassung fachlich orientierter Koordinationslogik (Dienstleistungsprozess) mitsamt entsprechendem Betriebs- und Ausführungssystem ein Beitrag geleistet werden. Als Grundlage hierfür werden in dieser Sektion drei Kategorien von Instrumenten/Techniken mit folgenden Schwerpunkten betrachtet:

- Eingliederung von Integrationsmechanismen in den betrieblichen Kontext
- Harmonisierung der zwischenbetrieblichen Komponenteninteraktion
- Modellierung und Koordination zwischenbetrieblicher Prozesse

Ein fundamentaler Schritt der Integration liegt zunächst in der Einbindung und Abstimmung der Integrationsmechanismen und -methoden in/auf den betrieblichen Gesamtkontext.[33] Hierzu wurden *Enterprise-Architektur (EA) Rahmenwerke* entwickelt, die bei der Integration als Referenzmodelle zum Einsatz kommen können. In der vorliegenden Arbeit wird ein solches Rahmenwerk dazu verwendet, die Konzepte und Techniken virt. Dienstleistungen innerhalb einer kompletten Integrationssicht abzugrenzen und einzuordnen. Dazu führt Sektion 3.3.1 EA-Grundlagen und ein konkretes Rahmenwerk – das *GERAM Enterprise Architecture Framework* – ein.

[33] Vgl. Vorgehensmodell der Integration 3.2.1.2.

Bei Voraussetzung des Internets als Basis zwischenbetrieblicher Vernetzung ist im VU dann vor allem eine systemtechnische Basis zur Realisierung VU-weit verteilter Anwendungssysteme zu etablieren. Dies beinhaltet Abstraktionen und Mechanismen zur Interaktion heterogener Systemkomponenten. Das dafür verwendete Paradigma[34] sollte nicht nur in Hinblick auf die Kooperationsunterstützungsmethoden, sondern auch die Anwendungsdomäne gewählt werden. In diesem Sinne sollen Anwendungsfunktionen im Zuge der Dienstleistungsproduktion als (Web) Services abstrahiert werden, die in Sektion 3.4 ausführlich dargestellt werden. Zunächst werden hier in Sektion 3.3.2 einige grundlegende Konzepte von *Middleware* eingeführt und besonders in Bezug auf Systemintegration im inner- und zwischenbetrieblichen Kontext (A2Ai/B2Bi) untersucht.

Die Perspektive zwischenbetrieblicher Integration wird schließlich eingeengt, indem der Fokus auf die für virt. Dienstleistungen maßgebliche Prozessintegration schwenkt. Dies geschieht in Hinblick auf eine Koordinationsebene für Dienstleistungsproduktion mitsamt Modellierung und Durchsetzung der Koordinationslogik von Dienstleistungsprozessen. Dazu werden in Sektion 3.3.3 zwischenbetriebliche WfMS (kurz *Inter-Enterprise Workflow-Systeme*) untersucht. Neben einer kurzen Einführung genereller WfMS werden hier verschiedene verwandte Forschungsprojekte, besonders auch im Zusammenhang mit VU, beschrieben.

3.3.1. Enterprise Architektur Frameworks als Integrationsrahmen

Die Fragestellung der vorliegenden Arbeit kann allgemein gesehen in das interdisziplinäre Gebiet der Enterprise-Architektur (EA) eingebettet werden. Deren ganzheitlich-technokratische Sicht auf Unternehmen und Unternehmensführung soll hier grundsätzlich übernommen werden. Jedoch ist es nicht das Ziel dieser Arbeit ein ganzheitliches EA-Konzept im Sinne eines vollständigen EA-Rahmenwerks (wie z. B. GRAI, PERA, C4ISR, CIMOSA, ZACHMAN oder ARIS; vgl. [Nor03]) zu schaffen. Vielmehr sollen die spezifischen Zusammenhänge der Produktion im VDU und der integrativen Potenziale von SOC-Techniken zur Automation der operativen Phase im Mittelpunkt stehen. In diesem Sinne entspricht das daraus resultierende (FRESCO-)Rahmenwerk einer spezifischen (technisch geprägten) Facette einer ganzheitlichen EA-Konzeption.

Der Zweck dieses Abschnitts liegt in der (rudimentären) Einführung des EA-Gedankens und der Abgrenzung der im weiteren Verlauf fokussierten Teilbereiche. Hierzu wird das GERAM Enterprise Architecture Framework herangezogen: Generalised Enterprise Reference Architecture And Methodology (GERAM) ist ein *generisches* Rahmenwerk, das in abstrakter Weise die wesentlichen Komponenten von *konkreten* Rahmenwerken beschreibt. GERAM eignet sich hierfür besonders, da es als Obermenge vielfältiger Konzepte gelten kann (vgl. [Nor03]) und somit ein umfassendes und konsolidiertes Bild liefert.

[34] Z. B. Software-Objekte, -Komponenten, -Agenten, -Services etc. (vgl. 3.2.2.2).

3.3.1.1. Enterprise Architecture, Engineering und Integration

Enterprise-Architektur im weiteren Sinn beschäftigt sich mit einem technokratisch geprägten Gesamtbild von Unternehmen und mit der ingenieurmäßigen Konstruktion, Operation (Betrieb) und Destruktion (Auflösung) dieser Unternehmen [BNS03, S. 1 ff.]. Ein Unternehmen wird dabei in Form eines sozio-technischen Gesamtsystems behandelt. Dieses ist, ganz wie ein Produkt, als spezifische Lösung zu konzipieren, zu definieren, zu entwerfen und zu fertigen. Ebenso durchläuft auch ein Unternehmen einen Lebenszyklus, der von stetiger Veränderung und zyklischer Erneuerung („Relaunches“) geprägt ist. Eine derartige Sicht auf das Unternehmen erfordert eine fundierte Vorgehensweise im Sinne einer Ingenieursdisziplin, was dementsprechend als *Enterprise Engineering* bezeichnet wird. Dem Unternehmensingenieur müssen dabei vielfältige Werkzeuge und Methoden zur Verfügung stehen, die möglichst auf formalen Theorien basieren sollten. Im Allgemeinen wird angenommen, dass eine derartige Disziplin sich dann in generischer Weise auf neue oder bestehende Unternehmen beliebiger Branchen anwenden lässt (so wie auch beim Produktdesign).

Das methodische Sortiment des Enterprise Engineering wird durch *EA-Rahmenwerke* bzw. *-Frameworks*[35] organisiert und systematisiert. Frameworks enthalten dabei u. a. Referenzarchitekturen, Vorgehensmodelle, Methoden und (Software-)Werkzeuge, die auf Basis wohldefinierter inhaltlicher Konzepte systematisiert werden. Eine wesentliche Funktion von Frameworks besteht darin, Unternehmen in Bezug auf ihren *Lebenszyklus* zu strukturieren. Dazu wird in der Regel eine Referenzarchitektur *(Enterprise Reference Architecture)* bereitgestellt. Diese gibt den (meist zyklischen) Verlauf des Enterprise Engineering vor, der mit der Identifikation des Unternehmens beginnt und über Konzeptentwurf, Architekturentwurf, Detailentwurf, Implementierung zur Operation und schließlich zur Stilllegung des Unternehmens führt.[36] Dabei wird das Unternehmen in jeder Phase durch *Unternehmensmodelle* beschrieben, für die das Framework entsprechende Modellierungssprachen *(Enterprise Modelling Languages)* und Software-Werkzeuge bereitstellt. Zu deren Verwendung wird dann ein generisches Vorgangsmodell spezifiziert. Diese wird ebenfalls durch Software-Werkzeuge unterstützt und ist an die Situation konkreter Unternehmen anzupassen. Schließlich sollte ein Framework zur besseren Nutzbarkeit wiederverwendbare (Teil-) Modelle und Bausteine unterstützen und vorgeben.

Wie bereits angesprochen, sind Wandel und Erneuerung des Unternehmens ein wesentlicher Aspekt des Enterprise Engineering. Während die Struktur des Unter-

[35] Es ist zu beachten, dass Enterprise-Architektur *keine* spezifische Form der Software-Architektur darstellt und das EA-Frameworks daher nicht mit Software-technischen Frameworks zu verwechseln sind. In Vorgriff auf die weitere Darstellung sei aber gesagt, dass die Enterprise-Architektur durchaus Software-Architekturen und -Frameworks beinhalten kann und diese dann mit der Geschäfts- und Systemarchitektur des Unternehmens in einen gemeinsamen Kontext stellt.

[36] Abb. 3.7 zeigt solche Lebenszyklen für verschiedene Organisationseinheiten/Elemente im VU.

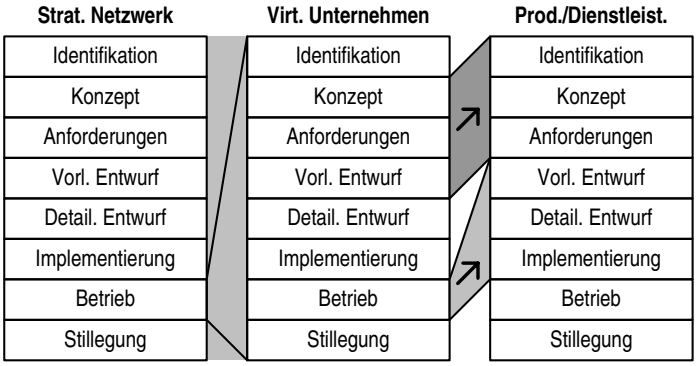

Abbildung 3.7.: EA-Lebenshistorie von VU (nach [BNS03, S. 11]

nehmens (Lebenszyklus) durch die EA beschrieben wird, kommt der dynamische Aspekt als Historie *(Life History)* dazu. Historie beschreibt, wie die Phasen des Lebenszyklusses zeitlich angeordnet sind. Dabei besitzen verschiedene Komponenten eines Unternehmens eigene Lebenszyklen, deren Phasen unter Umständen ineinander verschachtelt sind. Im Falle von VU vom Typ 3 wird z. B. zunächst ein strategisches Netzwerk gebildet, wobei die Lebenszyklusphasen bis zur Operation sequenziell aufeinanderfolgen. Im Rahmen seiner operativen Phase wird das Netzwerk dann verschiedene VU zur Produktion spezifischer Produkte und Dienstleistungen ins Leben rufen. Deren Lebenszyklen verlaufen parallel zur Operation des Netzwerks und ggf. zueinander. Die Konzeption (von der Konzeption bis zum vorläufigen Entwurf) des VU durch den Netzwerk-Broker beeinflusst schon direkt die Konzeptphasen des individuellen Produktes, das in der operativen Phase des VU produziert wird (vgl. Abb. 3.7).

Da Unternehmen sich nach heutiger Sicht im Wesentlichen durch ihre Geschäftsprozesse definieren, liegt ein wesentlicher Aspekt der EA in deren Erfassung. Wie schon weiter vorne gezeigt wurde, setzen sich solche Geschäftsprozesse aus vielfältigen operativen und Managementprozessen zusammen, die mehr oder weniger direkt zur Wertschöpfung beitragen. Eine EA muss alle diese Teilprozesse erfassen und ihre Verknüpfung im Sinne der Geschäftsprozesse durch geeignete Informationsstrukturen beschreiben (auch als *Informationsarchitektur* bezeichnet). Daneben muss sie auch eine Beschreibung von Ressourcen (Menschen und Maschinen) als Akteure in den Prozessen und ihrer Anordnung in Zeit und Raum liefern. Die Ableitung von IS- und IT-Architekturen zur Unterstützung der informationellen Prozesse durch integrierten IV ist dabei nur ein (wenn auch sehr wichtiges) partielles Ziel. Der Fokus liegt auf einer Unterstützung der ganzheitlichen *Enterprise Integration*. Diese zielt nicht nur auf die Unterstützung, sondern auf die Optimierung (d. h. Reorganisation) von Geschäftsprozessen und betrachtet dazu die Gesamtheit aller materiellen und informationellen Prozesse. Als Mittel zur Enterprise Integration erfasst und beschreibt die EA

Prozesse, Strategien, Organisationsstrukturen, Ressourcen, Ziele und Bedingungen. Sie erfasst damit die Geschäftsprozessanforderungen, identifiziert Lösungsoptionen, präsentiert alternative Entwürfe und liefert schließlich Implementierungswege. Das Vorgehensmodell der Enterprise Integration beruht somit auf der Lebenszyklusarchitektur und den Modellierungssprachen des EA-Framework. Entsprechende Werkzeuge müssen die Flüsse und Transformationen von Materialien und Informationen sowie die Organisationsstrukturen menschlicher Akteure in den Prozessen berücksichtigen, sowohl in als auch zwischen (ggf. virt.) Unternehmen. Solche Frameworks erlauben dann im Gegensatz zu den frühen Lösungen integrierter Informationsverarbeitung (z. B. im Bereich des Computer Integrated Manufacturing (CIM)) auch eine produktmäßige Entwicklung hoch integrierter betrieblicher Informationssysteme. Da deren Konstruktion in das ganzheitliche Vorgehensmodell der Unternehmensintegration eingebettet ist, gehen sie nicht nur mit Lebenszyklen des Unternehmens und seinen Komponenten konform, sondern zeigen sich auch bezüglich ihrer Historie flexibel.

3.3.1.2. Das GERAM Enterprise Architecture Framework

Die ersten Ansätze der oben beschriebenen EA-Sicht gehen auf frühe Entwicklungen um 1980-1990 zurück. Als Meilenstein gilt u. a. das ESPRIT-Projekt AMICE, das zur *Open Systems Architecture for CIM (CIMOSA)* führte [AMI93]. Parallel sind in der Vergangenheit eine Reihe weiterer EA-Frameworks entwickelt worden. Dazu kann auch das verbreitete *Zachman-Framework* gezählt werden [Zac87]. Weitere Beispiele umfassen etwa die GRAI-, PERA-, C4ISR-, und ARIS-Frameworks (vgl. z. B. [Nor03]). Aus der Vielfalt der Rahmenwerke entstand der Wunsch nach Vergleich und Systematisierung der Ansätze. Der direkte Vergleich zeigte sich jedoch auf Grund unterschiedlicher Sichtweisen und spezifischer Besonderheiten schwierig. Aus diesem Grund wurde im Rahmen einer kombinierten IFIP-IFAC Task Force das generische GERAM Enterprise Architecture Framework entwickelt. In GERAM wurden die Ansätze (Konzepte, Methoden, Werkzeuge etc.) der spezifischen Frameworks generalisiert und systematisch zusammengefasst (vgl. Abb. 3.8). Dadurch bildet GERAM einen abstrakt-generischen Rahmen, der einen Satz grundlegender Elemente eines jeden EA-Framework exakt definiert und einordnet[37]. Dies kann zum Vergleich verschiedener konkreter Frameworks verwendet werden, indem deren spezifische Ansätze jeweils relativ zu denen von GERAM eingeordnet werden [Nor03]. GERAM besitzt aber auch für sich allein großen Nutzen als Orientierungshilfe im umfassenden Sortiment von Konzepten, Modellen, Methoden und Werkzeugen für die Konstruktion und den Betrieb von (z. B. virt.) Unternehmen über ihre gesamte Historie.

GERAM enthält Beschreibungen aller Elemente, die für Integration und Engineering von Unternehmen als notwendig oder wünschenswert erachtet werden. Es werden jedoch keine konkreten Werkzeuge und Methoden vorgegeben, sondern nur

[37] Der normative Charakter von GERAM wurde durch die Übernahme in verschiedene Standards (z. B. ISO15704:2000) untermauert.

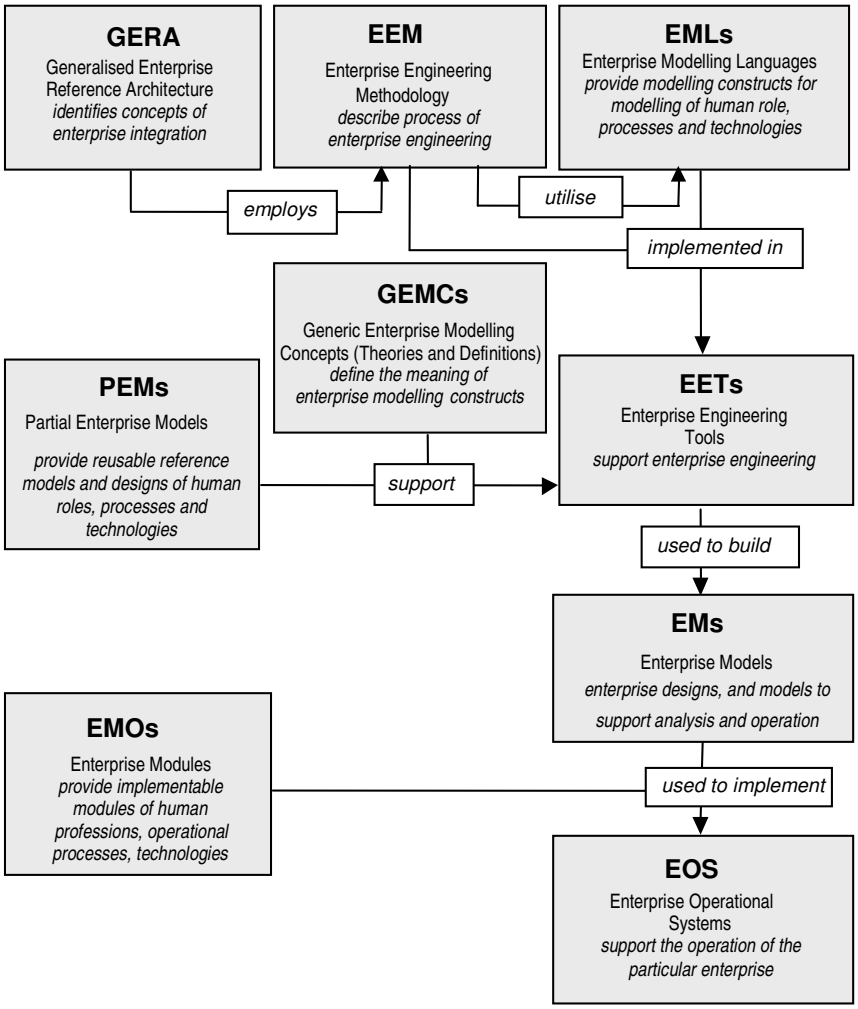

Abbildung 3.8.: Komponenten des GERAM-Framework [IFI03, S. 24]

die von solchen zu erfüllenden Kriterien. Grundsätzlich wird die Bedeutung von Unternehmensmodellen über verschiedene Phasen, in verschiedenen mehr oder weniger formalen Varianten und mit verschiedenen Modellierungssprachen hervorgehoben. Die einzelnen Komponenten von GERAM sind in Abbildung 3.8 dargestellt. Sie werden im Folgenden (angelehnt an den offiziellen Standard [IFI03]) kurz skizziert.

Die *Generalised Enterprise Reference Architecture (GERA)* beschreibt die wesentlichen Konzepte, die beim Enterprise Engineering genutzt werden. Dies beinhaltet zunächst Konzepte für Menschen in Unternehmen. Zum einen in Bezug auf deren organisatorische und operationale Rollen und zum anderen in Bezug auf deren Unterstützung beim Enterprise Engineering. Des Weiteren sind prozessorientierte Konzepte zur Beschreibung von Geschäftsprozessen enthalten, die insbesondere Lebenszyklus und Historie von Entitäten umfassen. Schließlich werden technische Konzepte berück-

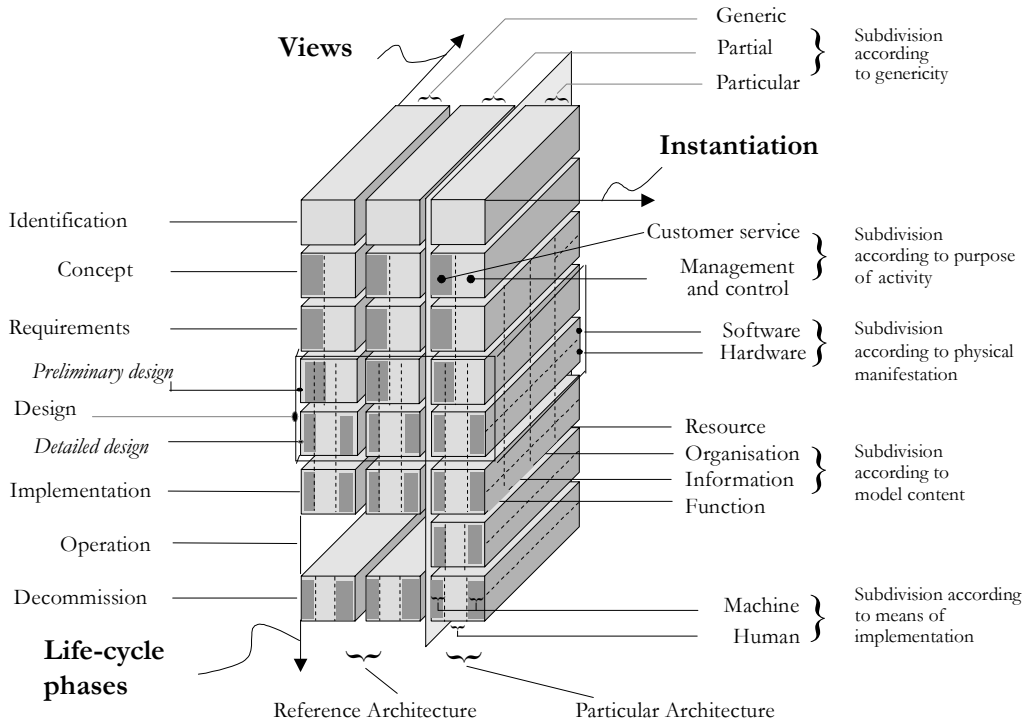

Abbildung 3.9.: GERA Modelling Framework [IFI03, S. 43]

sichtigt, die eine Beschreibung von Techniken zur Unterstützung des Engineering und Betriebs von Unternehmen (bzw. deren Geschäftsprozessen) erlauben. Zu GERA gehört insbesondere auch ein Modellierungsrahmen, der die Modelle verschiedener Lebenszyklusphasen, Sichten und Metaebenen strukturiert. Abbildung 3.9 zeigt den Modellierungsrahmen mit seinen Dimensionen und deren Ausprägungen.

GERAM unterscheidet weiter zwischen Enterprise Engineering Methodologien und Modellen. Die *Enterprise Engineering Methodology (EEM)* beschreibt in abstrakter Weise das Vorgehen bei Enterprise Engineering und Integration. Dieses orientiert sich am Lebenszyklusmodell der Architektur und kann im konkreten Fall in Form eines Prozessmodells oder einer strukturierten Prozedur ausdifferenziert werden. Des Weiteren werden der menschliche Faktor und das Projektmanagement berücksichtigt. Im Rahmen der Methodologie kommen *Enterprise Modelling Languages (EMLs)* zum Einsatz. Diese dienen nach Vorgabe des GERA Modellierungsrahmens zur Beschreibung von Entitäten in Bezug auf Inhalte, Strukturen und Verhalten. EMLs sollen sowohl den menschlichen Part im Geschäftsbetrieb als auch den Part der Geschäftsprozesse und der sie unterstützenden Technologie ausdrücken können. Die Semantik dieser Sprachen wird explizit durch *Generalised Enterprise Modelling Conceptss (GEMC)* beschrieben. Zu diesem Zweck können etwa Glossare, Metamodelle oder Ontologien zum Einsatz kommen.

Bei Verwendung von EMLs im Rahmen der EEM werden menschliche Akteure durch *Enterprise Engineering Tools (EET)* unterstützt. Dabei sollte auch die Möglichkeit bestehen, auf *Partial Enterprise Models (PEM)* zurückzugreifen. Dies sind Referenzmodelle oder wiederverwendbare Modellkomponenten, z. B. für Rollen, Prozesse oder Technologien. Am Ende entstehen konkrete *Enterprise Models (EM)*, die den Geschäftsbetrieb inkl. Produktionsaufgaben, deren Organisation und Management sowie den Informationssystemen repräsentieren. Die Modelle dienen u. a. als Grundlage der Implementierung von *Enterprise Operative Systems (EOS)*. EM werden aber auch zur Evaluation organisatorischer und operationaler Alternativen verwendet (z. B. durch Simulation). Bei der Implementierung wird eine möglichst direkte Überführung der Modelle (z. B. durch Codegenerierung) angestrebt. Hierbei können (und sollen) *Enterprise Modules (EMO)* helfen. Dies sind vorhandene Komponenten (z. B. Human Resources, IT-Services etc.), die als Teil des EOS dienen können.

Besonders die Rolle von IT-Services zur Unterstützung des Enterprise Engineering und des operativen Betriebs wird hervorgehoben: zum einen in Bezug auf Portabilität und Interoperabilität von Modellen durch eine unternehmensübergreifend integrierte Infrastruktur; zum anderen in Bezug auf modellgetriebene Unterstützung des operationalen Betriebs (Steuerung, Kontrolle) durch Echtzeit-Zugriff auf die Unternehmensumgebung. Dies wird in GERA durch ein (rudimentäres) Referenzmodell für *Enterprise Model Execution and Integration Services (EMEIS)* untermauert (vgl. Abb. 3.10). EMEIS beschreibt eine Infrastruktur integrativer Services, die den (unternehmensübergreifenden) Austausch von Modellen zwischen allen Phasen und Akteuren ermöglicht *(Shared Services)* und sowohl die Entwicklung *(Model Development Services)* als auch den Betrieb *(Model Execution Services)* von Entitäten unterstützt.

3.3.2. Middleware als systemtechnische Interaktionsplattform

Die Vernetzung von VDU kann heute durch das Internet als gegeben angesehen werden. Dies stellt jedoch lediglich eine Basis zwischenbetrieblicher Kommunikation dar, d. h., dass die Anwendungssysteme verschiedener Netzwerkunternehmen prinzipiell Daten austauschen können. Sollen diese jedoch im Sinne eines gemeinsamen Anwendungssystems interagieren, ist es zunächst notwendig, die Autonomie und Heterogenität der Systemkomponenten zu überwinden; kurz gefasst: es muss *Interoperabilität* hergestellt werden. Interoperabilität bezeichnet die Fähigkeit von Anwendungssystemen zur verteilten Zusammenarbeit auf Basis einer Kommunikationsinfrastruktur [Wen97, S. 907]. Dies bedingt geeignete Schnittstellen, Protokolle, Formate und Verfahren.

Interoperabilität kann durch Etablierung offener verteilter Systemplattformen erreicht werden, die gemeinsame Abstraktionen und Mechanismen zur Überbrückung der Unterschiede bereitstellen. Solche Systemplattformen werden im Allgemeinen als *Middleware* bezeichnet. Entsprechende Middleware-Abstraktionen erweitern das Programmiermodell von AS um Mittel zur wechselseitigen Interaktion. Middleware-

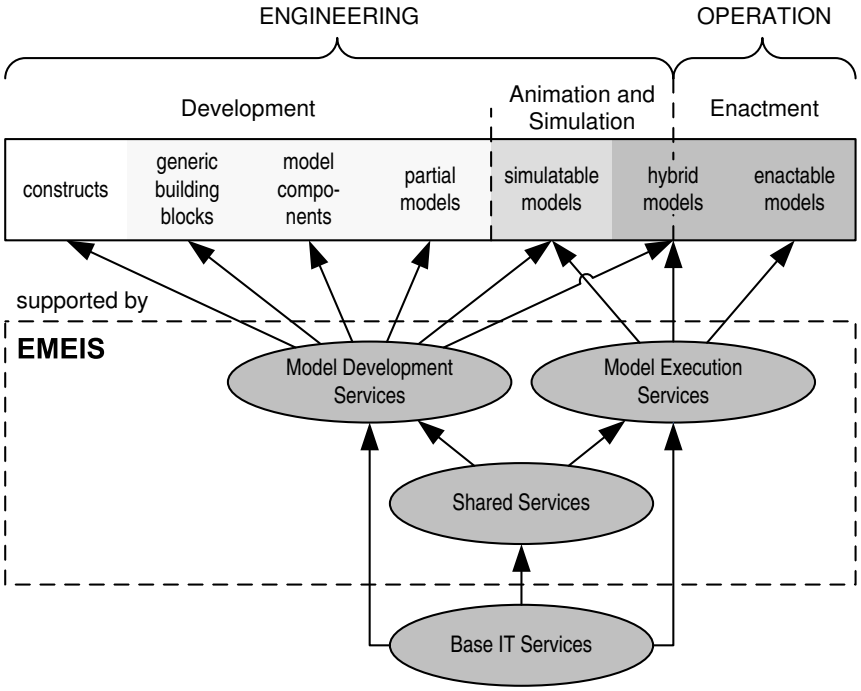

Abbildung 3.10.: GERA EMEIS (nach [IFI03, S. 42])

Mechanismen setzen die Abstraktionen in und zwischen verschiedenen Systemumgebungen um.

Moderne Middleware geht mittlerweile über systemtechnische Interoperabilitätsmechanismen hinaus. Frühe Middleware wurde vor allem für die Top-Down-Konstruktion verteilter AS in heterogenen Systemumgebungen genutzt. Heute wird immer mehr die Bottom-Up-Integration von Altsystemen gefordert. In diesem Sinne müssen Middleware-Mechanismen nicht nur die Kapselung von Altsystemen als Komponenten unterstützen, um dadurch atomare Interaktionen zu ermöglichen. Sie müssen vor allem auch die Logik komplexer Interaktionsabläufe erfassen und eine übergeordnete Koordinationsebene etablieren. Infolgedessen wächst Middleware zur umfangreichen *Integrationsplattform* an. Neben der innerbetrieblichen Systemintegration wird zum Teil auch die Integration von AS-Komponenten verschiedener Unternehmen berücksichtigt, was die Komplexität weiter erhöht und die Perspektive in Richtung (geschäftlicher) Anwendungsebene verschiebt.

Diese Entwicklung führt allerdings dazu, dass Middleware kaum noch als einzelne Technik betrachtet werden kann. Viel mehr umfassen die heutigen A2Ai- und B2Bi-Plattformen eine große Bandbreite sehr verschiedener Techniken wie Message Broker und WfMS. Die Basis wird jedoch nach wie vor durch die grundlegenden Interoperabilitätsmechanismen der „klassischen“ Middleware – allen voran RPC-Mechanismen und TP-Monitore – gebildet. Im Folgenden wird daher zunächst das klassische Middleware

Konzept verteilter Systemen eingeführt und dessen Evolution im innerbetrieblichen Kontext nachvollzogen. Im Anschluss wird die Weiterentwicklung von Middleware zur inner- und zwischenbetrieblichen Systemintegration erläutert.

3.3.2.1. Middleware als Software-Konzept für vert. Systeme

Die Entstehung von Middleware ist direkt mit der „historischen" Entwicklung verteilter Anwendungssysteme[38] verbunden. Als Triebfeder dieser Evolution wird heute hauptsächlich die rasante Verbreitung verteilter Systemlandschaften genannt, die ca. seit den 80er Jahren des letzten Jahrhunderts durch Verfügbarkeit leistungsstarker Einzelplatzrechner und deren zunehmende Vernetzung begünstigt wurde.[39] Ein *vert. System*[40] kann wie folgt definiert werden:

Definition 12 (vert. System (VS)) *Ein VS ist ein System, in dem sich Hardware- oder Software-Komponenten auf vernetzten Computern befinden und nur über den Austausch von Nachrichten kommunizieren und ihre Aktionen koordinieren.*

VS eröffnen erhebliche Potenziale zur Ressourcenteilung (z. B. Hardware, Daten, Funktionen und Prozesse). Die Definition impliziert jedoch auch eine erhebliche Komplexität: im Wesentlichen durch die Nebenläufigkeit der Systemkomponenten, das Fehlen eines gemeinsamen Zustands (z. B. Systemzeit) sowie die Möglichkeit unabhängiger Fehler und Ausfälle. Insgesamt müssen daher in vert. Systemen die Überbrückung von *Heterogenität*[41], die Gewährleistung von *Offenheit*[42], *Sicherheit* und *Skalierbarkeit*, die *Fehlerverarbeitung*, die Absicherung von *Nebenläufigkeit* und die Realisierung von *Transparenz*[43] berücksichtigt werden [CDK02, S. 34 ff.].

In Bezug auf die grundsätzlichen Rollen und Beziehungen der Komponenten eines VS gibt die so genannte *Cl/Srv-Architektur* eine fundamentale Struktur vor, auf die sich die meisten höherwertigen Paradigmen hinunterbrechen lassen [CDK02, S. 53 ff.].[44] Hierbei werden die Rollen von Client und Server unterschieden. Deren

[38] In diesem Kontext wird in der Literatur oft von *Informationssystemen* gesprochen. Dies mag auf den entsprechenden angelsächsischen Begriff zurückzuführen sein: Dieser steht dort für die Disziplin der Wirtschaftsinformatik bzw. entsprechender Anwendungssysteme.

[39] Vgl. z. B. [Mas05, S. 43 f.].

[40] Grundlagen vert. Systeme sind extensiv beschrieben worden, vgl. z. B. [CDK02], [TS03], [Ham05]. Die folgenden Ausführungen orientieren sich an Coulouris et al. [CDK02].

[41] D. h. Diversität oder Differenzen von Netzen, Betriebssystemen, Hardware und Programmiersprachen.

[42] D. h. Erweiterbarkeit durch Offenlegung relevanter Schnittstellen.

[43] D. h. Verbergen problematischer Aspekte der Verteilung wie Zugriffs-, Orts-, Nebenläufigkeits-, Fehler-, Mobilitäts-, Leistungs- und Skalierungstransparenz, vgl. z. B. [IJ95].

[44] Für *Cl/Srv-Architekturen* existieren verschiedene Variationen, z. B. in Bezug auf mobilen Code, Agenten und mobile Geräte [CDK02, S. 56 ff.]. Eine Alternative bieten *Peer-to-Peer-Architekturen*, bei denen die Prozesse gleichrangig (und ohne festes Call-Reply-Schema) interagieren [CDK02, S. 55 f.]. In der vorliegenden Arbeit werden aber im Folgenden ausschließlich SOC-Techniken betrachtet; diese lassen sich grundsätzlich als Cl/Srv-Architekturen einordnen.

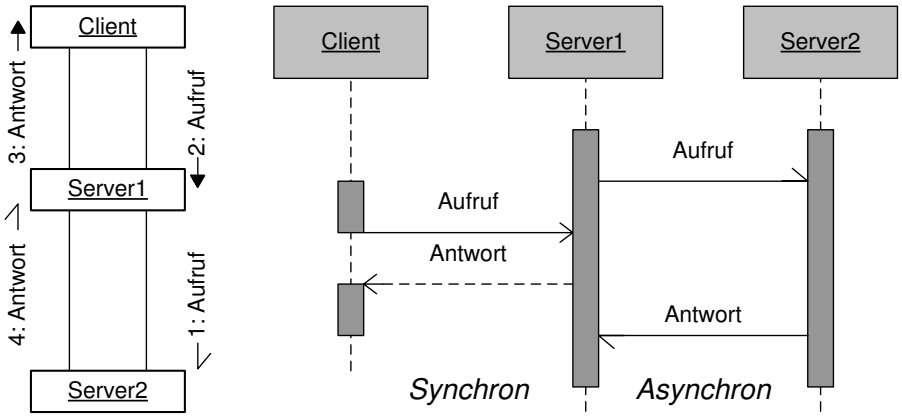

Abbildung 3.11.: UML-Kollaborations- und -Sequenzdiagramme mit Kommunikationsvarianten eines Cl/Srv-Systems

Interaktion läuft in zwei sequenziellen Schritten bestehend aus Aufruf (Call) eines Dienstes und Antwort (Reply) des Ergebnisses ab. Die Server-Rolle wird in der Regel von einem langlebigen Prozess übernommen, die Client-Rolle von einem kurzlebigen, wobei diese auf verschiedene Computer verteilt sein können. Ein Prozess kann hier auch gleichzeitig Client und Server-Rollen übernehmen (vgl. Abb 3.11 links).

Bzgl. der Kommunikation von Komponenten werden in der Regel *synchrone* und *asynchrone* Varianten unterschieden. Der Unterschied besteht darin, dass der anfragende Prozess bei synchroner Kommunikation bis zum Erhalt der Antwort mit seiner Verarbeitung aussetzt, während er bei asynchroner Kommunikation seine Verarbeitung fortsetzt (vgl. Abb 3.11 rechts).[45] Synchrone Kommunikation führt zu einfach nachzuvollziehbaren Strukturen mit festen Bindungen und genauerer Abstimmung der Elemente/Prozesse im VS, ist jedoch mit Koordinationsaufwand verbunden und oft unnötig. Asynchrone Kommunikation bietet hingegen eine lose, aber komplexe/indirekte Beziehung von Elementen/Prozessen. Diese Form gibt oft – gerade in zwischenbetrieblichen AS mit zum Teil erheblicher Latenz – die realen Verhältnisse besser wieder.

Zur Realisierung vert. Systeme unter Berücksichtigung der genannten Aspekte sind verschiedene Hardware- und Software-Konzepte entwickelt worden. Hardwarekonzepte betreffen im Wesentlichen Abgrenzung und Vernetzung autonomer Rechnerknoten (von der Multiprozessormaschine bis zu Workstations im LAN). Software-Konzepte betreffen im Wesentlichen die Erweiterung der Systemsoftware dieser Rechner zur Unterstützung des VS. Funktionen zur Nutzung des VS werden dabei über möglichst offene standardisierte Schnittstellen angeboten und mithilfe von Protokollen zur Kommunikation korrespondierender Knoten implementiert.

[45] Synchrone Aufrufe werden daher auch als *blockierend* und asynchrone als *nicht-blockierend* bezeichnet.

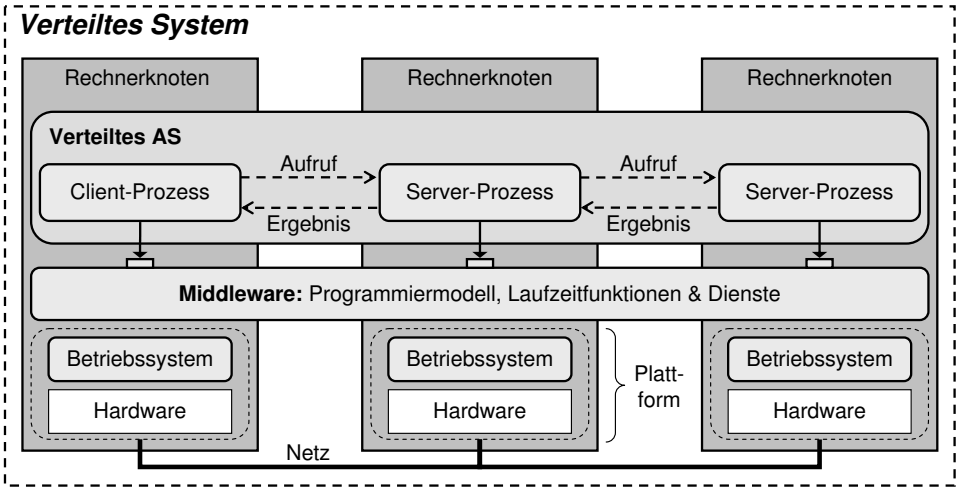

Abbildung 3.12.: Verteiltes System mit Middleware und Cl/Srv-basiertem AS

Die Systemsoftware basiert grundsätzlich auf einem *Betriebssystem* zur Ressourcenverwaltung.[46] Betriebssysteme können darüber hinaus die Aspekte eines VS berücksichtigen, wobei zwei Varianten unterschieden werden [TS03, S. 39 ff.]:

- Ein einheitliches *verteiltes Betriebssystem* wird auf verschiedenen homogenen Rechnerknoten installiert und lässt sie als monolithisches System erscheinen.
- Verschiedene *Netzwerkbetriebssysteme* werden auf heterogenen Rechnerknoten installiert und interagieren über offene Standards, um bei eingeschränkter Transparenz Ressourcen zu teilen.

Verteilte Betriebssysteme sind auf Grund der Heterogenität im VDU grundsätzlich auszuschließen. Die Kommunikationsfähigkeiten in Netzwerkbetriebssystemen reichen jedoch für die geforderten Kooperationsmechanismen im VDU nicht aus, da transparente Programmiermodelle zur Kommunikation von Anwendungssystemen fehlen. Bessere Voraussetzungen ergeben sich durch ein weiteres Software-Konzept, das auf Grund seiner Lage zwischen Betriebs- und Anwendungssystem als *Middleware* bezeichnet wird.

Genauer wird mit dem Begriff der Middleware eine Software-Schicht bezeichnet, die auf heterogenen Plattformen aufsetzt und auf jedem Rechnerknoten verfügbar ist [CDK02, S. 50 f.] (vgl. Abb 3.12). Die Hauptaufgabe dieser Schicht besteht darin, die Heterogenität der Plattformen zu verbergen. Darüber hinaus können verschiedene weitere Formen der Transparenz unterstützt werden, um das vert. System für Anwendungssysteme mehr oder weniger homogen erscheinen zu lassen.

[46] Rechner-Hardware und Betriebssystem bilden zusammen eine so genannte *(System-)Plattform* [CDK02, S. 50 f.] für verteilte System- und Anwendungssoftware.

Die Aspekte von Middleware lassen sich in orthogonalen Dimensionen strukturieren. Hammerschall unterscheidet zwei Klassen homogener Funktionalität, die eine Middleware für die Entwicklung verteilter AS bietet: zum einen die *Kommunikation* zwischen Systemkomponenten und zum anderen Mechanismen und Dienste zur Unterstützung generischer *Anwendungsfunktionen*. In diesem Sinne spricht sie von *kommunikations-* und *anwendungsorientierter Middleware* [Ham05, S. 19].

Alonso et al. treffen hingegen eine Unterscheidung in Hinblick auf die Erscheinugsformen der Middleware [ACK+04, S. 30 ff.]. Zum einen tritt sie als *Programmiermodell* in Erscheinung, das gängige Programmiersprachen um zusätzliche Abstraktionen erweitert. Zum anderen bildet sie eine *Infrastruktur* (oder Plattform) mit vielfältigen generischen Entwurfs- und Laufzeitmechanismen sowie offenen verteilten Diensten zur Implementierung des Programmiermodells.

3.3.2.2. Konventionelle Middleware

Middleware umfasst also verschiedene Mechanismen bzw. Dienste und deren programmiersprachliche Abstraktionen zur Realisierung und Anwendung von generischen Kommunikations- und Anwendungsfunktionen. Diese lassen sich nun am besten im Licht der historischen Entwicklung verteilter AS konkreter betrachten. Um diese Entwicklung nachzuvollziehen, muss zunächst noch etwas über die grundsätzliche Architektur von AS gesagt werden. In der Regel wird heute postuliert, dass der Entwurf komplexer Anwendungssysteme in drei funktionale Bereiche[47] zu trennen ist (vgl. Abb. 3.13 links):[48]

- *Präsentation:* Die Präsentationsebene dient der Realisierung von Anwender- bzw. Client-Schnittstellen. Darunter fallen etwa Benutzer- oder Software-Schnittstellen.
- *Anwendungslogik:* Die Ebene der Anwendungslogik dient der Realisierung der (geschäftlichen) Anwendungsfunktionen.
- *Ressourcenmanagement:* Auf Ressourcenmanagement-Ebene erfolgt der Zugriff auf Systemressourcen (z. B. DBs, Altsystem, Hardware etc.).

Die Trennung von Ebenen hat verschiedene (Software-technische) Vorteile, u. a. in Bezug auf Komplexitätsreduktion, Spezialisierung, Wiederverwendung, Wartung, Evolution etc. bei der Entwicklung von (Teil-)Systemen. Für die Realisierung des AS ist nun die Aufteilung (und damit Trennung) der Ebenen auf Knoten eines VS von entscheidender Bedeutung. Hierbei wurden im Laufe der Zeit im Wesentlichen

[47] Diese Bereiche werden in der englischsprachigen Literatur als *Layers* bezeichnet. Dessen direkte Übersetzung als „Schicht“ wird allerdings im Deutschen meist in Bezug auf die Architektur verteilter AS verwendet. In diesem Sinne wird dafür im Folgenden der Begriff der *Ebenen* verwendet.

[48] Vgl. z. B. [Ham05, S. 25], [ACK+04, S. 4 ff.].

drei Varianten durchlaufen, die durch ihre Anforderungen das Bild von Middleware-Plattformen entscheidend geprägt haben.

Zunächst wurden alle AS-Ebenen auf einem Mainframe realisiert (vgl. Abb. 3.13 Mitte oben). Als Clients dienten dabei passive Terminals ohne IV-Funktion. Diese *1-Schicht Architektur* hat unbestrittene Vorteile bei der Performanz, da die resultierenden monolithischen Systeme ohne Rücksicht auf externe Abhängigkeiten optimiert werden können. Die Systeme sind aber auch unflexibel und kommen heute in der Regel nur noch als Altlasten vor. Middleware existierte hier nicht im Sinne eines VS, sondern im Wesentlichen zur Erweiterung und Optimierung von Funktionen des Betriebssystems. Die wichtigste Kategorie bilden frühe Formen von Transaktionssystemen die auch als *TP-Monitore* bezeichnet werden.[49]

Mit der Verbreitung von Workstation, PC und LAN bildeten sich VS-Strukturen heraus, die die Verteilung von AS möglich machten. Dabei wurden verschiedene Varianten zur Verlagerung von AS-Ebenen vom Mainframe auf den Client realisiert.[50] Typisch ist die Verlagerung der Präsentationsebene auf den Client (vgl. Abb. 3.13 mittig). Die Vorteile dieser *2-Schicht-Architektur* liegen (a) in der Möglichkeit zur flexiblen Realisierung individueller Präsentationsebenen und (b) in der Möglichkeit zur spezifischen Wartung/Betreuung der Kernfunktionen.

Aus VS-Perspektive entspricht die 2-Schicht-Architektur der Cl/Srv-Architektur, die so zu Popularität gelangte. Middleware musste hierbei in erster Linie eine Kommunikationsinfrastruktur für verteilte AS-Komponenten in Client oder Server-Rolle bereitstellen. Dabei gab es anfangs im Wesentlichen zwei Typen:

- *Remote Procedure Call (RPC):* Prozedurales Programmiermodell, bei dem die Kommunikation zwischen AS-Komponenten durch den lokalen Aufruf entfernter Prozeduren abstrahiert wird. Damit einher geht eine Infrastruktur zur Transformation eines lokalen in einen entfernten Prozeduraufruf in einheitlicher und transparenter Weise.[51]

- *Object Broker (ORB):* OO-Programmiermodell, bei dem auf die Methoden und Attribute von Objekten einer entfernten AS-Komponente lokal zugegriffen wird. So eine Infrastruktur dient zur Abbildung der Objektmodelle verschiedener OO-Sprachen, zur Vermittlung entfernter Objektreferenzen und zur Transformation von Objektzugriffen in einheitlicher und transparenter Weise.[52]

[49] Das wohl bekannteste System dieser Art ist das IBM *Customer Information and Control System (CICS)* (vgl. z. B. [GR93]).

[50] Je nach Umfang der Auslagerung spricht man von *Thin-Clients* (nur Präsentation) oder *Fat-Clients* (zusätzlich Anwendungslogik und ggf. Ressourcenmanagement).

[51] Populäre Vertreter umfassen Sun-RPC (vgl. z. B. [CDK02, S. 224 ff.]) und das Distributed Computing Environment (DCE) der Open Software Foundation (OSF) (vgl. z. B. [TS03, S. 102 ff.]).

[52] Zu den verbreitetsten verteilten Objektsystemen gehören OMG CORBA, Java RMI und Microsoft DCOM (vgl. z. B. [Ham05, S. 127 ff.,S. 89 ff.,S. 190 ff.]).

Beiden Typen gemeinsam ist die Verwendung einer Interface Definition Language (IDL) zur expliziten Spezifikation von Schnittstellen sowohl zu den Diensten der verteilten AS-Komponenten als auch der Middleware selbst. Auf diese Weise wurde für verteilte AS schon früh ein Dienstbegriff geprägt und die Spezifikation von offenen Schnittstellen (Application Program Interfaces (API)) gefördert. Nachdem diese Kommunikationsinfrastruktur eine Konstruktion von Cl/Srv-Systemen ermöglicht hatte, folgte deren Unterstützung durch Erweiterung der schon auf dem Mainframe etablierten Transaktionssysteme. Entsprechende TP-Monitore erlaubten die Koordination verteilter Transaktionen auf Basis von RPC. *Objekt-Monitore* bildeten das Äquivalent für verteilte Objekte.

Vor allem innerhalb von größeren Unternehmen mit komplexen Organisationsstrukturen sammelte sich schnell eine Vielzahl von Cl/Srv-Systemen. Deren Dienste waren über offene API zugreifbar, die es möglich machten, beliebige Clients in unabhängiger Weise zu programmieren. Als Konsequenz entstand schnell der Wunsch, in einem Client verschiedene Server/Dienste zu kombinieren (vgl. Abb. 3.13 rechts oben). Dies führt allerdings dazu, dass die Logik zur Integration autonomer Server/Dienste (z. B. in Bezug auf syntaktische und semantische Unterschiede von Funktionen und Daten) innerhalb des Clients implementiert werden muss. Clients erhalten dadurch eine zusätzliche Ebene der Anwendungslogik, die durch ihre individuelle Abhängigkeit von autonomen Servern erhebliche Problemen bei der Konstruktion und Wartung mit sich bringt. Dieser Ansatz ist daher nicht sinnvoll und markiert die Grenzen der 2-Schicht-Architektur.

Die Verlagerung der Anwendungslogik auf eine eigene Schicht[53] ist vor allem auf die Defizite der 2-Schicht-Architektur bei der Integration mehrerer Server/Dienste zurückzuführen [ACK+04, S. 16]. Grundsätzlich ist aber zunächst festzustellen, dass eine dementsprechende *3-Schicht-Architektur* generell die Flexibilität sowie vor allem die Skalierbarkeit und Wartbarkeit von AS verbessert. Die Anwendungslogik ist hier nicht fest an ein spezifisches Ressourcenmanagement gebunden (vgl. Abb. 3.13 Mitte unten). Sie ist daher unabhängiger und kann besser wiederverwendet werden. Zur Laufzeit ist es möglich, die Anwendungsschicht auf kleinere Rechner zu verlagern und zur besseren Skalierung ggf. in einem Cluster zu replizieren. Dem steht bei 2-Schicht-Architekturen ein erhöhter Aufwand bei der Interaktion der nun getrennten Ebenen gegenüber. Bei homogenen AS (d. h. ohne Integration heterogener Altsysteme) sind daher Kosten und Nutzen abzuwägen.

Middleware erweitert sich in 3-Schicht-Architekturen von einer primären Kommunikationsinfrastruktur zu einer Infrastruktur für die generelle Unterstützung der Anwendungsschicht.[54] Grundsätzlich wurden dazu zunächst die schon vorhandenen

[53] Diese Schicht wird auf Grund ihrer Lage oft als *Middle-Tier* bezeichnet.

[54] Komplementär zur Ableitung des Middleware-Begriffs aus seiner Lage zwischen Betriebs- und Anwendungssystem kommt hier eine zweite Auslegung als Infrastruktur der Middle-Tier hinzu.

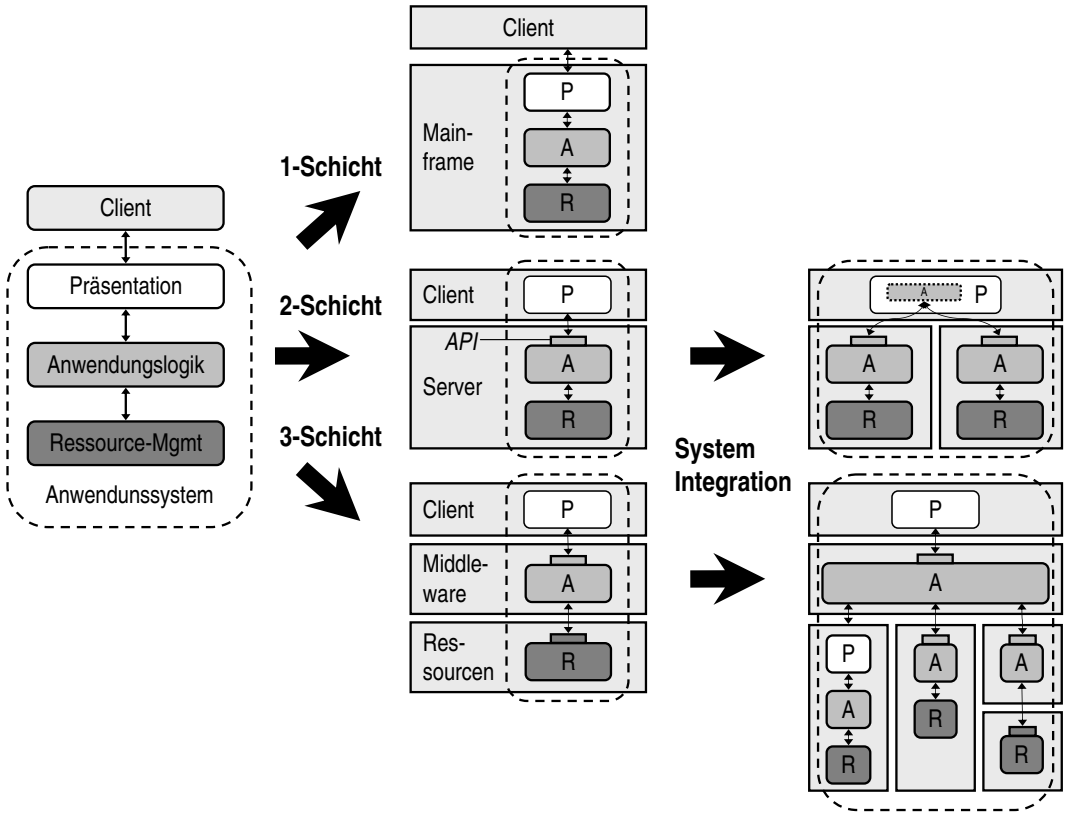

Abbildung 3.13.: Entwurfsebenen (links) und Schichtenarchitekturen für homogene (Mitte) und heterogene (rechts) verteilte Anwendungssysteme

RPC-, ORB- und TP-Techniken eingesetzt. Wegen der komplexeren Beziehungen zwischen den Systemkomponenten wurde zudem der Nutzen eines indirekten und asynchronen Nachrichtenaustauschs über Warteschlangen (so genannte Queues) erkannt. Derartige Mechanismen wurden schon von TP-Monitoren bereitgestellt und nun als separate *Message Oriented Middleware (MOM)*[55] gezielt eingesetzt. Hinzu kamen neue Mechanismen und Abstraktionen zur Unterstützung wünschenswerter Eigenschaften der Anwendungsschicht. Neben Transaktionalität gehörten dazu Sicherheit, Replikation, Protokollierung, Persistenz und viele weitere. In diesem Zusammenhang wurden auch modulare *komponentenorientierte Programmiermodelle* eingeführt. Angesichts diversifizierter innerbetrieblicher Systemlandschaften entstanden für solche Infrastrukturen umfassende Standards.[56]

[55] Vgl. z. B. [Ham05, S. 99 ff.].

[56] Ein wichtiges Beispiel für einen besonders umfassenden Standard ist CORBA, das im Laufe der Zeit von einem reinen ORB durch das CORBA Component Model (CCM) und diverse CORBA-Services für Persistenz, Transaktionalität, Sicherheit etc. erweitert wurde [OMG04].

Analog zur 2-Schicht-Variante führten 3-Schicht-Architekturen zur Etablierung offener APIs für das Ressourcenmanagement, die über die Anwendungslogik von 3-Schicht-AS angesprochen werden können. Darüber hinaus kann und wird die Anwendungslogik jedoch in vielen Fällen auch Integrationslogik zur Einbindung heterogener Altsystemen enthalten. Dies ist der betrieblichen Realität gewachsener Systemlandschaften geschuldet und eine notwendige Konsequenz des Bottom-Up Entwurfs von AS.[57] Hierbei muss die Anwendungsschicht je nach Architektur des Altsystems ggf. als Client eines Cl/Srv-Systems auftreten oder Wrapper für Mainframe-Systeme nutzen (vgl. Abb. 3.13 rechts unten). In dieser Situation stoßen 3-Schicht-Architekturen an ihre Grenzen. Das klare Strukturprinzip einer Eins-zu-Eins-Beziehung von Entwurfsebenen und Architekturschichten wird durch die Existenz zusätzlicher Anwendungslogik in den Ressourcen aufgeweicht. Des Weiteren ergeben sich bei einer Schachtelung von mehreren 3-Schicht-AS effektiv *N-Schicht-Architekturen*.[58] Zur Integration heterogener und vollkommen autonomer Systeme werden dann auch an die Middleware ganz neue Anforderungen gestellt. Diese können von den bisher vorgestellten Techniken für homogene verteilte AS nicht in vollem Umfang erfüllt werden. Entsprechende integrative Middleware-Techniken für heterogene verteilte AS werden im nächsten Abschnitt ausführlich behandelt.

Zuvor soll aber noch die Betrachtung homogener vert. AS in Bezug auf *N-Schicht-Architekturen* abgeschlossen werden. Architekturen mit mehr als drei Schichten unterscheiden sich konzeptionell nicht mehr wesentlich von der 3-Schicht-Variante. Prinzipiell wird dabei die Anwendungslogik in weitere Schichten aufgegliedert. Hierbei ist allerdings schnell eine Grenze erreicht. Jede weitere Schicht führt nämlich zu erhöhtem Aufwand bei der Interaktion und mindert die Performanz. In bestimmten Fällen sind weitere Schichten aber zwingend erforderlich. Ein wichtiges Beispiel dafür sind Web-AS, deren Clients über das WWW angebunden sind. Hierbei werden Clients durch Web-Browser realisiert, die über offene Internet-Protokolle (z. B. HTTP) mit einem Web-Server kommunizieren. Dieser ist dabei an eine spezifische Präsentationsschicht angebunden. Der Web-Server, der aus dem Internet zugreifbar ist, muss aus Gründen der Sicherheit vom innerbetrieblichen Netz und den darin befindlichen Systemen getrennt werden und bildet somit eine eigene Schicht.

Für die Klasse der Web-AS wurden spezielle Middleware-Techniken entwickelt, die als *Applikationsserver* bezeichnet werden. Sie sind dadurch gekennzeichnet, dass sie die Dokumenten-zentrierte Interaktion des Web berücksichtigen, die eine erheblich komplexere Präsentationslogik impliziert. Dazu bieten sie eine integrierte Unterstützung für die Konstruktion und den Betrieb von Anwendungs- und Präsentationsschicht. Das Ziel ist es, für WWW-basierte Zugriffskanäle eine möglichst hohe Effizienz zu erreichen und die Verwaltbarkeit des gesamten Web-AS zu erhöhen.

[57] Vgl. 3.2.1.2.

[58] Z. B. führt die Integration eines 3-Schicht-AS als Ressource eines anderen 3-Schicht-AS zu insgesamt vier Schichten – nachzuprüfen in Abb. 3.13 rechts unten.

Bei der Beschreibung von Applikationsservern kann man Unterstützungsmechanismen für die Anwendungs- und Präsentationsschicht unterscheiden [ACK+04, S. 102 ff.]. In Bezug auf die Anwendungsschicht basieren Applikationsserver in der Regel auf einem Kern konventioneller Middleware. Dies beinhaltet meist einen ORB und einen TP-Monitor. Darauf werden in neueren Ansätzen oft Komponentenmodelle aufgesetzt, die ein modulares Programmiermodell bieten und verschiedene nützliche Abstraktionen zur Realisierung der Anwendungslogik bereitstellen. So ist es meist mit wenig Aufwand möglich, die Transaktionalität von Operationen oder die Persistenz von Attributen zu realisieren. Ferner sind in der Regel Sicherheitsmechanismen für die Authentifizierung von Clients und Autorisierung von Zugriffen enthalten. Zur Konnektivität der Anwendungsschicht mit verschiedenartigen Ressourcen werden in der Regel vielfältige offene Schnittstellen unterstützt.

Die Unterstützung der Präsentationsschicht ist das charakteristische Merkmal von Applikationsservern. Hier ist zunächst der programmatische Umgang mit verschiedenen im Web gebräuchlichen Präsentationssprachen und Formaten von Bedeutung. Dabei spielt vor allem die Extensible Markup Language (XML) eine entscheidende Rolle. Semi-strukturierte XML-Dokumente erleichtern nämlich die Transformation von Informationen in beliebige Präsentations- und Austauschformate. Zudem ermöglichen Applikationsserver die Anbindung der Präsentationslogik an Server für verschiedene offene Internet-Protokolle, um dynamisch generierte und personalisierte Inhalte an verschiedene Typen von Clients im Web zu kommunizieren. Neben Web-Browsern werden in der Regel auch Interaktionen mit herkömmlichen Clients, mobilen Geräten, E-Mail Empfängern und Web Service Clients unterstützt. Letztere Form wird im weiteren Verlauf noch ausführlich behandelt.

3.3.2.3. Middleware als Integrationsplattform

In Sektion 3.2.1.2 wurde bereits dargestellt, welche Rolle eine generelle Systemintegration für die betriebliche IV spielt. Dabei wurde deutlich, dass die Interaktions- und Koordinationsfähigkeit heterogener AS eine entscheidende Facette bildet. Zur Umsetzung dieser Anforderungen wurden nun in den letzten Abschnitten das Konzept vert. Systeme und dessen Realisierung durch Middleware-Technik eingeführt. Hierbei wurde gezeigt, dass Systemintegration prinzipiell zu heterogenen vert. AS mit N-Schicht-Architektur führt. Es wurde klar, dass *konventionelle Middleware*[59] dafür zwar grundlegende, aber letztendlich unzureichende Mittel bietet.

Die Anforderungen an eine integrative Middleware für heterogene verteilte AS besteht darin, die Anbindung einer unabsehbaren Vielfalt verschiedenartiger Systeme zu ermöglichen. Dabei haben sich insbesondere zwei Szenarien als praktisch relevant erwiesen und sind zurzeit Gegenstand intensiver Forschung und Entwicklung:[60]

[59] Mit „konventioneller Middleware“ sind hier solche Techniken gemeint, die für die Konstruktion homogener vert. AS mit 3-Schicht-Architektur ausgelegt sind.

[60] Es sei darauf hingewiesen, dass in Bezug auf die verwendeten Begriffe in der Literatur

- *A2Ai* – Zunächst ist die Integration innerbetrieblicher Altsysteme zu nennen, was im Folgenden als *A2A-Integration (A2Ai)* bezeichnet wird.

- *B2Bi* – Davon ist die zwischenbetriebliche Integration von AS zur automatisierten Interaktion von Unternehmen zu unterscheiden, was im Folgenden als *B2B-Integration (B2Bi)* bezeichnet wird.

In beiden Fällen kommen als Basistechnik im Wesentlichen *Message Broker* zum Einsatz, die eine wirkungsvolle Unterstützung der Anwendungsschicht bei der Realisierung von Integrationslogik bieten. Es wird nun zunächst deren Grundform im A2A-Kontext eingeführt. Dann folgen Überlegungen zur Ausweitung auf B2Bi-Szenarien.

Das wesentliche Problem der A2A-Integration besteht in der Integration von *Altsystemen*[61] einzelner Unternehmen im Sinne integrierter IV. Ein AS wird dabei immer dann als Altsystem bezeichnet, wenn es zu einem anderen als dem ursprünglich geplanten Zweck eingesetzt wird. Zu den wesentlichen Merkmalen von Altsystemen gehören u. a. die völlige Unbestimmtheit ihrer Techniken und die Autonomie ihres Interaktionsverhaltens, d. h. dass diese sich u. a. in Bezug auf ihre Systemplattformen, Funktionen und Schnittstellen, Datenmodelle und -formate, Sicherheitsanforderungen und -mechanismen sowie Kommunikationsmodelle und -paradigmen zum Teil erheblich unterscheiden können.

Als unmittelbare Konsequenz ergibt sich, dass die Integrationslogik bei A2Ai in der Regel nicht Teil der Altsysteme ist – diese wissen meist gar nichts voneinander – sondern als separate Anwendungslogik hinzugefügt wird. In diesem Sinne ergibt sich ein neues AS höherer Ordnung. Theoretisch kann dieses neue AS irgendwann wiederum zum Altsystem und Gegenstand von A2Ai werden. Eine derartige Kaskadierung wird dann nur durch die Komplexität der entstehenden Architektur gestoppt. In jedem Fall ergibt sich aber eine Architektur mit verschiedenen Schichten unabhängiger Anwendungslogik (vgl. Abb. 3.13 rechts unten). Dies entspricht nicht mehr der klassischen 3-Schicht-Architektur, die den Zugriff auf verschiedene explizit als Server ausgelegte Systeme zum Ziel hat.

Altsysteme beinhalten in der Regel eine wesentlich komplexere Funktionalität als Server und sind zudem nicht per se zur Interaktion ausgelegt. Infolgedessen zeigt auch die Mittelschicht A2A-integrierter AS eine ganz andere Charakteristik als bei 3-Schicht-AS.

Zunächst muss in der Mittelschicht eine Anbindung der Altsysteme an die Integrationslogik realisiert werden. Entsprechende Konzepte kreisen in der Regel um

keine Einigkeit herrscht. So definiert z. B. Müller [Mül05, S. 36 f.] EAI als allg. Oberbegriff und unterscheidet genauso wie Busler [Bus03, S. 17] die konkreten Formen A2A und B2B. Alonso et. al [ACK+04, S. 67 ff.] und auch Ließmann [Mer97, S. 180 f.] bezeichnen hingegen nur die innerbetriebliche Integration von AS als EAI. Auf Grund dieser Unstimmigkeiten wird der EAI-Begriff im Folgenden nicht weiter verwendet.

[61] Altsysteme werden in der Literatur auch oft als *Legacy Systeme* bezeichnet.

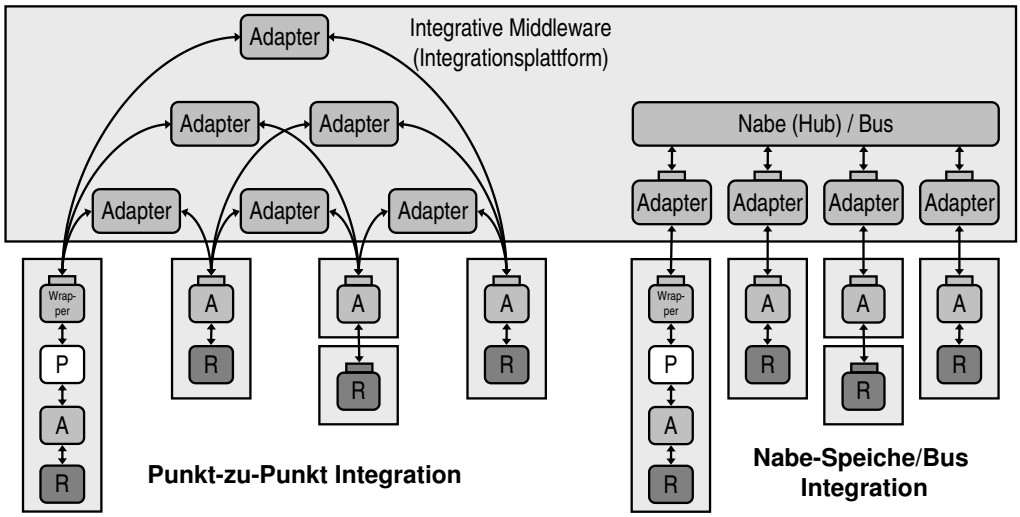

Abbildung 3.14.: Architekturvarianten bei A2A-Integration

Adapter- bzw. *Wrapper*-Techniken. Wrapper kapseln geschlossene AS und machen sie über offene Schnittstellen zugreifbar. Adapter gleichen die spezifischen Unterschiede heterogener Systeme aus, stellen Interoperabilität sicher und ermöglichen die Interaktion. Zur Realisierung der Integrationslogik werden dann im Wesentlichen zwei Architekturen unterschieden. Bei *Punkt-zu-Punkt Integration* wird die Integrationslogik (falls vorhanden) auf verschiedene Adapter aufgeteilt, die jeweils genau zwei Altsysteme verbinden. Die Gesamtheit aller betrieblichen A2A-Beziehungen ist durch eine oft große Menge von Adaptern realisiert, die bei jedem hinzukommenden Altsystem noch erheblich wächst. Hingegen werden bei *Nabe-Speiche Integration* alle Altsysteme über genau einen Adapter mit einer zentralen Komponente verbunden. Diese fungiert als neutraler Vermittler aller Interaktionsbeziehungen und realisiert dabei die Integrationslogik in einheitlicher zusammenhängender Weise. Ist die zentrale Komponente als verteiltes System realisiert, wird dies stattdessen oft als *Bus Integration* umschrieben. In beiden Fällen ist für jedes hinzukommende Altsystem nur ein zusätzlicher Adapter erforderlich (vgl. Abb. 3.14).

Die Integrationslogik selbst spiegelt meist die Unternehmensorganisation und ihre Geschäftsprozesse wider. Daraus folgt, dass sie vergleichsweise langwierige Interaktionen[62] zwischen Altsystemen beinhalten kann. Zudem muss sie sich den vergleichsweise häufigen Änderungen der betrieblichen Organisationsstrukturen anpassen.

Für die spezifischen Anforderungen des A2Ai zeigen sich die üblichen Unterstützungsmechanismen klassischer Middleware als unzureichend. Deren RPC- oder ORB-basierte Kommunikationsinfrastrukturen sind in der Regel für schnelle synchrone Interaktionen mit kurzen Nachrichten ausgelegt und optimiert. Zudem werden die

[62]Die Durchlaufzeit von Geschäftsprozessen kann leicht mehrere Stunden oder Tage betragen.

Kommunikationspartner hier in den Endpunkten direkt adressiert[63], was zu einer engen Kopplung führt und die Änderung von Interaktionsbeziehungen erschwert. Message Oriented Middleware bietet hier zwar asynchrone Interaktionsmechanismen, setzt aber meist eine direkte Adressierung von Empfängern und kurze Nachrichten voraus [ACK+04, S. 73 ff.].

In der Folge wurden für A2Ai spezielle Middleware-Mechanismen entwickelt, die oft als *Integrationsplattformen* bezeichnet werden. Als Kern solcher Plattformen hat sich (neben den nach wie vor aktuellen TP-Monitoren) eine erweiterte Form von MOM etabliert, die als *Message Broker* bezeichnet wird. Message Broker besitzen die Fähigkeit zur Ausführung individueller Logik für die Verarbeitung und Weiterleitung von Nachrichten. Diese Logik wird meist durch Regel-basierte Sprachen spezifiziert und den Nachrichten-Queues zugewiesen. Message Broker können eingehende Nachrichten analysieren und entsprechend ihres Absenders, Typs oder Inhalts zunächst transformieren und dann an geeignete Empfänger weiterleiten. Auf diese Weise lassen sich vielfältige Interaktionsmodelle realisieren und eine Entkoppelung von Sender und Empfänger erreichen.[64]

Im A2Ai-Kontext erlauben Message Broker eine direkte Umsetzung der Nabe-Speiche Architektur. Hierbei werden heterogene Altsysteme mithilfe spezifischer Adapter zum wechselseitigen asynchronen Nachrichtenaustausch mit dem Message Broker befähigt. Letzterer realisiert die Interaktion zwischen den Adaptern in Form von Regeln zur Verarbeitung und Weiterleitung von Nachrichten. Die eigentliche Integrationslogik einer administrativen Domäne (im Sinne der zu unterstützenden Geschäftsprozesse) wird dann in der Regel als zusätzlicher Client des Message Brokers realisiert. Dadurch liegt sie an zentraler Stelle vor. Ihre Entwicklung, Wartung und Anpassung kann dann unabhängig von den Altsystemen, auf Basis spezieller Modelle/Sprachen und in ganzheitlicher Weise erfolgen. Verschiedene administrative Domänen können leicht durch die Kopplung mehrerer Message Broker im Sinne einer Bus-Architektur überbrückt werden (Abb. 3.15).

Durch die Unterstützung von asynchronen Interaktionsmodellen zwischen entkoppelten Altsystemen auf Basis der Nabe-Speiche Architektur bilden Message Broker den Kern vieler Integrationsplattformen. Sie bilden dort eine Kommunikationsinfrastruktur, die je nach Anwendungskontext in verschiedene Richtungen erweitert werden kann und muss. Eine wesentliche Richtung wird dabei durch die Probleme von Planung, Realisierung und Betrieb der Integrationslogik vorgegeben. Die Regelspra-

[63] Dies gilt sowohl bei Verwendung von physikalischen Adressen (z. B. Netzwerkadressen) als auch bei Nutzung von Namensdiensten. Eine Ausnahme bilden *Trader* [MMJL94], die aber für die Vermittlung von Objekten, nicht aber von kompletten Systemen ausgelegt sind.

[64] Als Beispiel kann das verbreitete *Publish/Subscribe*-Modell genannt werden, bei dem Empfänger sich bei Interesse an bestimmten Nachrichtentypenparametern registrieren müssen und Sender ihre Nachrichten ohne Angabe des Empfängers abschicken. Ein Message Broker kann dann eingehende Nachrichten in Bezug auf ihre Typen/Parameter analysieren und an die registrierten Empfänger weiterleiten.

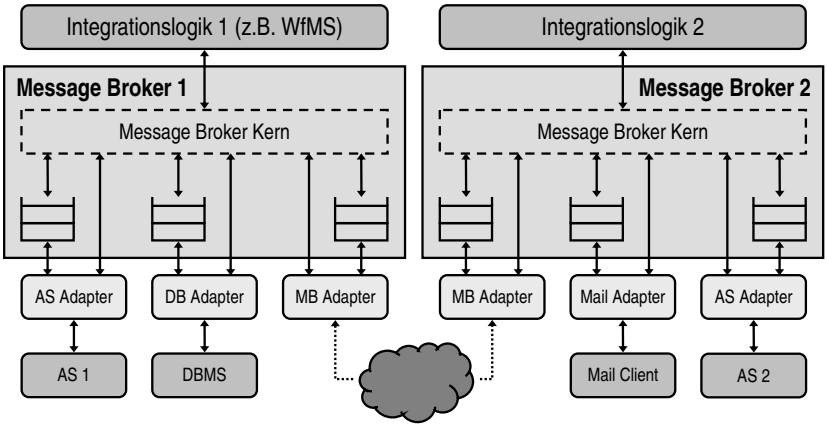

Abbildung 3.15.: EAI mit Message Broker Middleware

chen der Message Broker bilden nämlich keine geeignete Lösung zur Umsetzung und Wartung von Geschäftsprozessen. Diesbezüglich werden oft Lösungen diskutiert, die auf einer Erweiterung von Message Brokern durch Workflow-Management-Systeme basieren.[65]

3.3.2.4. B2B-Integration Middleware

Die Realisierung von B2Bi Szenarien bildet eine besondere Herausforderung für Integrationsplattformen. Unter Business-to-Business-E-Commerce bzw. Electronic Business (E-Business) wird im Allgemeinen die IV-gestützte Abwicklung zwischenbetrieblicher Geschäftstransaktionen verstanden. *B2B-Integration (B2Bi)* wird als Oberbegriff für IV-Techniken gebraucht, die die zwischenbetriebliche Interaktion von Anwendungssystemen unterstützen [MBB+03].[66] Der Einsatz von B2Bi-Techniken zur Lösung eines spezifischen Integrationsproblems resultiert in einem *B2Bi-Anwendungssystem*. Zur weiteren Strukturierung dieser Klasse identifizieren Medjahed et al. eine Menge von Merkmalen (Dimensionen), in denen sich B2Bi-AS maßgeblich unterscheiden können (vgl. Tab. 3.3).

Das Kernproblem von B2Bi besteht genau wie bei A2Ai in der Integration heterogener AS. Der wesentliche Unterschied liegt darin, dass diese AS Teile verschiedener Unternehmen bilden. Daraus ergibt sich eine Reihe von Konsequenzen für ihre Integration. Zunächst sind AS und Middleware im B2Bi Kontext nicht durch LAN, sondern WAN vernetzt, wobei Letztere heute in der Regel durch das Web gegeben sind. Zudem kann die Integration im Allgemeinen auf keiner gemeinsamen Wissens- oder Vertrauensbasis aufsetzen.

[65] Die Anwendung von WfMS bei der Systemintegration wird in Sektion 3.3.3 behandelt.

[66] Medjahed et al. definiert den Interaktionsbegriff als Kombination aus Interoperabilität und Integration von Anwendungssystemen.

Merkmal	**Ausprägungen**
Kopplung	Verschiedene Grade der Festigkeit der Kopplung von AS und Länge der Beziehung
Heterogenität	Verschiedene Grade der Homogenität von Daten/Strukturen/Anwendungslogik der AS
Autonomie	Verschiedene Grade der Berücksichtigung übergeordneter Regeln bei Entwurf/Kommunikation/Ausführung von AS
Externes Management	Verschiedene Grade der externen Sichtbarkeit, Steuerbarkeit und Kontrollierbarkeit von AS
Adaptierbarkeit	Verschiedene Grade der Anpassbarkeit von AS an interne/externe Änderungen
Sicherheit	Verschiedene Grade der Umsetzung von Sicherheitsmechanismen in den AS
Skalierbarkeit	Verschiedene Grade der Erweiterbarkeit der quantitativen Leistungsfähigkeit der AS

Tabelle 3.3.: Unterscheidungsmerkmale von B2Bi-Systemen [MBB+03]

Die Realisierung verteilter AS über das Web ist prinzipiell auf verschiedenen Wegen möglich. Neben der schon beschriebenen Kopplung von Browser-basierten Clients mit der Mittelschicht eines Applikationsservers, kann sie auch dadurch erreicht werden, indem Clients mit Clients, Mittelschicht mit Mittelschicht oder Mittelschicht mit Servern über das Web kommunizieren. Grundsätzlich sind dabei die verbindungsspezifischen Protokolle über das Internet abzuwickeln. Im einfachsten Fall kann dies unmittelbar erfolgen.[67] Da solche Verbindungen aber meist durch Firewalls unterdrückt sind, wird stattdessen in der Regel ein Indirektionsschritt eingebaut, bei dem die eigentlichen Protokolle in Trägerprotokolle wie HTTP verpackt werden, die Firewalls passieren können.[68] Auf diese Weise ist die Möglichkeit zur breit gefächerten Kommunikation in B2Bi-AS also generell gegeben. In der Praxis hat sich jedoch kein spezifisches Protokoll für eine breite Nutzung im Kontext des Web durchsetzen können.

Für den Einsatz von Message Brokern bietet sich nun aus praktischer Sicht eine zentralisierte Lösung an. Ein zentraler Message Broker kann den asynchronen und entkoppelten Nachrichtenaustausch zwischen Adaptern realisieren, die über das Web an die AS verschiedener Unternehmen gekoppelt sind. Tatsächlich wurde dieser Ansatz erfolgreich durch verschiedene so genannte *Commerce Hubs* wie Ariba oder CommerceOne realisiert, hat sich jedoch aus verschiedenen Gründen nicht durchsetzen können. Ein wesentliches Argument gegen zentralisierte Middleware ist das fehlende Vertrauen in den Betreiber. Dies betrifft z. B. Kontrollverlust und

[67] Ein Beispiel dafür ist die Verbindung CORBA-basierter Mittelschichten per TCP/IP und Internet Inter-ORB Protocol (IIOP).

[68] Im Fall von CORBA ist z. B. die Kombination IIOP-over-HTTP gebräuchlich.

Abhängigkeit in Bezug auf kritische Unternehmensprozesse sowie Vertraulichkeit von Transaktionsdaten und Nachrichteninhalten. Die nahe liegende Lösung besteht in einer Verteilung der Message Broker Middleware auf verschiedene Unternehmen. Die Message Broker werden in diesem Fall gekoppelt, indem sie sich wechselseitig als Clients registrieren. Die Implikationen eines fehlenden Vertrauenskontextes wirken sich dann wesentlich weniger problematisch aus. Einzelne Unternehmen behalten die Kontrolle über ihre Middleware und können dabei die Verbreitung und Sichtbarkeit von Nachrichten genau kontrollieren. Das Problem dieser Lösung liegt im geringen Grad der Standardisierung von Message Broker Protokollen. Die Folge ist, dass alle Unternehmen die gleichen Middleware Produkte nutzen müssen. Darüber hinaus müssen für verschiedene B2Bi-AS ggf. verschiedene Middleware-Produkte gleichzeitig und nebeneinander betrieben werden – eine Anforderung, die sich in Bezug auf Wartungsaufwand und Kosten als kaum realisierbar erweisen dürfte.

Das Fehlen einer gemeinsamen Wissensbasis hat darüber hinaus noch weitere Konsequenzen. Diese führen dazu, dass die Kommunikation von Nachrichten zwischen den AS verschiedener Unternehmen lediglich die unterste Ebene von B2Bi Interaktionen darstellt. Da nämlich über die Inhalte dieser Nachrichten nicht notwendigerweise Einigkeit bestehen muss, ist auf einer subsequenten Inhaltsebene die Syntax und Semantik von Nachrichten festzulegen. Zwei grundsätzliche Ansätze hierzu lassen sich am Beispiel der EDIFACT und XML-Standards erläutern. Der erste Ansatz besteht in der detaillierten Standardisierung möglicher Nachrichtentypen. So wird z. B. als Teil von EDIFACT exakt die Struktur von Nachrichten sowie die Bedeutung der einzelnen Elemente spezifiziert. Dadurch sind die Nachrichteninhalte jederzeit eindeutig und können automatisch ausgewertet und verarbeitet werden. Die Anzahl der möglichen Nachrichten und zu vermittelnden Inhalte ist jedoch naturgemäß begrenzt. XML[69] erlaubt im Gegensatz die syntaktische Spezifikation beliebiger Nachrichtentypen.[70] Auf dieser Basis ist eine automatisierte Typprüfung und Verarbeitung von Nachrichten möglich. Der potenziellen Vielfalt von Nachrichtentypen steht jedoch deren unbestimmte Semantik gegenüber. Diese muss daher durch erweiterte Sprachen[71] und/oder Standardisierung[72] ergänzt werden.

[69] Die Extensible Markup Language (XML) ist ein W3C-Standard [BPSM+04]. Eine Einführung findet sich z. B. bei Weitzel, Harder u. a. [WHB01].

[70] XML wird zur Spezifikation beliebiger semi-strukturierter Dokumente eingesetzt, die natürlich auch als Nachrichten verwendbar sind. Die Spezifikation eines Dokumententyps erfolgt mittels Document Type Definition (DTD) oder XML-Schema [WHB01, S. 25 ff.].

[71] Derartige Ansätze werden zurzeit intensiv im Kontext des *Semantic Web* untersucht [BLHL01]. An erster Stelle ist hierbei die *Web Ontology Language (OWL)* [MH04] zu nennen.

[72] Verschiedene Gremien und Konsortien haben mittlerweile Dokumententypen zum zwischenbetrieblichen Nachrichtenaustausch standardisiert und dabei deren Semantik in der Regel durch textuelle Beschreibung festgelegt. Als Beispiele hierfür können u. a. RosettaNet, Commerce XML (cXML) und XML Common Business Library (xCBL) genannt werden; viele weitere finden sich z. B. auf der Web-Seite von OASIS: `http://www.oasis-open.org/cover/` (Zugriff am 1.8.2005).

Selbst wenn die Kommunikation in Form von Nachrichten möglich ist und über deren Aufbau und Sinn Einigkeit erzielt werden kann, sind noch Fragen offen, die der Durchführung von B2B-Interaktionen im Wege stehen. Diese Fragen sind eng mit den Geschäftsprozessen der Interaktionspartner verbunden. In den Geschäftsprozessen drückt sich – vereinfacht gesagt – die Art und Weise aus, in der die Unternehmen ihre Geschäfte vollziehen. Falls diese Praktiken nicht in Einklang miteinander stehen, ist die Interaktion zum Scheitern verurteilt. Der Einklang der Geschäftsprozesse muss somit im Vorfeld geklärt und im Betrieb abgesichert werden. In Bezug auf den zwischenbetrieblichen Nachrichtenaustausch besteht ein wesentlicher Aspekt darin, eine Übereinkunft über die Typen der Nachrichten und die Reihenfolge ihres austauschs zu erreichen. Eine Spezifikation dieses Ablaufs wird als *B2B-Protokoll* bezeichnet [Bus01].[73] Die Verständigung der Interaktionspartner auf ein B2B-Protokoll ist nun von entscheidender Bedeutung. Das gemeinsame Prozesswissen verhindert nicht nur, dass Missverständnisse im Interaktionsverlauf die geschäftliche Transaktion zum Scheitern bringen. Es führt auch zu mehr Transparenz der B2B-Interaktion und ermöglicht deren Automatisierung [DHL01].

B2B-Protokolle werden in der Regel durch abstrakte Rollen sowie die möglichen Abfolgen von Nachrichtenübertragungen zwischen diesen Rollen beschrieben. Sie können in Form so genannter *öffentlicher Prozesse* aus Sicht einer Rolle oder in (äquivalenter) Form von *Choreografien* aus neutraler Perspektive vorliegen. Um eine Einigung über das B2B-Protokoll zu erzielen, müssen sich die Interaktionspartner zunächst auf gemeinsame Metamodelle und Spezifikationssprachen für Interaktionsprozesse verständigen [DHL01]. Hiermit können B2B-Protokolle dann entweder in standardisierter oder individueller Weise beschrieben werden.[74]

Das Management von B2B-Protokollen umfasst Mechanismen, um Protokolle von Partnern aufzufinden, mit den eigenen zu vergleichen, sie dann vertraglich zu sichern und schließlich den öffentlichen Prozessverlauf entsprechend zu überwachen [Bus01]. Um den reibungslosen Ablauf einer B2B-Interaktion zu gewährleisten, müssen sich die B2Bi-AS der Interaktionspartner exakt an den öffentlichen Prozess ihrer Rolle halten. Dabei ist zu berücksichtigen, dass alle B2B-Nachrichten in die privaten Geschäftsprozesse der interagierenden Unternehmen (z. B. realisiert durch die Integrationslogik einer A2A-Plattform) eingehen bzw. aus diesen hervorgehen. Diese innerbetrieblichen

[73] Für das Konzept existieren in der Literatur verschiedene äquivalente Bezeichnungen wie *geschäftliche Konversation* [YHU02, HNK02], *Choreografie* [Pel03] oder *kollaborativer Prozess* [DHL01, AMS02].

[74] Als Beispiele standardisierter und individueller B2B-Protokolle lassen sich RosettaNet und ebXML nennen. Das RosettaNet-Konsortium entwickelt Standards für B2B-Interaktionen der IT-Industrie [Ros04]. Dabei wird u. a. ein Katalog öffentlicher Geschäftsprozesse vorgegeben, die als Partner Interface Processes (PIP) bezeichnet werden. Electronic Business XML (ebXML) ist ein von UN/CEFACT und OASIS unterstützter generischer B2Bi-Standard [UO04]. ebXML bietet mit dem Business Process Specification Schema (BPSS) einen Rahmen zur individuellen Spezifikation öffentlicher Geschäftsprozesse und Choreografien. Diese werden als Basis zur vertraglichen Festlegung von Kollaborationen im Rahmen von Collaboration Protocol Agreements (CPA) verwendet.

Prozesse sind in der Regel vertraulich und sollen von spezifischen B2B-Protokollen unabhängig und flexibel bleiben. Daher werden private Geschäftsprozesse im Allgemeinen nach außen gekapselt und möglichst lose an die externen Schnittstelle(n) von B2Bi-AS gebunden [Bus01]. Die Implementierung ist dabei für die Interaktion grundsätzlich unerheblich, solange der dadurch realisierte öffentliche Geschäftsprozess dem B2B-Protokoll entspricht.

Aus erweiterter Perspektive bilden B2B-Protokolle eine wechselseitige Verbindung von privaten Geschäftsprozessen einzelner Partner, was dann zusammen zu einem (logischen) organisationsübergreifenden Gesamtprozess führt (Distributed Business Process (DBP)[75]). Bei dieser Perspektive tritt die B2B-Interaktion als Teilaspekt in den Hintergrund. Stattdessen wird die Modellierung, Optimierung und Automatisierung von DBP als ganzheitliche Wertschöpfungsketten hervorgehoben.

Da die Automation innerbetrieblicher Geschäftsprozesse durch Workflow-Modelle und WfMS-Techniken dominiert wird, knüpfen viele Ansätze zur Automation von sowohl B2B-Protokollen als auch DBP daran an. Die resultierenden Konzepte für zwischenbetriebliche Workflow Systeme (Inter-Enterprise Workflow-Systems) werden in der folgenden Sektion ausführlich dargestellt.

Zusammenfassend lässt sich festhalten, dass B2B-Interaktionen konzeptionell in die drei Ebenen der *Kommunikation*, des *Inhalts* und der *Geschäftsprozesse* gegliedert werden können [MBB+03]. Auf der Kommunikationsebene besteht das Ziel darin, eine nahtlose Integration von Kommunikationsprotokollen zu erreichen. Die Inhaltsebene soll zu einer nahtlosen Integration von Datenformaten, Datenmodellen und Sprachen führen. Der Fokus der Geschäftsprozessebene liegt auf der Befähigung autonomer heterogener Unternehmen zur wechselseitigen Veröffentlichung, Verhandlung und Erfüllung geschäftlicher Fähigkeiten und Bedingungen in elektronischer Form. Alle Ebenen müssen durch die Anwendungs- bzw. Integrationslogik von B2Bi-AS berücksichtigt werden. Dazu ist eine Infrastruktur erforderlich, die über den Umfang von A2Ai-Plattformen hinausgeht.

Die Bestandteile einer entsprechenden B2Bi-Plattform sollen hier anhand der generischen *B2Bi-Framework-Architektur* nach Medjahed et al. erläutert werden (Abb. 3.16). Diese unterteilt die Mittelschicht von B2Bi-AS in eine A2Ai- und eine B2Bi-Subschicht. Die A2Ai-Schicht integriert interne Ressourcen und Altsysteme, deren Kopplung über Adapter erfolgt. Die Integrationslogik setzt dabei organisatorische Regeln und Prozesse eines Unternehmens um. Dies kann in der betrieblichen Praxis durch sehr unterschiedliche Anwendungen und Middleware realisiert werden. Aktuelle Ansätze basieren in der Regel – wie schon gezeigt – auf Message Brokern und WfMS.

Zur zwischenbetrieblichen Interaktion werden die internen Systeme an eine übergeordnete B2Bi-Schicht (hier als *External Interaction Gateway* bezeichnet) angebunden. Aus interner Sicht stellt sich dies als Menge zusätzlicher Adapter dar, die zur An-

[75] Auf Grund der weiten Verbreitung wird im Folgenden das angelsächsische Akronym DBP verwendet.

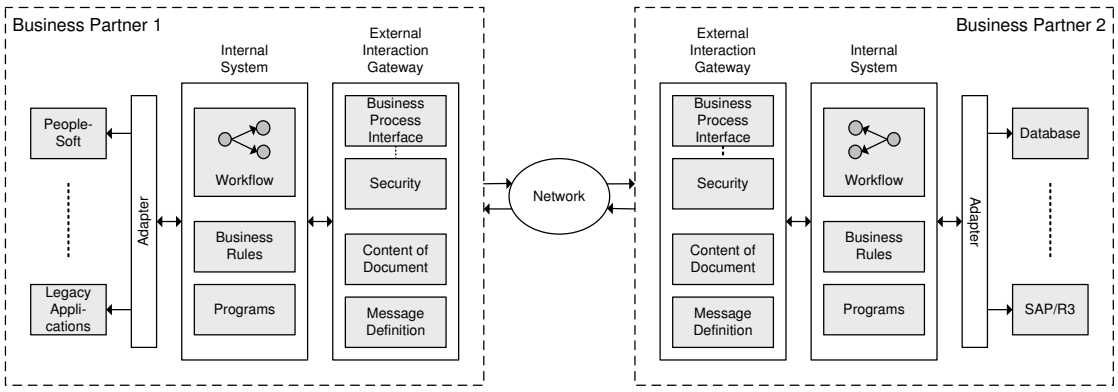

Abbildung 3.16.: Generische B2Bi-Framework-Architektur (nach [MBB+03])

bindung externer Systeme dienen. Diese Anbindung externer Systeme über alle drei Ebenen zwischenbetrieblicher Interaktion wird durch Mechanismen der B2Bi-Schicht unterstützt. Dazu gehören u. a. Mechanismen zur Transformation von Nachrichtenstrukturen und Übersetzung von Inhalten in gemeinsame Übertragungsformate, Übertragung von Nachrichten durch gemeinsame WAN- bzw. Web-basierte Austauschprotokolle, Management externer Geschäftsprozesse und Abgleich mit internen Geschäftsprozessen sowie Signatur und Verschlüsselung von Nachrichten zur Umsetzung gemeinsamer Sicherheitsrichtlinien. Die einzelnen Mechanismen richten sich in der Regel nach den Vorgaben von B2Bi-Standards oder zwischenbetrieblichen Absprachen. Dabei müssen oft verschiedene Standards parallel unterstützt werden, um die Interaktion mit verschiedenen Partnern zu ermöglichen.

3.3.3. Workflow als zwischenbetriebliche Koordinationsebene

In den vorangegangenen Sektionen wurde das Potenzial von Middleware als Interaktionsplattform für verteilte AS erläutert. Dabei wurde die Entwicklung der systemtechnischen Unterstützungsfunktionen angefangen von Interoperabilitätsmechanismen für heterogene Systemplattformen bis hin zur inner- und zwischenbetrieblichen Systemintegration nachvollzogen. Am Ende standen komplexe Integrationsplattformen, die Altsysteme mithilfe von Adaptern als homogene interoperable Komponenten kapseln und deren lose Kopplung durch flexible nachrichtenbasierte Interaktionsbeziehungen erlauben.

Es zeigte sich dabei, dass Middleware sowohl bei inner- als auch zwischenbetrieblicher Integration nicht nur Potenzial zur Kommunikation einzelner Nachrichten, sondern auch zur Koordination der dahinterstehenden Prozesse bietet. Diese Prozesse werden bei A2Ai durch die Integrationslogik für Altsysteme vorgegeben, die im Wesentlichen auf private Geschäftsprozesse zurückgeht. Bei B2Bi werden sie hingegen durch die in B2B-Protokollen spezifizierten öffentlichen Geschäftsprozesse bestimmt.

In beiden Fällen sollen Mechanismen prozessorientierter Middleware durch Abstraktionen und Infrastruktur bei der Realisierung von AS helfen. Beide Fälle sind auch für die Realisierung virt. Dienstleistungen von Interesse, um sowohl Teildienstleistungen als auch deren Zusammenspiel modellieren und koordinieren zu können.

Im Allgemeinen sind Konzepte und Techniken für Planung, Entwurf, Analyse, Ausführung, Beobachtung und Evolution von Prozessen Gegenstand des *Workflow Managements*. Im Rahmen der Kommunikationsinfrastruktur einer Integrationsplattform kann auch das Management von Integrations- bzw. Interaktionsprozessen durch Workflow-Management-Systeme (WfMS) realisiert werden. In Bezug auf virt. Dienstleistungsprozesse sind dabei insbesondere Ansätze für zwischenbetriebliche bzw. organisationsübergreifende Workflows von Interesse. In diesem Sinne werden im Folgenden zunächst einige Grundlagen des allgemeinen Workflow Managements eingeführt. Im Anschluss erfolgt eine Diskussion zwischenbetrieblicher Workflow Systeme (IEWS). Am Ende werden solche Ansätze hervorgehoben die in Hinblick auf VU entwickelt wurden.

3.3.3.1. Grundlegende Workflow-Konzepte

Workflow-Systeme entstammen ursprünglich dem Bereich der Büroautomation.[76] Unter dem Leitbild des „papierlosen Büros“ wurden gut strukturierte Arbeitsprozesse als *administrative Workflows* abgebildet, deren Aktivitäten mitsamt dazugehöriger (elektronischer) Dokumente automatisch an geeignete Mitarbeiter delegiert werden konnten [GHS95]. Mittlerweile decken verschiedene Formen von Workflow-Systemen einen breiten Bereich zwischen Computer Supported Cooperative Work (CSCW) und Transaction Processing ab.[77] Auf der einen Seite bilden *ad-hoc Workflows* relativ unstrukturierte kooperative Prozesse zwischen Menschen ab [GHS95]. Auf der anderen Seite setzen *Production Workflows* hochgradig automatisierte Prozesse um, bei denen die meisten Aktivitäten von AS durchgeführt werden [LR00].

Aus konzeptioneller Sicht dienen Workflows zur Koordination derjenigen Einheiten, die an der Ausführung von Prozessen (Geschäfts- oder Software-Prozessen) beteiligt sind [BzG02]. Koordination kann dabei in Anlehnung an Malone als Management von Beziehungen zwischen Aktivitäten aufgefasst werden [MC94]. Bei Workflows bezieht sich die Koordination auf das Management von a) Datenbeziehungen (die Ausführung einer Aktivität basiert auf dem Ergebnis einer anderen Aktivität) und b) geteilten Ressourcen (die Ausführung zweier Aktivitäten beruht auf den gleichen Ressourcen). Datenbeziehungen zwischen Aktivitäten werden im Workflow durch Kontroll- und Datenfluss geregelt. Das Teilen von Ressourcen zwischen Aktivitäten wird durch Terminplanung und Ressourcenzuteilung berücksichtigt. Die Eigenschaften eines Prozesses werden durch eine strukturierte Menge abstrakter *Aktivitäten*,

[76] Eine Übersicht der Workflow-Thematik findet sich z. B. in [GHS95]. Ausführliche Einführungen bieten u. a. [AH02, JBS97].

[77] Eine Klassifikationen verschiedener Workflow-Systeme findet sich z. B. in [BzG02].

Vorgangsdaten und Ressourcen in Form von *Bearbeitern* oder *Anwendungen* beschrieben; dies wird als *Prozessdefinition* bezeichnet.[78] Die Struktur der Prozessdefinition gibt Präzedenzen zwischen Aktivitäten wieder. Aktivitäten sind in diesem Stadium als abstrakte und Ressourcen als leere Rolle zu sehen. Der Koordinationsvorgang erfolgt dann für konkrete *Fälle* von Prozessen bzw. für *Prozessinstanzen*. Bei einem Fall werden die nun als *Aktivitätsinstanzen* bezeichneten Aufgaben akut. Für die damit assoziierten Rollen wird daher eine passende Ressource zugeteilt. Je nach Art der Ressource wird im Anschluss eine *Elementaraufgabe* in den *Eingangskorb* eines Bearbeiters gestellt oder eine Anwendung wird aufgerufen und erst jetzt erfolgt mit der Bearbeitung der Aufgabe die eigentliche Aktivität.

Die Repräsentation von Workflows als Prozessdefinitionen kann auf Basis so verschiedener Formalismen wie gerichteten Graphen (z. B. Flussdiagramme), Transitionssystemen (z. B. Zustandsdiagramme, Petrinetze) oder Prozess Algebra (z. B. π-Calculus) erfolgen. Heute ist eine breite Palette verschiedener produktbezogener oder standardisierter Modelle und Spezifikationssprachen für Workflow-Prozesse verfügbar. Dabei sind deutliche Unterschiede u. a. in der Ausdrucksmächtigkeit in Bezug auf Prozesseigenschaften und Prozessanalyse, in der formalen Exaktheit, z. B. in Bezug auf die Prozesssemantik, sowie in der Komplexität und Verständlichkeit bei praktischer Verwendung festzustellen. Diese Aspekte sind Gegenstand aktueller – und zum Teil kontroverser – Diskussion.[79] In der vorliegenden Arbeit wird die Auffassung vertreten, dass kein einzelnes Modell und keine einzelne Sprache allen möglichen Anwendungen des Workflow-Prinzips gerecht werden kann. Gleichzeitig sollen die im weiteren Verlauf angestellten Betrachtungen aber möglichst allgemeingültig sein und sich nicht auf spezifische proprietäre oder gar exotische Ansätze stützen. Im Folgenden wird daher der intuitive und weit verbreitete Ansatz gerichteter Graphen zugrunde gelegt.

Abb. 3.17 zeigt einen beispielhaften Workflow. Die informale Prozessdefinition beruht auf einer Graph-basierten Pseudo-Notation nach Alonso et al. [ACK+04, S. 85]. Die Knoten eines solchen Graphen repräsentieren jeweils entweder eine Aktivität (Arbeitsknoten bzw. Working Node), eine Ablaufsteuerung (Ablaufknoten bzw. Routing Node) oder Anfang/Ende des Prozesses (Start-/Stop-Knoten). Bei den Arbeitsknoten unterscheidet man im Wesentlichen zwei Varianten: Sie repräsentieren entweder atomare Aktivitäten, die an Ressourcen delegiert werden, oder sie verweisen auf *Teilprozesse*, was eine Verschachtelung von Prozessen ermöglicht. Prozesse beginnen immer an einem Start-Knoten und enden, wenn der Prozessverlauf an einem Stop-Knoten angelangt ist. Gerichtete Kanten repräsentieren Präzedenzen zwischen Knoten und werden als *Transitionen* bezeichnet. Transitionen können an *Übergangsbedingungen* geknüpft sein. Diese beziehen sich auf die im Prozessverlauf anfallenden

[78] Über die deutsche Workflow-Terminologie herrscht in der Literatur keine Einigkeit. Hier wird im Folgenden weitgehend die Terminologie der WfMC [WfM99] mitsamt empfohlener Übersetzung [Kre99] verwendet.

[79] Vgl. z. B. [PW03b, Aal04].

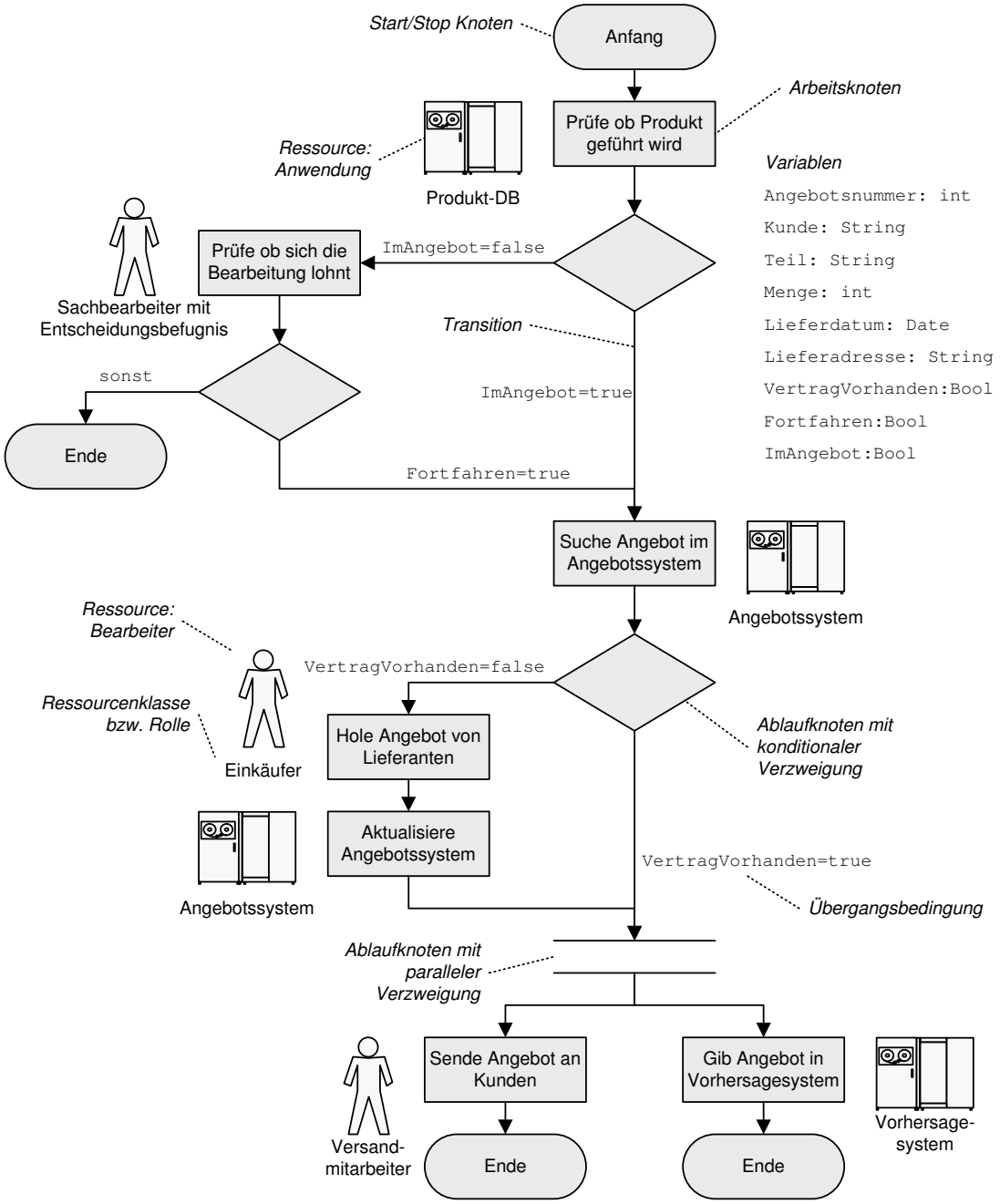

Abbildung 3.17.: Workflow-Prozessdefinition für einen Geschäftsprozess zur Bearbeitung von Angebotsanfragen (nach [ACK+04, S. 85])

Vorgangsdaten und werden mithilfe einer Regelsprache formuliert. Vorgangsdaten werden durch *Variablen* abgebildet und können in Aktivitäten einfließen oder aus ihnen hervorgehen. Mithilfe von Ablaufknoten lassen sich verschiedene Formen *bedingter Verzweigungen und asynchroner Zusammenführungen* bilden, mit denen Alternativen im Prozessverlauf modelliert werden können. Eine weitere wesentliche Form der Ablaufsteuerung ist die *parallele Verzweigung und Synchronisation* von Prozessanteilen. Die darin enthaltenen Aufgaben werden dann nebenläufig bearbeitet. Dies ist möglich, da den Aktivitätsinstanzen konkrete Fälle individuelle Ressourcen (z. B. Sachbearbeiter, IT-Systeme) zur Ausführung zugeteilt werden. Die Ressourcenzuteilung kann entweder fest vorgegeben, durch Regeln dynamisch bestimmt oder an einen Broker delegiert werden.

Workflow-Management-Systeme Die Realisierung von Workflows als AS erfolgt heute im Wesentlichen mithilfe von Workflow-Management-Systemen (WfMS). Deren Prinzip besteht darin, das Prozessmanagement von der Prozessausführung zu trennen. Im Gegensatz zu einem konventionellen Anwendungssystem, bei dem der Geschäftsprozesses lediglich implizit aus den Anweisungen für die Bearbeitung von Anwendungsaufgaben hervorgeht, macht ein Workflow-Management-System den Geschäftsprozess als Prozessdefinition explizit. WfMS konzentrieren sich dabei ausschließlich auf Management-Funktionen. Sie führen also keine Prozessaktivitäten aus, sondern sorgen dafür, dass Aufgaben zum richtigen Zeitpunkt an geeignete Ressourcen delegiert werden. Als Workflow-basiertes AS oder *Workflow-System* wird dann ein WfMS verstanden, das für die Prozesse und Ressourcen eines spezifischen Anwendungsbereichs konfiguriert wurde.

In den letzten ca. 20 Jahren sind eine Vielzahl entsprechender kommerzieller und wissenschaftlicher WfMS mit zum Teil recht unterschiedlichen Eigenschaften entwickelt worden.[80] Deren konkrete Workflow Management-Funktionen zeigen dabei in der Regel Unterschiede in Art und Umfang. Durch diese Vielfalt wird das Verständnis und die Einordnung bestehender Ansätze sowie die Interoperabilität verschiedener Produkte erschwert. Um dem entgegenzuwirken, sind Workflow-Modelle und -Spezifikationssprachen sowie die Funktionen, Komponenten und Schnittstellen von WfMS von verschiedenen Gremien und Konsortien standardisiert worden.

Der wohl wichtigste Ansatz ist das *Workflow Reference Model (WfMC-RM)* [Hol95] (vgl. Abb. 3.18) der *Workflow Management Coalition (WfMC)*, in der die überwiegende Zahl der Hersteller organisiert ist.[81] Das WfMC-RM definiert eine generische Architektur für WfMS. Dabei legt es insgesamt fünf allgemein anerkannte Funktionsbereiche als Komponenten fest. Die Beziehungen zwischen den Komponenten sind durch ebenfalls fünf API beschrieben, die in diesem Zusammenhang als Workflow Application Program Interfaces (WAPI) bezeichnet werden.

[80] Eine Übersicht findet sich z. B. bei van der Aalst et al. [AH02, S. 171 ff.].

[81] Für Informationen zu Aktivitäten und weiteren Standards der WfMC siehe [WfM04].

Der erste Funktionsbereich eines WfMS dient der *Prozessmodellierung* (im WfMC-RM als *Prozessdefinition* bezeichnet). Das Ziel besteht darin, die im Rahmen der betrieblichen Prozessgestaltung identifizierten Geschäftsprozesse als Prozessdefinitionen abzubilden. Entsprechende WfMS-Teilkomponenten schließen neben einem Werkzeug zur *Definition* von Prozessen meist auch noch solche zur *Klassifikation* von Ressourcen und *Analyse* von Prozessdefinitionen ein. Die allgemeine Grundlage der Modellierung wird durch ein konzeptionelles *Workflow-Metamodell* gelegt: Dieses bestimmt u. a. die Arten von Modellelementen, aus denen Workflows aufgebaut werden können (z. B. Aktivitäten, Transitionen etc.). Die Beschreibung eines Workflow als Prozessdefinition erfolgt dann mithilfe einer *Workflow-Sprache*. Im WfMC-Referenzmodell ist dazu ein generisches Workflow-Metamodell mitsamt der Spezifikationssprache XPDL [WfM02] standardisiert. *Workflow-Definitionswerkzeuge* unterstützen meist in Form eines grafischen Editors den Entwurf und führen zu Prozessdefinitionen. *Klassifikationswerkzeuge für Ressourcen* erlauben die Modellierung von Ressourcen-Klassen oder auch Rollen und organisatorischen Einheiten. *Analysewerkzeuge* untersuchen Prozessdefinitionen in qualitativer und quantitativer Hinsicht auf ihre Effektivität und Effizienz. Dabei kann umso mehr über die Eigenschaften eines Workflow festgestellt werden, je genauer die Semantik der Workflow-Sprache im Metamodell festgelegt wurde. Basiert die Sprache z. B. auf Petrinetzen,[82] können Prozessdefinitionen durch strukturelle Analyse u. a. auf so wichtige Eigenschaften wie Sicherheit und Lebendigkeit überprüft werden.[83]. Bei weniger formalen Ansätzen oder quantitativer Analyse können oft Aussagen über Prozesseigenschaften auf Basis von Simulation gemacht werden.

Ein weiterer Funktionsbereich beinhaltet die Implementierung und Ausführung von Workflows. Dies wird primär durch eine als *Prozessbearbeitungsservice (Workflow Enactment Service)* bezeichnete Komponente übernommen. Dabei handelt es sich um ein Laufzeitsystem, das aus einer oder mehreren *Workflow Engine(s)* besteht. Workflow Engines werden oft als das „Herzstück“ von WfMS betrachtet. Sie treten in Aktion, wenn ein vorher bestimmtes *Ereignis*, das einen Geschäftsvorfall signalisiert (z. B. der Empfang einer Kundenanfrage) erkannt wird. In diesem Fall wird die Prozessdefinition des entsprechenden Geschäftsprozesses herangezogen und daraus eine Prozessinstanz bzw. ein Fall konstruiert. Dabei muss es möglich sein, mehrere (viele) Instanzen der gleichen Prozessdefinition parallel zu bearbeiten. Aus Gründen der Robustheit und Skalierbarkeit kommen dazu ggf. mehrere Engines zum Einsatz. Die wesentlichen Aufgaben bestehen dann zum einen in der Festlegung von Terminen für die Ausführung von Aktivitäten und zum anderen in der Auswahl geeigneter Ressourcen. Ausgehend vom Start-Knoten werden die Transitionen des Prozesses verfolgt und jeweils diejenigen Knoten bestimmt, die als nächstes zu aktivieren sind. Bei Erreichen eines Ablaufknotens werden dessen Bedingungen überprüft und die

[82] Für Petrinetz-basierte Workflow-Spezifikation siehe z. B. *WF-nets* [AH02, S.107].
[83] Vgl. z. B. Marinescu [Mar02, S. 135 ff.] oder Van der Aalst et al. [AH02, S.107].

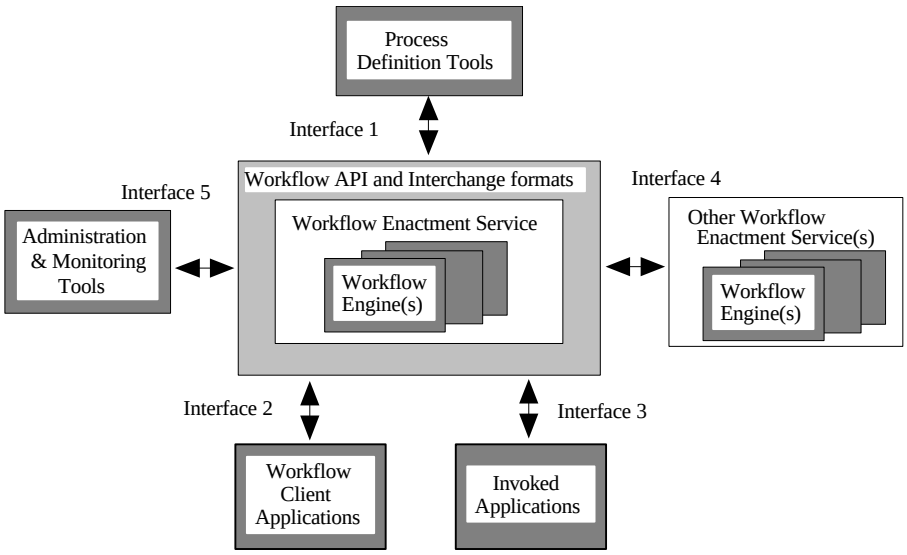

Abbildung 3.18.: Workflow Reference Model der WfMC [Hol95]

weiterführenden Transitionen ermittelt. Bei Erreichen eines Arbeitsknotens beginnt zunächst die meist Regel-basierte Auswahl einer geeigneten Ressource. Mit dieser tritt die Engine dann in der Folge zwecks Delegation der Aktivität in Verbindung.

Verschiedene Möglichkeiten zur Anbindung von Ressourcen bilden wiederum eigenständige Funktionsbereiche/Komponenten der Architektur. Je nachdem, ob es sich bei den Ressourcen um Mitarbeiter oder AS handelt, werden dabei *Workflow-Clients* und *aufgerufene Anwendungen* unterschieden. Workflow-Clients realisieren die Interaktion mit Mitarbeitern in Form eines Eingangskorbs für Elementaraufgaben. Bei aufgerufenen Anwendungen handelt es sich hingegen um eigenständige AS, die nicht selbst Teil des WfMS sind, aber mit dem Prozessbearbeitungsservice per WAPI in Verbindung stehen. Aus dessen Sicht sind beide Fälle in ähnlicher Weise durch einen Nachrichtenaustausch mit den Ressourcen gekennzeichnet. Grundsätzlich sind dabei *Push oder Pull Modelle* möglich, bei denen die Ressourcen neue Aufgaben entweder aktiv anfordern (Pull) oder passiv angewiesen bekommen (Push).

Nach der Bearbeitung einiger Prozessinstanzen ist es meist im Interesse des betrieblichen Managements, deren Ablauf im nachhinein zu untersuchen. Dies fällt in den Funktionsbereich von *Workflow Monitoring-Komponenten*. Entsprechende Werkzeuge sind über ein WAPI an den Prozessbearbeitungsservice angebunden. Von diesem beziehen sie Metadaten über den Verlauf von Prozessinstanzen. Im nachhinein können mit den meist DB-basierten Werkzeugen verschiedene Analysen durchgeführt werden, die deutliche Parallelen zum Data-Mining/-Warehousing aufweisen. Das WfMC-RM ordnet demselben Funktionsbereich des Weiteren die *Workflow-Administration* zu. Entsprechende Komponenten erlauben u. a. die Konfiguration des WfMS, die Verwaltung von Ressourceninstanzen und den Eingriff in laufende Prozessinstanzen,

z. B. zum Eingriff bei unvorhergesehenen Problemsituationen. Unter Umständen werden beim Monitoring auch Schwächen entdeckt, die zu einer Änderung der Prozessdefinition Anlass geben. Weitere Gründe für eine Überarbeitung von Workflows sind Änderungen in den Geschäftsprozessen und/oder der technischen Umgebung. Ein WfMS muss daher Möglichkeiten bieten, Workflows in geregelter Weise weiterzuentwickeln. Dies wird in den meisten Fällen für Prozessdefinitionen unterstützt, die dann beim nächsten Geschäftsvorfall in geänderter Form ausgeführt werden. In manchen Fällen ist es darüber hinaus auch für Prozessinstanzen möglich. Dies ist jedoch mit erheblichen Schwierigkeiten verbunden und nur bei wenigen Systemen realisiert [Aal99].

Der letzte Funktionsbereich des WfMC-RM bezieht sich auf *externe Prozessbearbeitungsservices*. Im Gegensatz zu den verteilten Engines eines homogenen WfMS wird hierbei die so genannte *Workflow-Interoperabilität* zwischen verschiedenen potenziell heterogenen WfMS unterstützt. Diese gehören unterschiedlichen *Workflow-Domänen* an, die sich z. B. bei verschiedenen Abteilungen oder Unternehmen ergeben können. Workflow-Interoperabilität spielt in bestimmten Szenarien des B2B und vor allem auch bei VU eine wichtige Rolle, da sich dabei organisationsübergreifende und prozessorientierte AS ergeben. Aus diesem Grund werden generelle Inter-Enterprise Workflow-Systeme (IEWS) und deren VU-spezifische Varianten in der Folge noch etwas eingehender untersucht.

Zunächst sei noch darauf hingewiesen, dass die Entwicklung von Workflow-Systemen deutliche Parallelen zur Entwicklung von Software-Systemen zeigt [Mar02, S. 23 f.]. Auch Workflow-Spezifikationen zeigen mit ihren Kontrollstrukturen, Aufrufen (von Ressourcen) und Datenstrukturen Ähnlichkeiten zu Software-Programmen. Im Falle von Production-Workflows, die auch den meisten der für die vorliegende Arbeit interessanten IEWS zugrunde liegen, verschwimmen die Unterschiede dann noch weiter. Hier kann die Workflow-Entwicklung als „Programming in the large“ oder „Megaprogramming“ aufgefasst werden [ACK+04, S. 87 ff.]. Workflows dienen dabei als Ausdrucksmittel für die Integrationslogik von großen Software-Komponenten oder ganzen AS. Um dieser Rolle gerecht zu werden, müssen die Workflow-Modelle und -Sprachen sowie auch das WfMS einige zusätzliche Aspekte berücksichtigen. Eine wichtige Forderung ist dabei die zuverlässige und transaktionale Ausführung von Workflows. Um Workflows auch bei Ausfällen zuverlässig abschließen zu können, müssen die Engines alle Vorgangssteuerungsdaten persistent speichern und ein Fortführen im letzten registrierten Zustand ermöglichen (*Forward Recovery*). Transaktionalität kann bei den potenziell lange laufenden Prozessen nicht auf klassische Weise durch Sperren von Ressourcen erfolgen. Stattdessen müssen für alle Aktivitäten kompensatorische Maßnahmen vorgesehen und bei Bedarf eingeleitet werden (*Backward Recovery*). Des Weiteren sollte der Workflow Mittel bieten, um Laufzeitfehler und ihre Behebung beim Entwurf zu antizipieren, wie es in vielen Programmiersprachen durch *Exceptions* möglich ist. Ein wichtiges Mittel zur Erkennung solcher Fehler sind u. a. zeitliche Fristen (*Deadlines*) bei den Aufrufen von AS aus einer Workflow

Engine. Entsprechende WfMS werden häufig mit Middleware-Mechanismen wie Message Brokern zu umfassenden prozessorientierten Integrationsplattformen kombiniert (vgl. Abb. 3.15).[84]

3.3.3.2. Inter-Enterprise Workflow-Systeme

Sollen auf Basis von Workflow sehr große Anwendungssysteme integriert werden, dann ist zur Erhaltung einer ausreichenden Qualität der Ausführung (u. a. in Bezug auf Performanz, Verlässlichkeit und Skalierbarkeit) die Verteilung des WfMS geboten. In der WfMC-Architektur ist dies durch multiple Engines innerhalb eines homogenen Prozessbearbeitungsservices berücksichtigt. Derartige Bedingungen finden sich jedoch meist nur innerhalb eines Unternehmens und selbst dort oft nur innerhalb einzelner Abteilungen oder Bereiche. Die Mehrheit kommerzieller WfMS war und ist auf diesen Kontext ausgelegt. Derartige Systeme eignen sich gut, um potenziell komplexe innerbetriebliche Geschäftsprozesse abzubilden und mittels relativ großer AS zu automatisieren.

Wie schon aus ökonomischer Perspektive dargestellt wurde, ist darüber hinaus oftmals eine Automatisierung zwischenbetrieblicher Prozessketten von Interesse.[85] Dies gilt in besonderem Maße für VU: Die flexible Integration von Geschäftsprozessen unterschiedlicher Netzwerkunternehmen unter Verzicht auf zentrale Managementfunktionen basiert zu einem wesentlichen Teil auf deren IV-Integration. Dementsprechend beinhalten die Anforderungen an eine IV-Infrastruktur für VU die Koordination von AS-Komponenten auf Basis gemeinsamer Geschäftsprozesse.[86] Für eine entsprechende Unterstützung der Abbildung und Automatisierung zwischenbetrieblicher Geschäftsprozesse (Distributed Business Processes (DBP)) sind klassische WfMS jedoch nicht gut geeignet [MBB+03]. In diesem Kontext kann die Existenz einer einheitlichen Umgebung aus WfMS und zugehörigen Integrationsplattformen nämlich nicht mehr vorausgesetzt werden. Stattdessen liegen potenziell heterogene Workflow-Domänen vor. Weitere Probleme entstehen aus den speziellen Anforderungen zwischenbetrieblicher Workflow-Systeme: u. a. wird dabei hohe Transparenz, Performanz und Adaptionsfähigkeit verlangt [Aal99]. Die Realisierung entsprechender Inter-Enterprise Workflow-Systeme (IEWS) setzt daher nicht nur die technische Interoperabilität heterogener WfMS voraus. Sie verlangt darüber hinaus problemspezifische Konzepte für DBP und deren Management [Aal99].

Aus technischer Perspektive sieht die WfMC-Architektur die Schnittstelle Nr. 4 zur Workflow-Interoperabilität vor. Die hierin festgelegten Dienste und Austauschformate teilen sich in solche, die den Entwurf und solche, die die Ausführung von

[84] Beispiele für derartige Systeme sind *IBM WebSphere MQ Workflow, Vitria BusinessWare, Tibco BPM, BEA WebLogic Integration* oder *Microsoft BizTalk Orchestration* [ACK+04, S. 82 f.].

[85] Siehe hierzu die Ausführungen zu zwischenbetrieblicher Prozessverkettung (Sektion 2.3.1.2).

[86] Vgl. Tab. 3.2 und Sektion 3.2.2.1.

Workflows betreffen. Für den gemeinsamen Entwurf von Workflow-Prozessen ist im Wesentlichen der Austausch von Workflow-Definitionen vorgesehen. Dazu werden einheitliche Übertragungsformate benötigt. Im Rahmen der WfMC-Standardisierung wird zu diesem Zweck die Spezifikationssprache XPDL vorgeschlagen, die auch (und eigentlich) WfMS-intern für Schnittstelle Nr. 1 verwendet wird. Parallel dazu existiert eine extensive Menge alternativer Standards diverser Gremien und Konsortien.[87] Zur gemeinsamen Ausführung von Workflows visiert das WfMC-RM die Interaktion heterogener Prozessbearbeitungsservices an, wobei verschiedene Arten von Laufzeitinformationen ausgetauscht werden. Interaktionen beinhalten u. a. die Erzeugung neuer Prozessinstanzen und die Abfrage/Änderung von Attributen wie insbesondere dem Prozessstatus, den Vorgangsdaten und den Vorgangssteuerungsdaten. Im Verlauf der Standardisierung wurde schließlich durch die Wf-XML-Spezifikation ein Protokoll zur Workflow-Interoperabilität festgelegt [WfM00].[88]

Die Internet-basierten Workflow Interoperability-Protokolle realisieren eine Schnittstelle, die es erlaubt, WfMS verschiedener Workflow-Domänen technisch zu koppeln. Damit auf dieser Basis ein IEWS entstehen kann, fehlen allerdings noch Konzepte zur Verteilung eines organisationsübergreifenden Workflow-Prozesses auf beteiligte Unternehmen und zur Aufteilung der Managementaufgaben auf deren WfMS. Das WfMC-RM unterscheidet hier verschiedene grundsätzliche Szenarien zur Workflow-Interoperabilität. Diese wurden von van der Aalst aufgegriffen, erweitert und auf ihre Eignung für IEWS untersucht [Aal99]. Die verschiedenen Varianten werden in Tabelle 3.4 aufgelistet, wobei jeweils eine Charakterisierung des Interoperabilitätskonzepts und eine Bewertung der IEWS-Eignung erfolgt. Grundsätzlich kann bei der zwischenbetrieblichen Aufteilung von Workflows zwischen einer horizontalen und einer vertikalen Partitionierung unterschieden werden. Horizontale Partitionierung zergliedert die Workflow-Definition, deren Fragmente dann einzelnen Organisationen zugewiesen werden (Chained Execution, Subcontracting, Loosely Coupled). Bei vertikaler Partitionierung besitzen alle Organisationen die gleiche Workflow-Defi-

[87] Als früher und häufig referenzierter Ansatz kann hier z. B. das *Process Interchange Format (PIF)* genannt werden [LY94]. PIF ist als Grundlage zum automatisierten Austausch von Prozessbeschreibungen zwischen diversen Prozesswerkzeugen ausgelegt. Der Ansatz ist später zum Teil in den NIST-Standard *Process Specification Language (PSL)* [SGT+00] eingeflossen. Neuere Ansätze finden sich im Rahmen diverser B2B-Standards; Bei Bernauer et al. findet sich z. B. ein Überblick verschiedener Spezifikationssprachen im Kontext von ebXML und Web Services [BKK+03b].

[88] Der erste WfMC-Standard zur Interoperabilität war die konzeptionelle *„Interoperability Abstract Specification“*, die durch das *„Interoperability Internet e-mail MIME Binding“* konkretisiert wurde. Die abstrakte Spezifikation wurde durch die *Workflow Management Facility (jointFlow)* in den CORBA-Kontext übertragen und dort durch ein Objektmodell untermauert. Auf dessen Basis wurde eine Teilmenge der Operationen als HTTP-basiertes Internet-Protokoll ausgewählt, das als *Simple Workflow Access Protocol (SWAP)* bezeichnet wird. Diese Evolution der Workflow Interoperability wurde von der WfMC durch *Wf-XML* fortgeführt: dabei handelt es sich um die XML-basierte Erweiterung des SWAP Standards. Eine Übersicht dieser Entwicklungen findet sich z. B. in [HPS+00].

Form	Merkmale	IEWS-Eignung
Capacity Sharing	Es findet keine Partitionierung des Workflows statt. Ein zentrales WfMS bearbeitet alle Fälle und teilt deren Aktivitätsinstanzen an Ressourcen in verschiedenen Organisationen zu.	Der Ansatz ist ungeeignet, da die Koordination zentralisiert ist.
Chained Execution	Workflow-Definitionen werden partitioniert und an verschiedene Organisationen verteilt. Deren WfMS arbeiten ihre Workflow-Partition jeweils komplett ab und übergeben dann die Kontrolle an die nächste Organisation.	Der Ansatz ist ungeeignet, da keine wechselseitige Konversation der Organisationen möglich ist.
Subcontracting	Workflow-Definitionen werden partitioniert und an verschiedene Organisationen verteilt. Das WfMS der leitenden Organisation bearbeitet einen übergeordneten Kontrollprozess. Einzelne Aufgaben sind als Sub-Prozesse realisiert und werden von WfMS einzelner untergeordneter Organisationen bearbeitet.	Der Ansatz ist geeignet, setzt aber hierarchische Beziehungen zwischen den Organisationen voraus.
Case Transfer	Alle Organisationen verfügen über die gleiche Workflow-Definition. Bei neuen Fällen beginnt eine der Organisationen mit der Bearbeitung. Unter vorherbestimmten Bedingungen wird der gesamte Fall zur Laufzeit an jeweils eine andere Organisation übergeben.	Der Ansatz ist gut geeignet schränkt aber die Nebenläufigkeit ein.
Loosely Coupled	Jede Organisation realisiert einen eigenständigen Workflow-Prozess; nur die öffentlichen Prozesse sind gegenseitig bekannt. Die Bearbeitung der einzelnen Prozesse erfolgt autonom und vollkommen nebenläufig. An beliebigen Stellen finden wechselseitige Aufrufe statt.	Der Ansatz ist auf Grund seiner Flexibilität gut geeignet. Der hohe Grad nebenläufiger Bearbeitung macht den Entwurf fehleranfällig.

Tabelle 3.4.: Formen der Workflow-Interoperation nach [Aal99]

nition und verteilen die Workflow-Instanzen untereinander (Case Transfer). Unter Berücksichtigung der Anforderungen von IEWS im B2B-Kontext zeigen sich besonders die Interoperabilitätsformen „Loosely Coupled“ und „Case Transfer“ als vorteilhaft [Aal99].

Die lose Kopplung von Workflows ist als Interoperabilitätskonzept für IEWS im B2B-Kontext weit verbreitet. Derartige Lösungen zeichnen sich durch einen hohen Grad an Parallelität, Performanz und Flexibilität aus. Einzelne Fälle können in mehreren Organisationen gleichzeitig bearbeitet werden, was Skalierbarkeit und Performanz der Bearbeitung zugute kommt. Änderungen der Workflow-Definition betreffen hingegen immer nur einzelne Organisationen und sind daher leicht zu koordinieren. Auf der anderen Seite sind die nebenläufigen Prozesse wenig transparent und ihr Entwurf ist fehleranfällig. Zudem erhöht die Kommunikation zur Synchronisation zwischen Prozessinstanzen den Aufwand zur Laufzeit. Die einfachste Form des

Konzepts legt den DBP offen.[89] Neuere Formen kapseln jedoch in der Regel die internen Prozesse einzelner Organisationen und beziehen sich im Rahmen eines öffentlichen Prozesses ausschließlich auf deren Schnittstellen.[90] Im Gegensatz zur losen Kopplung ist der Transfer von Fällen wenig verbreitet.[91] Hierbei wird der DBP als einzelne Workflow-Definition entworfen und jeder Fall jederzeit mitsamt allen Informationen bei genau einer Organisation bearbeitet. Dadurch wird zum einen Transparenz gewährleistet. Zum anderen werden die meisten Fehlerquellen nebenläufiger Bearbeitung ausgeschlossen. Dementsprechend ist der Entwurf hier einfacher. Prozessdefinitionen werden jedoch von allen Unternehmen geteilt. Sie sind daher von jedem Teilnehmer komplett einsehbar und müssen zudem bei jeder Änderung mit entsprechendem Koordinationsaufwand an alle verteilt werden. Hinzu kommt, dass sich die beschränkte Parallelität negativ auf die Performanz auswirkt.

Auf Grund der Defizite kommerzieller WfMS sind generische Ansätze zur Unterstützung von DBP mittels IEWS vor allem in verschiedenen Forschungsprojekten untersucht worden. Einige Projekte/Ansätze sollen hier kurz angesprochen werden:[92]

- **Mentor** [MWW+98, WGR+00, SGW01] und Mentor-lite sind umfassende WfMS, die an der Universität des Saarlandes Saarbrücken und am Max-Planck-Institut für Informatik von Weikum u. a. entwickelt wurden. Hierbei wird die Unterstützung einer organisationsübergreifend verteilten Ausführung von Workflows fokussiert. Er basiert auf einer horizontalen Partitionierung von Workflow-Definitionen, die in Form von Zustandsdiagrammen gegeben sind. Jede Organisation verfügt über einen eigenen Prozessbearbeitungsservice. Dessen Workflow Engine besteht aus einem *Workflow-Interpreter* sowie einem *Communication-* und einem *Log-Manager*. Der *Workflow-Interpreter* realisiert die Ausführung von Workflow-Partitionen. Der *Communication Manager* dient dem Austausch von Nachrichten zur Synchronisation zwischen Prozessbearbeitungsservices. Auf Basis des TP-Monitors Tuxedo können Nachrichten als Transaktionen zusammengefasst werden. Der *Log Manager* erlaubt die Dokumentation und das Recovery von Workflow-Instanzen in allen beteiligten Organisationen.

[89] Vgl. z. B. den Ansatz im **Mentor**-Projekt [MWW+98].

[90] Generelle Überlegungen hierzu finden sich z. B. in [Bus01, Aal99, DHL01]. Vgl. auch Sektion 3.3.2.4.

[91] Ein Beispiel für diese Form findet sich im umfangreichen **Sagitta**-Projekt [AH02, S. 243 ff.].

[92] Im allgemeineren Kontext sollen noch zwei weitere Ansätze Erwähnung finden, die das Workflow-Grundkonzept in Bezug auf Interaktion bzw. Kooperation erweitern: Das **Proclet**-Modell/-Framework [ABE+01] von Barthelmess, Ellis, van der Aalst, u. a. unterstützt ein innovatives Konzept zur Strukturierung von Workflow-Prozessen, die nicht durch einen einzelnen Fall abzubilden sind. Stattdessen werden interagierende Proclets eingeführt: leichtgewichtige Workflows, die individuell instanziert werden können. Proclets kommunizieren dabei über Channels und tauschen strukturierte Nachrichten (Performatives) aus. **COO**[GCP+00] ist ein Workflow-Modell von Godart, Perrin, u. a. für kooperative Prozesse/DBP, die insbesondere die Koordination *kooperativer Aktivitäten* erlauben, bei denen verschiedene Ressourcen in einem Transaktionskontext Zwischenergebnisse austauschen können.

- **CPM** (Collaborative Process Management) [CH00, CH02] ist das Leitmotiv eines experimentellen Framework, das in den HP und Commerce One Labs von Chen und Hsu entwickelt wurde. CPM basiert auf einer generellen Trennung von internen Prozessen (*Business Processes*) und externen Prozessen (*Conversations*). Das CPM-Framework dient zur Realisierung von IEWS für die B2B-Integration. Dazu definiert es eine abstrakte Architektur, die aus *Conversation*, *Process* und *Action Manager* besteht. *Conversation* und *Process Manager* dienen der Automatisierung von externen und internen Prozessen, wobei die Unterschiede dieser Aufgaben hervorgehoben werden. Der *Action Manager* ist hingegen für die Steuerung und Kontrolle interner Aktivitäten zuständig. Der wesentliche Beitrag von CPM liegt in der Untersuchung verschiedener Umsetzungsmöglichkeiten des Framework auf Basis von WfMS. Dabei werden zum einen Lösungen berücksichtigt, die zur Umsetzung aller Komponenten WfMS einsetzen, was zu Varianten des Instanzen-Transfers führt. Zum anderen wird der *Conversation Manager* exklusiv auf Basis von B2B-Standards wie ebXML-BPSS realisiert, was der losen Kopplung entspricht. Insgesamt liegt der Schwerpunkt der Arbeit auf praktisch-technischen Aspekten der Workflow-Kopplung und Implementierung von B2B-IEWS auf Basis vorhandener WfMS. Derartige Fragestellungen besitzen eine erhebliche praktische Relevanz und sind auch von diversen anderen Autoren untersucht worden, die hier nicht im Einzelnen erwähnt werden können.[93]

- **Meteor** (Managing End-To-End OpeRations) ist ein Projekt der Universität Georgia, in dem seit 1991 von Sheth, Kochut, Miller u. a. WfMS erforscht und entwickelt werden [LSD05]. In dem Teilprojekt **Meteor**$_2$ wurden insbesondere WfMS fokussiert, die die Koordination von Mitarbeitern und automatisierten Aufgaben in großen, zwischenbetrieblichen, heterogenen IT-Umgebungen auf Basis von CORBA und Java Middleware unterstützen [KS95]. Das System beinhaltet u. a. ein GUI-Toolkit für Entwurf, Simulation und Monitoring von Workflows sowie die Codegenerierung für verteilte Workflow-Systeme aus grafischen Entwürfen, Terminplanung, EDI-Unterstützung, Fehlerbehandlung und Recovery auf Basis von Transaktionen und ein Sicherheitskonzept in Form von Rollen-basierter Zugriffskontrolle. Neben einem produktreifen System[94] führte Meteor zu Forschungsergebnissen u. a. bzgl. Workflow-Adaption, kombinierter Koordination und Kollaboration sowie zwischenbetrieblicher Workflows.

- Das **WebFlow**-System wurde an der Universität Leipzig u. a. von Rahm und Greiner entwickelt. Es erlaubt im Wesentlichen die Definition und Ausführung kooperativer Workflows auf Basis einer Mediator-Architektur [GR03]. In Workflows können Web Services über die Grenzen einer Organisation hinweg

[93] Siehe z. B. Sayal, Casati et al. [SCD+02].

[94] Meteor wurde von Infocosm vermarktet.

eingebunden werden. Der Schwerpunkt des Systems liegt auf der Spezifikation und Überwachung qualitativer Ausführungsbedingungen. Ausnahmen werden regelbasiert erkannt und es besteht die Möglichkeit zur Reaktion durch Adaption des Workflow.

- Der **P2P**-Ansatz (Public-to-Private) der Technischen Universität Eindhoven geht auf van der Aalst und Weske zurück [AW01]. Der Ansatz beinhaltet eine Methode zur Entwicklung lose gekoppelter IEWS. Er basiert auf einer Erweiterung von Petrinetzen (Workflow-Netze) und betont die Verifikation von Systemeigenschaften unter Berücksichtigung der autonomen Entwicklung privater Teilprozesse. Die Methode beginnt mit der Erstellung einer öffentlichen Workflow-Definition. Diese wird dann partitioniert und den einzelnen Organisationen zugeteilt. Bei der Realisierung der Partitionen können die einzelnen Organisationen in autonomer Weise individuelle Erweiterungen vornehmen. Das konsistente Verhalten des Gesamtprozesses wird durch einen Vererbungsbegriff für Workflow-Netze gewährleistet: Es wird sichergestellt, dass private Prozesse stets Sub-Klassen ihrer ursprünglichen Partition bleiben, d. h. dass sie deren Verhalten erben.

Bei den diversen Ansätzen können laut Baïna sechs verschiedene Formen der Prozesskopplung unterschieden werden [BBG03]: (1) Bei *Nachrichten-orientierter Prozesskommunikation* erfolgt ein asynchroner Austausch typisierter Nachrichten mittels diverser Paradigmen (Push, Pull, Subscribe etc.) zwischen Workflow-Prozessen. (2) *Ereignis-basierte Synchronisation von Prozessen* erweitert die Nachrichtenkommunikation mit Sprachen zur Koordination von Ereignissen und Algebras zur Synchronisation überlappender Prozesse. (3) Bei *Interoperabilität von Prozessdaten und -schnittstellen* werden einheitliche Frameworks für operationale Schnittstellen und Datenformate zugrunde gelegt. (4) Der *kontrollierte, nebenläufige Zugriff auf Prozessdaten* erweitert die Interoperabilität von Daten durch Zugriffskontrolle auf gemeinsame Workflow-Daten. (5) Bei *transaktionaler Kontrolle von Prozessen* werden Workflow-Prozesse als Transaktionen angesehen und entsprechende Modelle für die Workflow-Ausführung und das Datenmanagement bereitgestellt. (6) *Service-orient. Prozesskopplung* abstrahiert Prozesse als Services, deren eindeutige Spezifikation die Basis der Prozesskopplung bildet. Im weiteren Verlauf wird insbesondere die Service-orient. Prozesskopplung eine wesentliche Rolle spielen. Zuvor soll aber noch eine generelle Betrachtung einiger Ansätze für IEWS im Kontext von VU erfolgen.

IEWS im virt. Unternehmen Die generische Koordinationsunterstützung zwischen Netzwerkunternehmen sowie die spezifische Fähigkeit zur Definition und Ausführung von DBP in der operativen Phase sind wesentliche Anforderungen an die IV-Infrastruktur zur zwischenbetrieblichen Integration in virt. Unternehmen.[95] Für beide Aspekte

[95] Vgl. Sektion 3.2.2.1.

bieten sich u. a. Workflow-Techniken an. Dies ist bereits in einigen Forschungsprojekten untersucht worden. Dabei handelt es sich meistens um Erweiterungen des WfMC-RM in Bezug auf die Handhabung zwischenbetrieblicher Workflows. In diesem Rahmen wurden erste Mechanismen u. a. für Ausnahmenbehandlung, mehrstufige Koordination und deren Beziehung zu verschiedenen koordinativen Rollen von VU-Teilnehmern sowie flexible Workflow-Modelle zur Unterstützung wenig strukturierter Prozesse entwickelt [CMA03, S.668]. Im Folgenden werden einige wesentliche Ansätze kurz charakterisiert.[96]

- Das Projekt **WISE** (Workflow-based Internet SErvices), das unter Schek, Alonso, Plattner u. a. an der ETH Zürich durchgeführt wurde, zielt auf eine Infrastruktur zur ganzheitlichen Unterstützung zwischenbetrieblicher Geschäftsprozesse in VU [AFL+99, LAS+00, LSA+01]. Die Architektur teilt sich in die vier Komponenten *Definition, Enactment, Monitoring* und *Coordination*. Bei der Definition werden virtuelle Geschäftsprozesse eingeführt („virtual business processes“, VBP). VBP bauen auf Komponenten eines Katalogs auf. In diesen Katalog bringen die Partner einer Trading Community (TC) ihre spezifischen „Services“ ein. Durch Definition eines VBP formieren sich die Unternehmen der TC zu einem virt. Unternehmen. Die Enactment-Komponente übersetzt den VBP in ein ausführbares Format und kontrolliert den entsprechenden Aufruf von Services der TC. Für die Workflow-Ausführung in WISE wird der Prozess-Kernel **OPERA**[97] eingesetzt. Die Monitoring-Komponente vollzieht den Verlauf des VBP nach und nutzt die Informationen zur Optimierung. Durch die *Coordination*-Komponente werden Führungsinformationssystem (FIS)-Mechanismen wie z. B. Konferenzen der TC unterstützt.

- Das Esprit-Projekt **CrossFlow** (Cross-Organizational Workflow Support in Virtual Enterprises) [GAH+00] wurde u. a. von Aberer (GMD), Grefen (Universität Twente) und Hoffner (IBM Zurich Research Laboratory) geleitet. Der dort entwickelte Ansatz zielt darauf ab, den Einsatz von Workflows innerhalb dynamisch gebildeter virt. Unternehmen auf höherer Ebene zu unterstützen. Hierzu wird von den Details technischer Services abstrahiert und erweiterte Kooperationsunterstützung angeboten. VU werden dynamisch durch vertragsbasierten Abgleich zwischen Anbietern und Konsumenten technischer Services gebildet. CrossFlow definiert Interaktionen zwischen Organisationen dabei nicht auf Basis ihrer WfMS, sondern auf einer darüberliegenden Abstraktionsebene. Die Realisierung von VU als Kooperation auf Basis dynamischer Partnerschaften bedient sich

[96] Weitere Beispiele, die hier noch kurz erwähnt werden sollen, sind die beiden VOSTER-Projekte **PRODNET** von Afsarmanesh, Camarinha-Matos u. a. [CMAO01] (VU-Infrastruktur mit Workflow-basierter Koordination und DBP-Management) und **ISTforCE** von Katranuschkov, Scherer, Turk u. a. [KST01] (Simultanes Management multipler VU-Projekte durch Integration ihrer unterschiedlichen Workflows).

[97] Ausführlich beschrieben in der Dissertation von Hagen [Hag99].

primär des spezifischen Konzepts des Vertrags [KGV00]. Solche Verträge bilden die Grundlage zur Spezifikation von Interaktionen und dienen dazu, Services zu veröffentlichen. Dafür nutzen Unternehmen einen Vertrags-Manager, um ein Vertrags-Template an einen Broker bzw. eine so genannte Matchmaking Engine zu senden. Vertrags-Templates kommen andersherum auch zur Ausgliederung von Services zum Einsatz. Dann sendet ein nachfragendes Unternehmen dieses Template an einen Broker, um nach passenden Anbietern zu suchen. Bei Übereinstimmung von Anforderungen und Angebot wird ein elektronischer Vertrag durch Ausfüllen des Templates aufgesetzt. Basierend auf den Vertragsdetails wird dann eine dynamische Vertrags- und Diensterbringungsinfrastruktur aufgesetzt. Die symmetrische Infrastruktur auf Anbieter- und Konsumentenseite beinhaltet Proxy Gateways, welche die Interaktionen kontrollieren. Sobald der Vertrag erfüllt wurde, können diese dynamisch erzeugten Module wieder entfernt werden.

- Auch die **CMI** (Collaboration Management Infrastructure) von Baker, Georgakopoulos, Schuster u. a. beinhaltet eine Architektur für unternehmensübergreifende Workflows [SGC+00, GSC+99, SBC+00], die speziell für VU ausgelegt wurde. Die CMI-Architektur besteht im Wesentlichen aus *Core*, *Coordination* und *Awareness Engine*. Die *Core Engine* stellt grundlegende Abstraktionen und Primitive zur Verfügung. Diese ermöglichen die Definition von Ressourcen, Rollen und generischen Automaten. Das zugrunde liegende *Koordinationsmodell* erweitert die traditionellen Workflow-Koordinationsprimitiven um Konzepte wie z. B. *Placeholder*. Dieses Konzept ermöglicht die dynamische Etablierung von Geschäftsbeziehungen. Eine Placeholder-Aktivität wird dabei zur Laufzeit durch eine konkrete Aktivität mit gleichen Eingangs- und Ausgangsparametern ersetzt. Eine Auswahlstrategie bestimmt dabei diejenige konkrete Aktivität, die den Platzhalter ersetzen soll. Falls mehrere Anbieter Implementierungen für eine Aktivität anbieten, kann mithilfe eines Brokers diejenige ausgewählt werden, die in Bezug auf ihre Qualitätsmerkmale das Optimum aufweist. Das von CMI eingeführte *Awareness Model* erlaubt die Erfassung von Informationen über die Rollen und Zustände teilnehmender Prozess-Partner. Dies erfolgt auf Basis so genannter *Awareness Specifications* und ermöglicht u. a. die korrekte Zuordnung von konkreter Aktivität und *Placeholder*. Zusammengenommen erlauben die Methoden und Werkzeuge der CMI-Architektur eine Komposition von Services verschiedener Unternehmen. Dabei entsteht z. B. durch Festlegung gemeinsamer Austauschformate und Kommunikationsprotokolle eine relativ feste Kopplung der Partner. Deren Interoperabilität wird u. a. auch durch OO-Proxies unterstützt, die den einheitlichen Zugriff auf Informationsquellen ermöglichen. Zur Modellierung von Services beinhaltet CMI anwendungsspezifische endliche Automaten und Operationen. Dies ermöglicht das selektive Monitoring von Zustandsänderungen in externen Services. Ferner wird durch

spezifische Operationen wie *Optional* und *Inhibitor* auch die Adaptierbarkeit unterstützt, um unvorhersehbare Situationen zu berücksichtigen.

Die gezeigten Arbeiten zur Unterstützung von VU durch IEWS repräsentieren Pioniere dieser Technik, die allesamt wesentliche Akzente gesetzt haben. Schon diese Ansätze können in Bezug auf ihre Abstraktionsmittel für betriebliche Anwendungsfunktionen als *Service-basiert* gelten. Im weiteren Verlauf der Entwicklung hat vor allem der technische Trend zur Service-orient. zu wesentlichen neuen Impulsen geführt. Entsprechende Arbeiten, denen ein unmittelbar *Service-orient.* Ansatz zugrunde liegt oder die auf der Komposition von Web Services basieren, werden in Sektion 3.4.3 behandelt. Zuvor wird in den nächsten Sektionen das dabei zugrunde liegende SOC-Paradigma eingeführt.

3.4. SOC-Techniken als prozessorientierte VDU-Middleware

In der vorangegangenen Sektion wurden im Hinblick auf die Fragestellung der vorliegenden Arbeit verschiedene grundlegende Techniken einer integrativen IT-Basisinfrastruktur für VDU diskutiert. Konkret betraf dies – im konzeptionellen Gesamtrahmen von Enterprise Architecture Frameworks – vor allem (a) die Systemintegration durch Middleware und (b) die Prozessintegration mittels Workflow-Techniken. Dabei zeigte sich, dass Middleware nicht nur geeignete Abstraktionen und Mechanismen bietet, um moderne verteilte AS zu konstruieren, sondern heute auch die Integration heterogener Systemlandschaften erlaubt. Daneben fand sich mit Workflow-Management-Systemen eine Technik zur Erweiterung von Middleware in Bezug auf die Unterstützung von Prozessintegration; d. h. Entwurf und Kontrolle von Geschäfts- und Software-Prozessen zur Koordination von Systemen und Mitarbeitern. Allerdings wurden auch die Defizite konventioneller Middleware- und Workflow-Techniken deutlich, die sich in Bezug auf die erweiterte Problemstellung im zwischenbetrieblichen Kontext ergeben. Hier besteht das Problem darin, eine gemeinsame Basis zu etablieren. Es zeigte sich, dass weder die Zentralisierung einer gemeinsamen Infrastruktur noch die Kopplung verteilter, aber streng homogener Infrastruktur-Produkte eine plausible Lösung bieten.[98]

An dieser Stelle versprechen die Konzepte des *Service Oriented Computing (SOC)* einen Ausweg.[99] Dabei spielen *Web Services (WS)* die zentrale Rolle: Dies sind autonome, plattformunabhängige Software-Einheiten, die über das Web in automatisierter Weise gefunden, analysiert, zugegriffen und als Komponenten massiv verteilter Anwendungssysteme verwendet werden können [Moh02]. Der wesentliche Beitrag von

[98] Vgl. Sektion 3.3.2.4, S. 131.

[99] Eine kurze, aber prägnante und äußerst treffende Einführung in dieses weite Feld findet sich bei Papazoglou und Georgakopoulos [PG03]. Der Überblick von Pilioura und Tsalgatidou fokussiert hingegen die Grundaspekte [PT01]. Forschungsarbeiten zum allg. SOC-Paradigma finden sich z. B. in [ZPL+04].

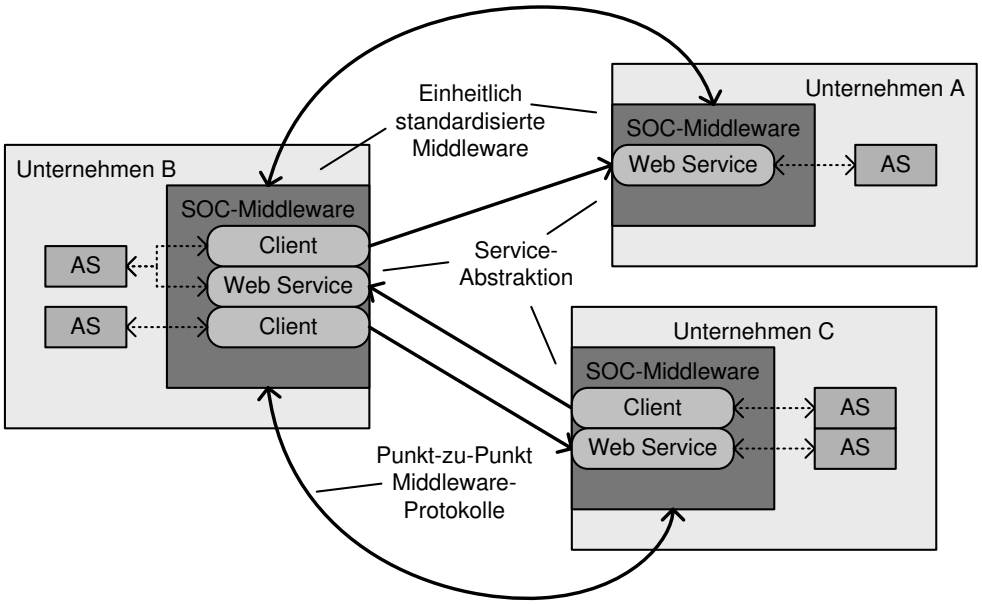

Abbildung 3.19.: Faktoren SOC-basierter zwischenbetrieblicher Integration

Web Services zur zwischenbetrieblichen Systemintegration liegt in ihrer Funktion als Zugangspunkt zu unternehmensinternen Anwendungssystemen [CNW01]. WS liefern dabei ein einheitliches Komponentenmodell und fungieren quasi als Wrapper interner Systeme. Sie haben insofern gewisse Ähnlichkeiten mit bekannten verteilten Komponentenmodellen auf Basis von Remote Procedure Call oder Remote Method Invocation. Im Gegensatz zu konventioneller Middleware begünstigt Service Oriented Computing jedoch die zwischenbetriebliche Integration u. a. durch drei zusätzliche fördernde Faktoren: das *Service-Paradigma*, *organisationsübergreifende Middleware-Protokolle* und *Standardisierung* [ACK+04, S. 131 ff.] (vgl. Abb. 3.19).

Zunächst liefert das Service-Paradigma eine für den zwischenbetrieblichen Kontext passende Abstraktion: Services bilden ein Bindeglied zwischen Anwendungs- und Technikebene. Sie bieten über eine explizite Schnittstelle eine Anwendungsfunktion mit klar umrissenen funktionalen und nicht-funktionalen Eigenschaften [OEH02]. Dabei ist der Service aus technischer Sicht über das Web leicht und weitgehend automatisiert zu finden und zuzugreifen. Seine Realisierung ist jedoch sowohl in Bezug auf die Anwendungslogik als auch in Bezug auf die technische Realisierung durch Kapselung verborgen. Aus der Perspektive einer Service-orient. Architektur sind nun alle Geschäftsfunktionen Services. Dadurch wird nicht nur Interoperabilität auf allen Ebenen sichergestellt, sondern auch eine lose Kopplung an innerbetriebliche Organisationsstrukturen erreicht. Die Verwendung autonomer Services als Bausteine zwischenbetrieblich verteilter AS führt zu erhöhter Flexibilität und Modularität der Systeme und fördert die Komposition und Wiederverwendung von Komponenten.

Des Weiteren verfolgt SOC eine auf den zwischenbetrieblichen Kontext ausgerichtete Strategie zur Realisierung voll verteilter Middleware-Mechanismen. Dabei wird weder Zentralisierung noch Homogenität von Middleware-Plattformen vorausgesetzt. Zudem werden die spezifischen Anforderungen zwischenbetrieblicher Interaktionen berücksichtigt. In diesem Sinne werden für spezifische Abstraktionen im Zusammenhang mit Web Services wie z. B. Kompositionen, langlebigen Transaktionen und Sicherheitsmechanismen einheitliche Protokolle neu entworfen. Dies geschieht derart, dass einheitliche Service-orient. Middleware-Plattformen verschiedener Unternehmen in Punkt-zu-Punkt Manier interagieren können. Transaktionen werden z. B. nicht durch 2PC realisiert, da hierfür ein zentraler Koordinator nötig wäre und Ressourcen untragbar lange gesperrt werden könnten [TP02, S. 156]. Stattdessen wurden spezifische Protokolle entworfen, bei denen die Transaktionsmechanismen vollkommen dezentralisiert sind und keine Ressourcen gesperrt werden.

Schließlich ist die Tatsache von Bedeutung, dass viele SOC-Techniken frühzeitig offen gelegt und standardisiert wurden. Dies ist zwar auch bei anderen Middleware-Techniken wie z. B. im Falle von CORBA geschehen, hier jedoch mit deutlich geringerer Akzeptanz. SOC-Standards wurden nicht nur von den im Web-Kontext maßgeblichen Standardisierungsorganisationen wie W3C und OASIS getragen,[100] sie wurden auch in bislang nicht gekannter Einhelligkeit von den größten Unternehmen und Konsortien der IT-Industrie übernommen und propagiert.[101]

Wegen der genannten Eigenschaften empfehlen sich SOC-Techniken auch und gerade als Grundlage für die Basisinfrastruktur im VU.[102] Insbesondere besticht die Idee, mit SOC-Techniken das Middleware-Konzept von einzelnen Unternehmen auf Unternehmensnetzwerke auszuweiten. Dadurch können viele nützliche Abstraktionen und Mechanismen wie Transaktionen, Sicherheitsmechanismen sowie auch Workflows, die bislang nur im abgegrenzten Unternehmensrahmen verfügbar waren, im erweiterten Kontext eines virt. Unternehmens nutzbar gemacht werden. In diesem Zusammenhang ergeben sich durch die Übertragung von Workflow-Konzepten in den SOC-Kontext neue Möglichkeiten der Prozessunterstützung im VU. Dies ist das wesentliche Argument zur Berücksichtigung von SOC-Techniken in der vorliegen-

[100] Zur Entwicklung und Förderung der Web Service-Standards wurden innerhalb des World Wide Web Consortiums (W3C) [W3C02d] die *Web Service Activity* [W3C02a] sowie diverse Arbeitsgruppen, z. B. für die Beschreibung [W3C02b] und die Architektur [W3C02c] von WS, ins Leben gerufen. Die Organization for the Advancement of Structured Information Standards (OASIS) [OAS05b] ist an der Konzeption der Service-orient. Architektur sowie an der Entwicklung diverser SOC-Protokolle wie BPEL und UDDI beteiligt [OAS05a].

[101] Die grundlegenden WS-Standards des W3C wurden z. B. sowohl von Microsoft im Rahmen von .Net [Mic05] als auch von IBM im Rahmen von WebSphere [IBM05] übernommen und gefördert. In diesem Zusammenhang bemüht sich die Web Service Interoperability Organization (WS-I) um die Anwendung der WS-Standards für herstellerunabhängige Interoperabilitätslösungen [WI05].

[102] Nach Camarinha-Matos et al. begründet das Service-Paradigma heute eine der drei wesentlichen Strategien zur Realisierung einer VU-Infrastruktur [CM05, S. 91 f.].

den Arbeit. Komposition und Koordination von Web Services sollen die Basis zur Steuerung virt. Dienstleistungsproduktion durch E-Services bilden.

In den folgenden Abschnitten dieser Sektion werden die Grundlagen für diese Vorgehensweise gelegt. Zunächst führt Abschnitt 3.4.1 in die generellen Basiskonzepte Service-orient. Modelle und in die technischen Realisierungsstrategien Service-orient. Architekturen ein. Dann stellt Abschnitt 3.4.2 konkrete Techniken rund um die Web Service-Architektur vor, die im weiteren Verlauf der vorliegenden Arbeit Anwendung finden werden. Schließlich fokussiert Abschnitt 3.4.3 wieder den Aspekt der Prozessunterstützung auf der Basis des Service-Paradigmas und verschiedener WS-Techniken.

3.4.1. Service-orient. Modelle und Architekturen

Der folgende Abschnitt dient dem Einstieg in das Feld des Service Oriented Computing. Zunächst werden in Abschnitt 3.4.1.1 verschiedene grundlegende und erweiterte Aspekte des Paradigmas herausgearbeitet. Im Anschluss wird in Abschnitt 3.4.1.2 die technische Umsetzung der Konzepte auf Architekturebene diskutiert.

3.4.1.1. Service-orient. Modelle (SOM)

Die Konzepte des Service Oriented Computing basieren in erster Linie auf dem Begriff des Web Service. Wie bereits kurz angesprochen, bilden Web Services eine Abstraktion, die im zwischenbetrieblichen Kontext den Zugriff auf innerbetriebliche Anwendungssysteme ermöglicht. Dabei werden die innerbetrieblichen Anwendungssysteme derart gekapselt, dass sie sich nach außen als klar definierte Anwendungsfunktion zur Bindung anbieten und dann über standardisierte Web-Techniken zugegriffen werden können. Bindung und Zugriff von Web Services erfolgt nach dem grundlegenden Interaktionsmuster des Service-orient. Modells (SOM), das den konzeptionellen Grundrahmen des SOC bildet. Genau wie bei den konventionellen Middleware Services bedarf dieses Paradigma jedoch einer Reihe von weiteren Abstraktionen und Mechanismen, die eine adäquate Nutzung im jeweiligen Anwendungskontext sicherstellen. In gleicher Weise wie z. B. Services auf Basis von RPC im innerbetrieblichen Kontext u. a. oft durch Sicherheits-, Replikations-, Transaktions- oder Workflow-Mechanismen erweitert werden, brauchen auch Web Services im zwischenbetrieblichen Kontext entsprechende Pendants. Daher ist im SOC-Kontext eine Erweiterung des SOM-Grundmusters vonnöten, die insgesamt ein kohärentes Bild zwischenbetrieblicher SOC-Funktionalitäten ergibt. In diesem Sinne beinhaltet die Darstellung der SOC-Konzepte im Folgenden zunächst Web Service Definitionen und dann grundlegende sowie schließlich erweiterte Service-orient. Modelle.

Der (Web) Service-Begriff: Definitionsansätze Der Service-Begriff ist in der praktischen Informatik nicht neu. Er wird seit langem in verschiedenen Bereichen als

funktionale Abstraktion eingesetzt.[103] Vor allem im Kontext klassischer Middleware-Plattformen liegt fast allen Interaktionsmechanismen (vor allem RPC und RMI) die Client/Server-Architektur zugrunde. Hierbei steht die Service-Abstraktion im Mittelpunkt. Konkret manifestieren sich Services in Middleware Frameworks durch Schnittstellenbeschreibungen auf Basis von Interface Definition Languages (IDL). Daraus resultiert ein Service-Begriff, der eng am Begriff der operationalen Schnittstelle orientiert ist.[104] So eine eher technische Sicht setzt a priori Wissen u. a. über die Service-Semantik und die Aufrufreihenfolge von Operationen voraus. Dies ist im innerbetrieblichen Kontext adäquat, weil hier oft Service und Client vom gleichen Team entwickelt werden.

Im zwischenbetrieblichen Bereich stößt der Service-Begriff klassischer Client/Server-Middleware an seine Grenzen. In diesem Fall sind Client und Server bzw. Service nicht nur technisch heterogen, sondern auch organisatorisch getrennt. Einziges Bindeglied auf allen Ebenen ist dann das Web. Über dieses müssen sowohl alle zur Service-Bindung notwendigen Informationen vermittelt als auch alle zum Service-Zugriff notwendigen Interaktionen abgewickelt werden [CNW01]. Bindungsinformationen müssen auf geeignete Weise im Web publiziert werden und alle technischen und geschäftlichen Aspekte beinhalten, die für das Auffinden, die Nutzungsentscheidung und den Zugriff benötigt werden. Der Service-Zugriff muss interoperabel erfolgen und sowohl die Besonderheiten der zwischenbetrieblichen Interaktion als auch des Web als Kommunikationsmedium berücksichtigen. Für solche Services, die auf den organisationsübergreifenden bzw. zwischenbetrieblichen (B2B-)Kontext ausgelegt sind und die sich daher in Bezug auf die genannten zusätzlichen Aspekte gegenüber Middleware Services abgrenzen, hat sich der Begriff des *Web Service* etabliert.

Mittlerweile findet sich für den Begriff des Web Service eine kaum überschaubare Menge an Definitionen. Ganz allgemein kann man dabei ein Spektrum ausmachen, das sich von sehr allgemeinen bis hin zu sehr speziellen und von eher konzeptionellen hin zu mehr technischen Auslegungen ausdehnt.

Eine große Zahl von Begriffsauslegungen ist äußerst unspezifisch. Dabei werden oft jegliche Software-Applikationen, die über das Web zugegriffen werden, können als Web Service bezeichnet ("... anything that has a URL is a Web Service." [ACK+04, S. 124]). Es gibt auch Fälle, bei denen der Zugriff auf so genannte „Web Services" nicht als automatisiert, sondern interaktiv charakterisiert wird. Oft findet dabei eine

[103] Ein Bereich, in dem der Service-Begriff besondere Verbreitung erreicht hat, ist die *Kommunikation in Rechnernetzen.* Dies zeigt sich z. B. im OSI-Modell [Ker92]. In der vorliegenden Arbeit wird die Kommunikationsebene jedoch komplett abstrahiert und daher nicht weiter betrachtet.

[104] Es sei erwähnt, dass einige Forschungsarbeiten im Middleware-Kontext auch schon frühzeitig Erweiterungen des Service-Begriffs vorgenommen haben. So sind z. B. Middleware Services für das so genannte Trading (d. h. Vermittlung) von Services mit *Service-Attributen* [MMJL94], *deklarativer Semantik* [Pud95] oder *Protokollautomaten* [GZMW01] erweitert worden.

begriffliche Vermischung mit dem Dienstleistungsbegriff statt.[105] Dies ist z. B. dann der Fall, wenn Web-Seiten von Dienstleistungsunternehmen, die eine Interaktion mit Kunden über HTML-Dokumente erlauben (z. B. online-Tracking bei FedEx[105]), als Web Services bezeichnet werden. Derartige Ansätze leisten in Hinblick auf VDU-Basisinfrastrukturen keinen Beitrag.

In der vorliegenden Arbeit ist ein solcher WS-Begriff von Interesse, der die Abstraktion von Software-Einheiten in den Vordergrund stellt und dabei zum einen die dahinter stehenden Einsatzkonzepte und zum anderen deren technische Realisierung berücksichtigt. In diesem Sinne finden sich vor allem in der Forschung viele Definitionen, die vor allem das Paradigma betreffen und von dessen technischen Details abstrahieren. Z. B. charakterisiert Papazoglou Web Services wie folgt:

> "(Web) Services are self-describing, open components that support rapid, low-cost composition of distributed applications. (Web) Services are offered by service providers – organizations that procure the service implementations, supply their service descriptions, and provide related technical and business support. Since (Web) services may be offered by different enterprises and communicate over the Internet, they provide a distributed computing infrastructure for both intra- and cross-enterprise application integration and collaboration." [PG03]

Diese Definition hebt neben den schon genannten grundlegenden Konzepten einige besondere Aspekte hervor. Es wird hier betont, das Web Services den Charakter von Software-Komponenten besitzen, die explizit zur Komposition verteilter Anwendungssysteme vorgesehen sind, wobei Effekte wie Kosteneffizienz und Entwicklungsgeschwindigkeit positiv zum Tragen kommen. Andere Arbeiten betonen in diesem Zusammenhang u. a. noch die Flexibilität, lose Kopplung und Dokumentenzentrierung von WS-Komponenten [CNW01]. Ferner wird die enge Verbindung von technischen und geschäftlichen Aspekten der Service-Erbringung betont. Schließlich wird die Infrastrukturfunktion in Hinblick auf B2Bi und zwischenbetriebliche Kollaboration in den Mittelpunkt des WS-Konzepts gestellt.

In der obigen Charakterisierung wird außer dem Verweis auf das Internet nichts über die technische Realisierung gesagt. Technisch konkretere Definitionen beziehen sich ganz überwiegend auf die bislang von der W3C spezifizierten Standards. Viele davon beziehen ganz konkrete Standards ein und sind dadurch sehr spezifisch. Die W3C selbst beschreibt Web Services in einer technisch konkreteren und doch noch generellen Weise wie folgt:

> "A Web service is a software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. A Web service supports direct interactions

[105] Siehe z. B. „e-Services" bei Tiwana et al. [TR01].

> with other software agents using XML-based messages exchanged via internet-based protocols." [Sch02a]

Diese Definition bezieht die grundlegenden Einsatzkonzepte des Service-Paradigmas ein und macht deren Web-Bezug durch Techniken wie URI, XML und Internet-Protokolle wesentlich konkreter. Auf der anderen Seite fehlen jedoch die erweiterten Einsatzkonzepte des vorherigen Ansatzes. In der vorliegenden Arbeit soll daher nun ein Mittelweg beschritten werden. In diesem Sinne soll der WS-Begriff im engeren Sinne einer Software-Einheit mitsamt der wichtigsten Einsatz- und Realisierungskonzepte durch folgende Definition erfasst werden:[106]

Definition 13 (Web Service) *Web Services sind autonome und plattformunabhängige Software-Einheiten, die auf Basis von XML-basierten Web-Protokollen in automatisierter Weise beschrieben, publiziert, gefunden, orchestriert und programmiert werden können um als flexible, lose gekoppelte Komponenten bei der Entwicklung massiv verteilter organisationsübergreifender Anwendungssysteme zu dienen.*

Diese Definition charakterisiert Web Services im Sinne der vorliegenden Arbeit als Software-Komponenten, die als Bausteine zur Komposition zwischenbetrieblicher Anwendungssysteme eingesetzt werden können. Eine solche Technik macht wesentliche Middleware-Funktionen im Kontext einer VU-Infrastruktur verfügbar.

Service-Beschreibung, -Auffindung und -Interaktion Die vorangegangenen Definitionen fassen die Konzepte und Techniken des Service-Paradigmas so knapp und prägnant wie möglich zusammen. Im Folgenden sollen nun einige dieser Aspekte noch etwas konkreter ausgeführt werden. Dies betrifft zunächst das grundlegende Interaktionsmuster, das oft als *Service-orient. Modell (SOM)* im engeren Sinn bezeichnet wird.

Das SOM beschreibt die grundlegenden Rollen und Rollen-Funktionen in einer Service-orient. Architektur [PT01].[107] Das darin beschriebene Interaktionsmuster ist äußerst generisch und findet sich z. B. in ähnlicher Form auch schon im Kontext von Middleware-Plattformen.[108] Das SOM beinhaltet die drei fundamentalen Rollen des *Service Providers*, *Brokers* und *Clients*. Die wesentlichen Funktionen dieser Rollen sind durch *Beschreibung/Publikation*, *Auffindung/Auswahl* und *Bindung/Interaktion* gegeben: Der Provider stellt einen Service zur Verfügung, der aus Anwendungssicht

[106] Die Formulierung lehnt sich an eine Arbeitsdefinition der Europäischen Arbeitsgemeinschaft „Network of Excellence in Service-Oriented Computing" an, in der der Autor mitgewirkt hat.

[107] Das Service-orient. Modell wird zum Teil selbst als Service-orient. Architektur bezeichnet (z. B. auch in [PT01]). In der vorliegenden Arbeit wird jedoch eine Trennung von konzeptionellem Modell (SOM) und technischer Architektur (SOA) vorgenommen, die so auch weitgehend üblich ist.

[108] Hier ist ein entsprechendes Interaktionsmuster im ODP-Referenzmodell im Zusammenhang mit der so genannten *Trading Function* zu finden [IJ95, Part 3: S. 60]. Ähnliche Ansätze sind auch mit dem *Trading Object Service (TOS)* von CORBA übernommen worden [OMG04].

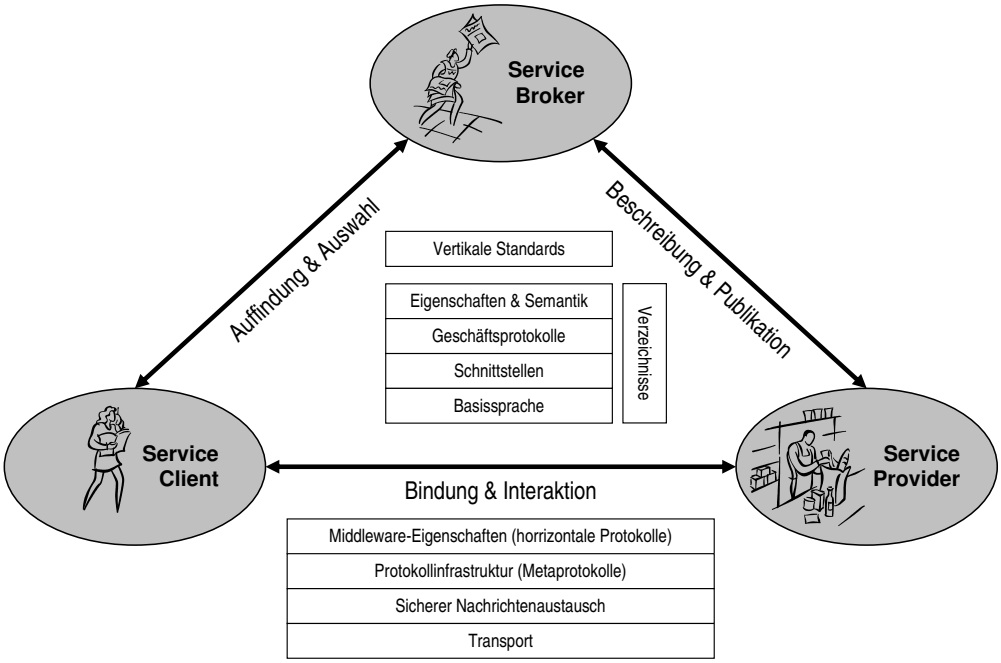

Abbildung 3.20.: Basismodell und -instrumente des SOC

eine Geschäftsfunktion und aus technischer Sicht ein Software-System darstellt. Zur Publikation des Service muss der Provider diesen aus beiden Perspektiven beschreiben. Die Beschreibung hinterlegt er dann bei einem Broker. Der Client kontaktiert den Broker zur Suche in dessen Verzeichnis von Service-Beschreibungen. Er ist an der Geschäftsfunktion des Service und daher auch an der technischen Integration des entsprechenden AS interessiert. In der Service-Beschreibung findet er die notwendigen Informationen zur Auswahl des passenden Service. Des Weiteren findet er dort die nötigen Informationen zur Bindung seines eigenen AS an das AS des Service über das Web. Dadurch können diese Systeme dann in automatischer Weise die zum Zugriff auf den Service notwendigen Interaktionen abwickeln. Dieses Muster wird in Abb. 3.20 durch die klassische Form eines Dreiecks dargestellt.

Die Realisierung des Service-orient. Modells erfordert spezifische (Middleware-)Techniken zur Unterstützung der Rollenfunktionen. Unabhängig von der konkreten Technik können dabei für die einzelnen Funktionen verschiedene zusammenhängende Instrumente identifiziert werden [ACK+04, S. 136 ff.]. Diese sind in Abb. 3.20 durch „Stapel“ dargestellt, um die funktionalen Abhängigkeiten anzudeuten. Im Folgenden werden die Instrumente nach Rollenfunktionen grundsätzlich charakterisiert:[109]

- *Beschreibung und Publikation:* Zur Beschreibung von Services im zwischenbetrieblichen Kontext ist zunächst eine organisationsübergreifend einheitliche und

[109]Konkrete Beispiele werden in Sektion 3.4.2 am Beispiel der W3C/OASIS WS-Architektur erläutert.

akzeptierte (maschinenlesbare) *Basis-* oder *Metasprache* erforderlich. Hierfür eignet sich XML, das sich im B2Bi-Kontext etabliert hat und heute praktisch konkurrenzlos dasteht.[110] Mit der Basissprache ist die technische Schnittstelle des Anwendungssystems zu beschreiben. Bei konventioneller Middleware (z. B. CORBA) werden hierzu die Konstrukte von IDLs verwendet. Bei WS wird die *Schnittstellenbeschreibung* durch XML-Sprachen realisiert. Durch die technische Schnittstellenbeschreibung ist der Nachrichtenaustausch mit einem Service geklärt. Parallel zur Situation im B2Bi sind nun noch Aspekte der korrespondierenden betrieblichen Anwendungsfunktion zu klären. Dies betrifft zum einen das *(Geschäfts-)Protokoll* zur Konversation zwischen Client und Provider. Zum anderen müssen die *Semantik der Operationen/Nachrichten* sowie die *funktionalen und nicht-funktionalen Eigenschaften des Service* beschrieben werden. Diese allesamt generischen Beschreibungsmittel können ggf. durch branchenspezifische (vertikale) Standards reglementiert werden, die z. B. Ontologien einheitlicher Fachbegriffe, Dokumentenstrukturen oder gängige Transaktionen vorgeben. Am Ende muss die Beschreibung in einem öffentlich zugänglichen Verzeichnis bzw. Repository hinterlegt werden. Hierzu ist ein offener Standard notwendig, der *Registrierung und Management von Service-Beschreibungen* regelt. Typischerweise erfolgt dies durch Vorgabe einer API für Service Provider im Rahmen einer weitergehenden Konzeption zur Service-Auffindung.

- *Auffindung und Auswahl:* Die Service-Beschreibung des Providers zielt darauf ab, dem Client ein Auffinden des Service zu ermöglichen und eine Auswahl aus verschiedenen Alternativen in Bezug auf funktionale und nicht-funktionale Eigenschaften zu treffen. Zum Auffinden wird ein entsprechendes Instrument benötigt, das auch als *Discovery-Mechanismus* bezeichnet wird. Dieser von einem Broker betriebene Mechanismus gibt das Interaktionsmuster der beteiligten Rollen bei der Suche nach Services vor. Hierbei ist zunächst die Architektur festzulegen, die zentralisiert oder verteilt sein kann. Entsprechend der Architektur sind die entsprechenden Funktionen der Rollen als APIs bereitzustellen. Dazu gehören z. B. generische Sicherheitsmechanismen, Managementfunktionen für den Provider und Suchfunktionen für den Client. Komplementär dazu muss ein abstrakter Rahmen für die Datenstrukturen des Repository vorgegeben werden. Hierin ist zu regeln, welche Rahmenbedingungen z. B. für die Repräsentation von Providern, ihrer ökonomischen Dienstleistungen und technischen Services bestehen.

- *Bindung und Interaktion:* Die Bindung eines Service dient im Wesentlichen der Integration des Provider-AS mit den Client-AS über das Web. Für den anschließenden Zugriff des Clients auf die Geschäftsfunktionen bzw. Prozesse des Providers wird zunächst eine Kommunikationsverbindung benötigt. Web Ser-

[110] Vgl. Repräsentation von Nachrichtensemantik im B2Bi mit XML in Sektion 3.3.2.4, S. 133.

vices abstrahieren hier grundsätzlich von den unteren Kommunikationsschichten. Stattdessen wird eine Transportschicht auf Basis des Web vorausgesetzt, die heute in der Regel durch HTTP realisiert wird.[111] Aufbauend erfordert das dokumentenzentrierte Service-Paradigma einen Protokoll-Standard für die Interaktion der Systeme durch Nachrichtenaustausch. Hierfür hat sich heute wiederum XML als Basissprache etabliert. Für XML-basierte Nachrichten sind verschiedene Aspekte festzulegen. Dazu gehört u. a. der grundsätzliche Aufbau, die Sicherheit des austauschs, die Adressierung und die Wegewahl. Um auf Basis solcher Nachrichteninteraktionen eine verteilte Middleware für Web-basierte Cl/Srv-Architekturen realisieren zu können ist es notwendig, dass eine *Protokoll-Infrastruktur* für die Interaktion verteilter Middleware-Plattformen existiert. Dazu müssen *Meta-Protokolle* vorgegeben werden, die einen Interaktionsrahmen für konkrete Middleware-Protokolle zwischen den lokalen Plattformen von Client und Provider schaffen. Metaprotokolle dienen z. B. der Einigung auf spezifische Methoden zur Ausführung des Geschäftsprotokolls oder zur Sicherstellung der Service-Qualität. Konkrete *Middleware-Protokolle* setzen dann höherwertige Abstraktionen des Service-Paradigmas um. Diese Protokolle beziehen sich also nicht auf einen spezifischen Service-Inhalt, sondern werden für die verteilte Realisierung beliebiger Service-Zugriffe benötigt. Beispiele hierfür sind die Ausführung von Geschäftsprotokollen oder die Sicherung von Transaktionalität.

Service-Komposition und -Management Das Modell des Broker-Dreiecks gibt das grundlegende Service-Prinzip in sehr anschaulicher Weise wieder. Die Vision des Service Oriented Computing geht jedoch darüber hinaus. Das komplette Bild ergibt sich durch eine Erweiterung der Perspektive auf die Anwendungsebene. Hierbei wechselt der Fokus vom isolierten Service als Software-Komponente mit partieller Anwendungsfunktion auf die Kombination (bzw. Komposition) mehrerer solcher Services zu einem kompletten Anwendungssystem, das eine ganzheitliche geschäftliche Lösung bietet. Im kritischen betrieblichen Einsatz tritt dann die Kontrolle und Steuerung (bzw. Management) solcher Lösungen in den Vordergrund. Diese beiden (visionären) Aspekte des SOC werden von Papazoglou und Georgakopoulos sehr anschaulich durch ein erweitertes SOM erfasst (Vgl. Abb. 3.21).

Die erste Ausbaustufe des erweiterten SOM betrifft die Komposition von Web Services. Dieser Aspekt ist von zentraler Bedeutung, da Web Services per Definition Software-Komponenten darstellen sollen, d. h. dass Services von Grund auf zur Kombination mit anderen Services entwickelt werden, um jeweils einen in sich geschlossenen Teil der Funktionalität eines übergeordneten Anwendungssystems zu erbringen. Diese Vorgehensweise führt zu verschiedenen Vorteilen wie einer schnelleren Entwicklung von AS, einer Wiederverwendung von Service-Komponenten und einer

[111] Vgl. Systemintegration über WAN in Sektion 3.3.2.4, S. 131.

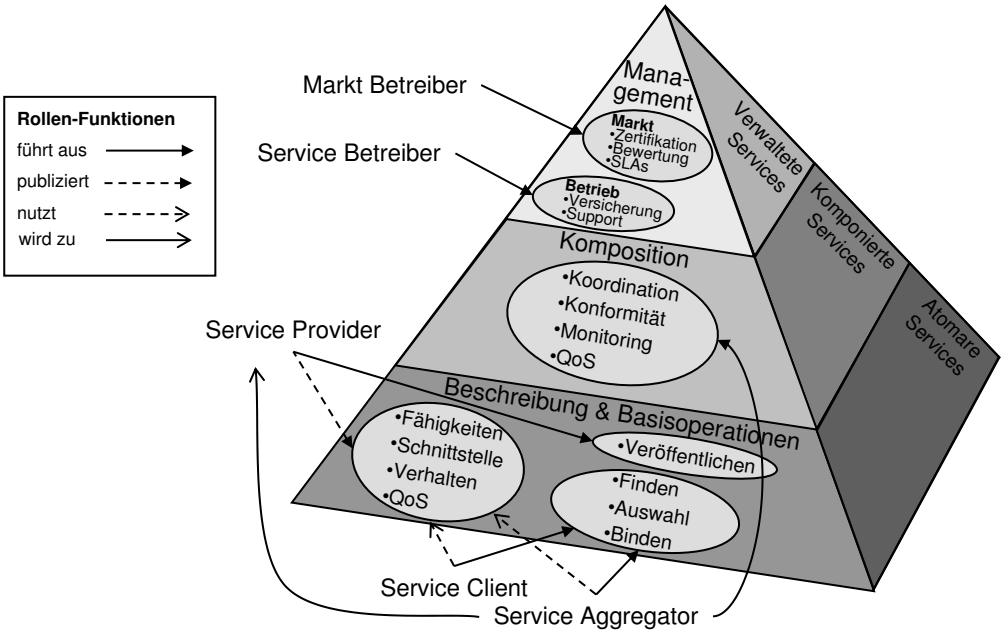

Abbildung 3.21.: SOM aus erweiterter Perspektive (nach [PG03])

Vereinfachung komplexer Nutzungszenarien multipler Services [MM04]. Im Kontext der Service-Orientierung werden dabei vor allem mehrstufige Komponentenmodelle betrachtet: d. h. die Komposition einer Menge von Services führt zu einem AS, das wiederum als Service angeboten werden kann [ACK+04, S. 141]. In diesem Sinne unterscheidet man komponierte von atomaren Services. Komponierte Services werden auf der Basis anderer Services – den Service-Komponenten – realisiert. Atomare Services basieren ausschließlich auf dem lokalen System. Die Komposition von Services wird im erweiterten SOM der zusätzlichen Rolle des Service-Aggregators zugeschrieben [PG03]. Dieser ist naturgemäß Client in Bezug auf Service-Komponenten und ggf. auch Provider in Bezug auf komponierte Services. Clients des Aggregators bemerken auf Grund der Kapselungseigenschaft des Service-Paradigmas keinen Unterschied zwischen atomaren und komponierten Services.

Zur Realisierung komponierter Services sind Instrumente nötig, die es dem Aggregator erlauben, verschiedene spezifische Aspekte zu berücksichtigen. Dabei ist grundsätzlich zu beachten, dass die Komposition von Services gegenüber komponentenorientierten Software-Techniken oder Workflow-basierter A2A-Integration einige Besonderheiten aufweist. Dies liegt nach Yang et al. an den spezifischen Charakteristika der Komposition im Rahmen eines Web-weiten Service-Marktes [YHP02]. Das Angebot an Service-Komponenten fluktuiert und beinhaltet Alternativen; die Komposition muss flexibel und dynamisch auf aktuelle Anforderungen von Clients eingehen, schließlich sind zusätzliche Integrationsaspekte zu beachten. Daraus folgen spezifische Anforderungen an die Kompositionsmechanismen: Milanovic et al.

geben diesbezüglich Konnektivität, Beachtung nicht-funktionaler Service-Qualität, Sicherstellung von Korrektheit und die Möglichkeit zur Skalierung der Instrumente an [MM04].

Ein entsprechendes Instrumentarium zur Service-Komposition lässt sich z. B. auf die Phasen eines Lebenszyklus komponierter Services [YP04] zurückführen, wobei grob Planung, Definition und Implementierung unterschieden werden können [YHP02]. Die Planungsphase beinhaltet den Entwurf eines komponierten Service mitsamt der Auffindung möglicher Service-Komponenten, Überprüfung ihrer Kombinierbarkeit und Berücksichtigung möglicher Alternativen. Dazu wird eine Sprache zur Spezifikation der Komposition benötigt. Zudem sind Mittel zur statischen Analyse z. B. der Kompatibilität von Komponenten [MPC01] hilfreich. In der Definitionsphase wird eine vorläufige Spezifikation mit den Anforderungen des Clients abgeglichen und konkretisiert. Dabei werden Anpassungen an der Spezifikation vorgenommen und konkrete Service-Komponenten ausgewählt. Hierzu sind Mechanismen erforderlich, die aus den nicht-funktionalen Eigenschaften von Service-Komponenten die Qualität des komponierten Service ableiten können (z. B. Gesamtkosten, Performanz, Sicherheit, Authentizität, Privatsphäre, transaktionale Integrität, Verlässlichkeit, Skalierbarkeit und Verfügbarkeit).[112] Zudem ist es erforderlich, die Konformität der gefundenen Service-Komponenten in Bezug auf Geschäftsprotokolle, Datenformate etc. zu überprüfen und diese ggf. aneinander anzupassen. Am Ende dieser Phase steht eine ausführbare Definition der konkreten Komposition. Schließlich werden in der Implementierungsphase die Service-Komponenten gebunden und im Sinne der Kompositionsdefinition ausgeführt. Dies bedingt die skalierbare *Koordination* der Plattformen und Service-Komponenten verschiedener Provider. Zur Gewährleistung ausreichender Konnektivität muss hier eine *Überwachung (Monitoring)* spezifischer Ereignisse einzelner Service-Komponenten möglich sein.

Die zweite Ausbaustufe des erweiterten SOM adressiert das Management von Services. Dies ermöglicht es Unternehmen, kritische Anwendungen und Märkte zu steuern und zu kontrollieren, die auf Basis von Services realisiert werden.

Für den Einsatz von Service-orient. AS in kritischen Unternehmensbereichen sieht das erweiterte SOM Funktionen zum Management des Service-Betriebs vor.[113] Dies schließt ein Management der Service-Plattformen, der Verteilung von Services und der Anwendungsfunktionen ein. Ein Beispiel für solche Managementfunktionen ist die Erstellung von Leistungsstatistiken, die etwa eine Effektivitätsanalyse von Anwendungen erlauben, Einblick in geschäftliche Transaktionen gewähren oder zur Signalisierung von Ereignissen führen. Die Verantwortung für diese Aktivitäten liegt bei der Rolle eines Service-Betreibers, die eine Spezialisierung eines Providers oder Aggregators darstellt. Der Betreiber benötigt dazu Instrumente, die das Management

[112] Ein solcher Mechanismus, der auf Mustern basiert, wird z. B. von Tut et al. diskutiert [TE02].

[113] Eine ausführliche Abhandlung zum Management von Web Services in Unternehmen findet sich bei Sahai und Graupner [SG05].

seiner internen AS um das Management von Web Services erweitern. Letzteres umfasst u. a. das Management des Nachrichtenaustauschs, der Geschäftsprotokolle und der Kompositionen von Services sowie die Korrelation von Daten der Service-Plattform mit anderen Schichten des Gesamtsystems [ACK+04, S. 308 ff.].

Schließlich wird im SOC-Kontext auch die Entstehung neuer Märkte antizipiert. Im Gegensatz zu manchen der bestehenden branchenspezifischen vertikalen Marktplätze sagen Papazoglou und Georgakopoulos hier offene Märkte voraus, auf denen sich Anbieter und Nachfrager auf Basis ihrer Services in beliebiger Weise gruppieren können.[114] Die Macher solcher Märkte geben den Teilnehmern eine einheitliche Sicht auf Dienstleistungen und Produkte, Terminologie und Geschäftsprozesse. Hinzu kommen Mehrwertdienste zur Unterstützung geschäftlicher Transaktionen. Hierbei leisten verschiedene Funktionen zum Management von Märkten Hilfestellung: angedacht sind diverse Funktionen wie die Unterstützung geschäftlicher Verhandlung, Vereinbarung/Vertragslegung, Zertifizierung, Versicherung, Rating und Statistik sowie Verhandlung und Überwachung von Service-Vereinbarungen. Markt-Management dient so der Administration offener Service-Märkte und wird von deren Machern zur Instandhaltung konstanter Rahmenbedingungen eingesetzt.

Das Modell offener Service-Märkte ist freilich nach wie vor eine Vision. Es deutet aber in überspitzter Weise die Richtung an, in der auch Konzepte zur Unterstützung von virt. Dienstleistungsunternehmen liegen können.

3.4.1.2. Service-orient. Architekturen (SOAs)

In der letzten Sektion wurde ein ganzheitliches Modell des Service Oriented Computing mitsamt der Konzepte und Instrumente auf verschiedenen Ebenen eingeführt. Es stellt sich nun die Frage, wie dieses Modell sich in die Architektur eines verteilten Systems einfügt. Dabei ist der Ausgangspunkt durch Middleware-basierte Mehrschichtarchitekturen für verteilte Anwendungssysteme gegeben, wie sie in Sektion 3.3.2 eingeführt wurden. Diese müssen nun entsprechend erweitert werden.

Zur Erläuterung ist eine Differenzierung nützlich, die sich auf zwei grundlegende Facetten des SOC bezieht [ACK+04, S. 141 ff.]: Zum einen sollen WS den externen Zugang zu den internen Systemen einer Organisation ermöglichen. Dies erfordert die Erweiterung der betrieblichen Middleware im Sinne einer *internen SOA*. Zum anderen sollen WS der organisationsübergreifenden Integration dienen. Dies erfordert organisationsübergreifende Middleware-Mechanismen im Sinne einer *externen SOA*. Diese Bereiche werden im Folgenden getrennt skizziert.

Interne Service-orient. Architektur Aus innerbetrieblicher Perspektive bedingt die Einführung des SOM eine Erweiterung der internen Middleware. Konzeptionell

[114] Siehe hierzu auch Merz [Mer96].

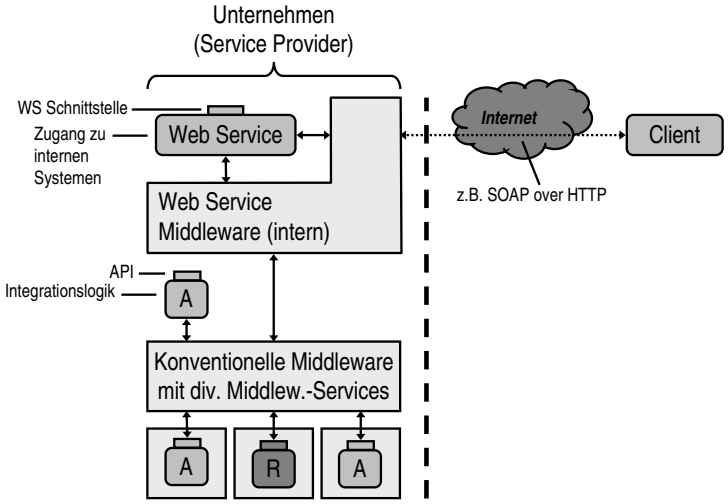

Abbildung 3.22.: SOA aus interner Perspektive (nach [ACK+04, S. 145])

kann dazu die Schichtenarchitektur eines AS um eine weitere Schicht ergänzt werden, in der die WS-Middleware zum Tragen kommt (vgl. Abb. 3.22).

Jede Middleware stellt Mechanismen bereit, um die Interaktion zwischen Software-Einheiten, die Integration von Daten und Funktionen und die Abstraktion als höherwertige Services zu ermöglichen. Dies kann bei N-Schicht-Architekturen prinzipiell beliebig oft wiederholt werden. Interne WS-Middleware setzt auf den diversen Schichten innerbetrieblicher AS auf und ermöglicht den Zugriff auf deren lokale Services über das Web. Ähnlich wie bei Applikationsservern besteht die wesentliche Aufgabe dieser WS-Middleware in der Transformation von Aufrufen und Datenformaten lokaler Services in die Formate des Web gemäß der verwendeten Nachrichtenaustausch- und Transportprotokolle für den Web Service-Zugriff.

Es ist anzumerken, dass die interne SOA den Einsatzbereich von WS klar beschränkt. Der zusätzliche Aufwand der WS-Middleware lässt sich nämlich nur dann vertreten, wenn der Zeitaufwand für die zusätzlichen Transformationen gegenüber der Ausführungszeit der Anwendungsfunktion nicht ins Gewicht fällt. Daraus folgt, dass WS aus Software-technischer Sicht nicht wie verteilte Objekte, sondern eher wie Software-Komponenten zu verwenden sind.

Externe Service-orient. Architektur Interne Web Service Middleware realisiert Web Services als Komponenten, durch die spezifische Anwendungsfunktionen betrieblicher Anwendungssysteme organisationsübergreifend zugreifbar sind. Das Ziel des Service Oriented Computing ist jedoch darüber hinaus die organisationsübergreifende Integration verschiedener WS-Komponenten. Die dazu notwendigen Middleware-Mechanismen überschreiten die Grenzen betrieblicher Domänen. Man spricht daher auch von externer WS-Middleware. Ihre Struktur wird durch die externe SOA bestimmt.

Die Diskussion der externen SOA geht konform zu derjenigen des B2Bi.[115] Das Problem besteht darin, die Integrationsfunktionen, die von konventionellen und insbesondere integrativen Middleware-Mechanismen[116] im lokalen Kontext erbracht werden, auf den organisationsübergreifenden Kontext zu übertragen. Dazu sind zwei Architekturmuster möglich:

- *Zentralisierte Broker* – Die Integrationsfunktion wird in zentraler Form realisiert. Ein vertrauenswürdiger und/oder neutraler Partner betreibt die gesamte externe Middleware.

- *Protokoll Infrastruktur* – Die Integrationsfunktion wird in verteilter Form realisiert. Jeder Partner betreibt seine eigene externe Middleware. Eine gemeinsame Protokoll-Infrastruktur erlaubt die Interaktion der Plattformen und Abwicklung von Punkt-zu-Punkt Middleware-Protokollen.

Für die externe SOA kommen heute beide Muster zum Einsatz. Zentralisierte Broker erlauben eine direkte Übernahme vieler Konzepte, die für konventionelle Middleware entwickelt wurden. Das wesentliche Gegenargument ist ein Vertrauensproblem. Entsprechend werden unkritische Funktionen, wie z. B. das Auffinden von Services, in der Regel durch zentrale Broker realisiert. Die Kontrolle kritischer Funktionen, wie z. B. das Management geschäftlicher Transaktionen, wird aber von Unternehmen meist nicht an externe Partner übergeben. In diesem Fall gewährleistet eine verteilte Middleware die Unabhängigkeit gegenüber einzelnen Partnern und beschränkt deren Zugriff auf sensible Bereiche.

Für die Realisierung einer verteilten Middleware bietet sich zunächst ein organisationsübergreifender Einsatz homogener Middleware-Plattformen an. Dies ist aber, wie im Kontext des B2Bi bereits erläutert wurde, nicht immer plausibel oder wünschenswert. SOC-Techniken zielen deshalb darauf ab, die verteilten Mechanismen auf Basis heterogener Middleware-Plattformen zu realisieren. Dazu muss zunächst eine Infrastruktur für die Abwicklung von Middleware-Protokollen geschaffen werden. Diese beinhaltet Metaprotokolle, die grundlegende Interaktionsmuster für heterogene Middleware-Plattformen umsetzen. Auf Basis einer solchen Protokoll-Infrastruktur können dann konkrete Middleware-Protokolle realisiert werden. Dabei müssen zum einen die Protokolle konventioneller Middleware derart angepasst werden, dass sie statt innerbetrieblicher/zentralisierter in organisationsübergreifender/Punkt-zu-Punkt Manier arbeiten. Zum anderen ist zu beachten, dass dies eine plattformübergreifende Standardisierung von Middleware-Funktionen impliziert.

Zurzeit ist das zentrale Broker-Muster die bekannteste Struktur der externen SOA. Dies liegt daran, dass der dominierende Mechanismus zum Publizieren und Auffinden

[115] Vgl. Sektion 3.3.2.4.

[116] Z. B. Message-Broker und WfMS.

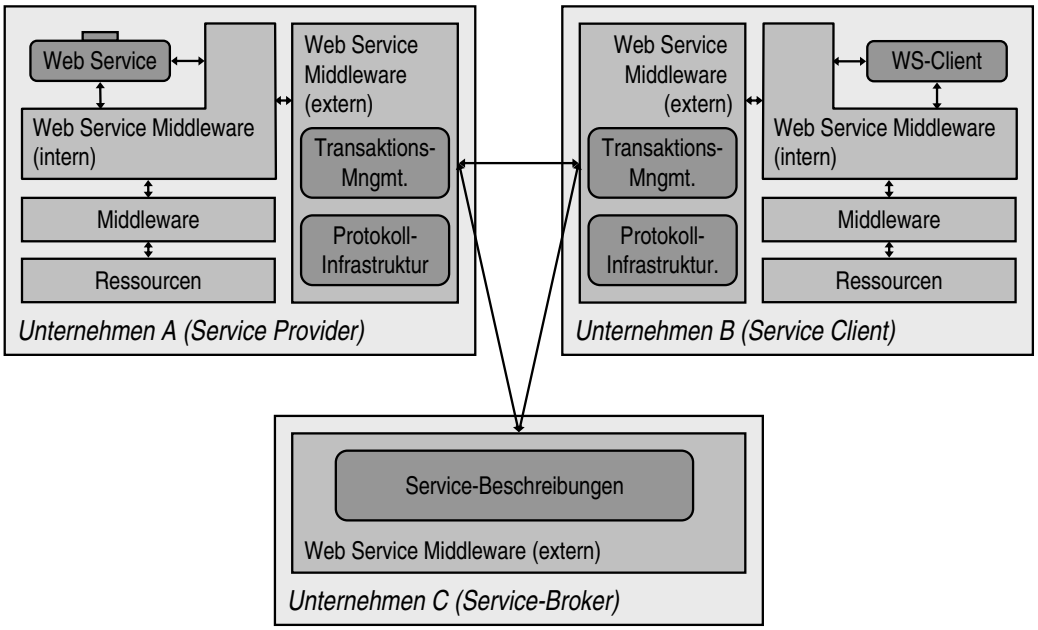

Abbildung 3.23.: SOA aus externer Perspektive (nach [ACK+04, S. 148])

von Services über Verzeichnisse in dieser Art realisiert ist.[117] Die verteilte Protokoll-Infrastruktur ist hingegen weniger verbreitet. Bislang wird diese vor allem im Zusammenhang mit der Punkt-zu-Punkt Realisierung von Protokollen zum Management von Transaktionen eingesetzt.[118] Abbildung 3.23 zeigt die externe SOA am Beispiel dieser SOM-Mechanismen. Für viele andere potenzielle Integrationsfunktionen ist die externe Architektur noch unklar. Dies gilt u. a. auch für die Komposition von Services. Hierbei sind sowohl zentrale als auch verteilte Middleware-Lösungen denkbar. Bislang liegt über die optimale Form keine einheitliche Erkenntnis vor.

3.4.2. Ausgewählte Web Service (WS) Techniken

Im vorangegangenen Abschnitt wurden Web Services auf abstrakter Ebene als wesentliche Elemente des SOC eingeführt. An dieser Stelle sollen nun konkrete Techniken folgen, die gemeinhin mit dem Schlagwort „Web Services“ assoziiert werden [Bet01].

Gemäß der vorangegangenen Darstellung werden zunächst Techniken für das grundlegende Service-orient. Modell des Broker-Dreiecks vorgestellt. Hier hat sich eine über-

[117] Gemeint ist hier Universal Description, Discovery and Integration (UDDI). Es gibt jedoch auch voll verteilte Discovery-Mechanismen auf Basis von P2P-Netzen [Gre05].

[118] Hierbei liegt das vertikale Metaprotokoll *WS-Coordination* zugrunde [CCF+05c]. Darauf bauen Middleware-Protokolle für klassische (*WS-AtomicTransaction* [CCF+05a]) und geschäftliche Transaktionen (*WS-BusinessActivity* [CCF+05b]) auf.

schaubare Menge von Basistechniken etabliert. Diese *Web Service-Basisarchitektur* bildet eine minimale Grundlage zur Realisierung Service-orient. Anwendungssysteme.

Die restlichen beiden Abschnitte beziehen sich auf spezifische Aspekte des erweiterten Service-orient. Modells. Da im Kontext der vorliegenden Arbeit vor allem die Potenziale der WS-Techniken zur Prozessintegration von Interesse sind, folgt hierbei eine Darstellung externer WS-Middleware-Mechanismen zur Unterstützung von Interaktionsprozessen. Hierbei bilden Koordination und Komposition einzelner Interaktionen die zentralen Aspekte. WS-Koordination befasst sich mit der *Regelung zusammenhängender Web Service-Interaktionen*. Web Service-Komposition ist hingegen eine Technik zur *Implementierung geregelter Web Service-Interaktionen* durch einen Provider.

3.4.2.1. Die Web Service-Basisarchitektur

Als Web Service-Basisarchitektur soll hier eine Systemarchitektur auf Basis der drei Techniken SOAP, WSDL und UDDI verstanden werden [Bet01]. Diese beinhalten Systemfunktionen zur nachrichtenbasierten Interaktion, zur Service-Beschreibung und zur Service-Auffindung, mit denen Service-orient. AS im Sinne des Broker-Dreiecks realisiert werden können (vgl. Abb. 3.24). Da mittlerweile alle Techniken vom W3C standardisiert wurden und die Web Service-Basisarchitektur damit weitgehend bekannt ist, wird sie im Folgenden nur kurz skizziert.[119] Dazu wird zunächst die Gesamtarchitektur dargestellt. Anschließend erfolgt eine kurze Beschreibung der drei Techniken im Einzelnen.

Die wohl grundlegendste Funktion des Service-orient. Modells besteht im wechselseitigen Nachrichtenaustausch zwischen potenziell heterogenen Systemplattformen verschiedener Organisationen über das Internet. Hierzu dient in der WS-Architektur das Simple Object Access Protocol (SOAP). SOAP definiert interoperable Nachrichtenformate und deren Transport mithilfe von Internetprotokollen. Als Basissprache für die Nachrichtenbeschreibung wird XML verwendet. Dadurch können Nachrichten zum einen strukturiert und zum anderen plattformübergreifend repräsentiert werden. SOAP gibt hierzu u. a. die Nachrichtenstruktur sowie Richtlinien zur Kodierung verschiedener Datenformate vor. Um die lose Kopplung Service-orient. Systeme hervorzuheben, sieht SOAP grundsätzlich die asynchrone Kommunikation von Nachrichten vor. Wegen der weiten Verbreitung prozeduraler Paradigmen wird aber auch die synchrone Kommunikation im Sinne von RPC unterstützt. Die Kommunikationsform wird dabei durch das verwendete Transportprotokoll bestimmt. Hierbei berücksichtigt SOAP die zwei verbreitetsten Internet-Protokolle SMTP und HTTP für den synchronen bzw. asynchronen Nachrichtenaustausch.

Nachdem SOAP die Möglichkeit bietet, mit Web Services Nachrichten auszutauschen, besteht der nächste Schritt darin, Web Services zu beschreiben, so dass Clients

[119] Für eine ausführliche Darstellung siehe z. B. [ZDN+01].

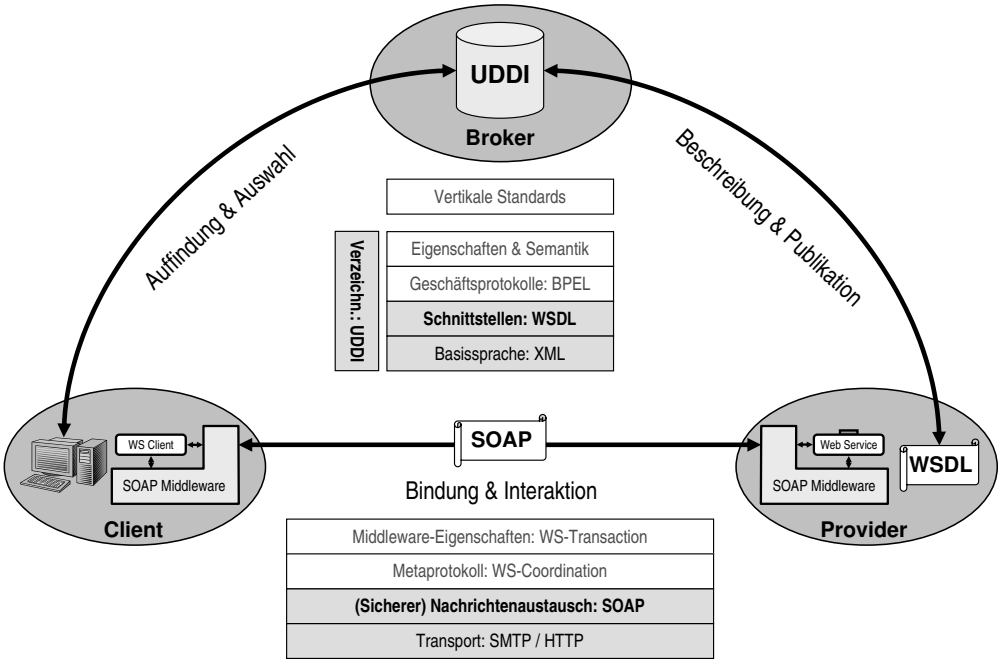

Abbildung 3.24.: Web Service-Basisarchitektur

sich an diese binden können. Das Format entsprechender Service-Beschreibungen wird durch die Web Service Definition Language (WSDL) vorgegeben. Diese Sprache besitzt große Ähnlichkeit zu den IDLs konventioneller Middleware: Mit ihrer Hilfe kann die Schnittstelle eines Web Service derart beschrieben werden, dass ein Client dessen Funktionen mittels seiner internen Web Service Middleware transparent zugreifen kann. Wegen des Fehlens zentraler Middleware-Mechanismen müssen jedoch mehr Informationen als bei klassischen IDL erfasst werden. Auf Grundlage der Basissprache XML spezifiziert ein WSDL-Dokument zunächst die nachrichtenbasierten Interaktionen im Kontext der Service-Funktionen. Daneben ist aber auch die Realisierung dieser Interaktionen durch ein Nachrichten- und/oder Transportprotokoll sowie die Adresse eines Endpunkts, über den der Service-Provider den Dienst zugänglich macht, enthalten. In den meisten Fällen werden SOAP und HTTP verwendet und der Endpunkt ist eine URI.

Um dem Client eine Auswahl und Bindung von Web Services zu erlauben, muss er deren WSDL-Dokumente lokalisieren und darauf zugreifen können. Diese Funktion wird u. a. durch Service-Verzeichnisse zur Verfügung gestellt. In der WS-Basisarchitektur kommt dafür die UDDI-Technik (Universal Description, Discovery and Integration) zum Einsatz. UDDI beinhaltet ein Konzept zur Publikation und Auffindung von Services, das auf zentralisierten Brokern basiert. Broker dienen als zentrale Anlaufstelle für die Registrierung von Service-Beschreibungen durch Provider und die Suchanfragen von Clients. Um diese zusammenzubringen, betreiben sie eine (logische)

Datenbank mit Service-relevanten Informationen. UDDI gibt hierfür eine umfangreiche Datenstruktur zur Repräsentation vielfältiger Informationen über Provider, deren Services und technischen Zugriff (u. a. in Form von WSDL) vor. Die Interaktion mit dem Broker erfolgt über verschiedene APIs. Diese erlauben es Providern sich selbst, die Arten ihrer Services und deren konkrete Angebote zu registrieren und diese Daten zu verwalten. Clients können hingegen konkrete Daten abfragen oder im Katalog blättern. Um es Clients und Providern zu erlauben, die UDDI-Funktionen mit den internen Anwendungssystemen zu integrieren, sind diese selbst als Web Services konzipiert, d. h. dass die UDDI-APIs mit WSDL beschrieben sind und über SOAP darauf zugegriffen werden können.[120]

Web Service-Zugriff mit SOAP Das Ziel des Simple Object Access Protocol (SOAP)[121] liegt darin, eine generische Möglichkeit zu schaffen, um zwischen den Web Services verschiedener Organisationen Nachrichten austauschen zu können. Dabei bilden die Techniken und Standards des Internets und des Web die Grundlage. Neben den dort etablierten Kommunikations- und Transportprotokollen spielt vor allem XML eine wesentliche Rolle. Diese Metasprache bietet sich als Basis an, da sie die einheitliche und strukturierte Repräsentation von Daten erlaubt. Zudem sind für die meisten Systemplattformen Mechanismen zur automatischen Interpretation und Verarbeitung von XML-Dokumenten vorhanden. In SOAP wird nun zunächst festgelegt, wie mithilfe von XML-Nachrichten aufzubauen sind. Dies betrifft in erster Linie die *Nachrichtenstruktur*. Es werden aber auch Richtlinien zur *Datenkodierung* vorgegeben. Als Nächstes zeigt SOAP verschiedene Möglichkeiten auf, wie XML-Nachrichten über das Internet versendet werden können. Dies geschieht durch eine *Bindung an Transportprotokolle* für die gängigsten Varianten. Im Sinne eines Kommunikationsprotokolls legt SOAP schließlich noch einige *Verhaltens- und Interaktionsmuster* fest, die von den Kommunikationspartnern einzuhalten sind.

Das durch SOAP festgelegte Nachrichtenformat basiert ganz auf der Tradition hierarchischer Protokolle auf einem Umschlag, der sämtliche Bestandteile einer Nachricht aufnimmt. Dieser *SOAP-Envelope* enthält genau einen *SOAP-Header* und einen *SOAP-Body*. Der Header ist optional und enthält Informationen, die zum Versand und zur Verarbeitung der Nachricht auf ihrem Weg zum Empfänger relevant sind. Dies können z. B. Informationen zur Sicherheit, Wegewahl oder Transaktionalität bei der Nachrichtenübertragung sein. Er kann aus verschiedenen Teilen *(Parts)* bestehen, die unterschiedliche Aspekte betreffen. Der Body ist hingegen obligatorisch. Er kann ebenfalls aus verschiedenen Parts bestehen, die alleine für den Empfänger der Nachricht bestimmt sind. Die Body Parts werden dabei natürlich wie die gesamte Nachricht auch durch XML-Strukturen repräsentiert. Die Daten, die in einer Nachricht

[120] Dies wurde in Abb. 3.24 abstrahiert.

[121] Die folgenden Betrachtungen der vorliegenden Arbeit beziehen sich auf den zur Zeit der Erstellung aktuellen Standard *SOAP 1.2* [GHM+03].

verschickt werden sollen, liegen jedoch nicht in jedem Fall im XML-Format vor. Deren XML-Kodierung muss dann bei Sender und Empfänger einheitlich erfolgen. SOAP erlaubt daher die Festlegung beliebiger Kodierungsregeln. Darüber hinaus werden zwei konkrete Kodierungen vorgegeben: Das *SOAP Encoding* beschreibt eine Möglichkeit zur Repräsentation generischer Datentypen durch XML. Das *Literal Encoding* regelt den Versand von XML-Dokumenten als Body Parts, indem ganz einfach deren XML-Schemata die Kodierung bestimmen.

SOAP-Nachrichten werden von einem Sender an einen Empfänger verschickt und können auf ihrem Weg beliebig viele Zwischenstationen (Knoten) passieren. Jeder Knoten untersucht den Header auf für ihn relevante Informationen, verarbeitet die Nachricht, ändert, löscht oder ergänzt ggf. Header Parts und sendet die Nachricht schließlich weiter. SOAP gibt diesbezüglich verschiedene Verhaltensregeln vor. So ist es u. a. möglich, die Berücksichtigung von Header Parts vorzuschreiben. Jeder Knoten, der dies nicht kann, ist verpflichtet, die Übertragung zu beenden und einen Fehler zu melden.[122] Es ist anzumerken, dass SOAP-Knoten meist nicht durch Rechnerknoten verschiedener Organisationen gegeben sind. Sie ergeben sich vielmehr oft aus den verschiedenen Schichten einer verteilten Systemarchitektur, in der Nachrichten durch ggf. multiple Middleware-Plattformen an die Software-Komponenten verteilter AS geleitet werden.

Mithilfe von SOAP-Nachrichten können verschiedene Interaktionsmuster zwischen Sendern und Empfängern realisiert werden. Grundsätzlich ist dabei eine asynchrone Interaktion vorgesehen, d. h. die Beziehungen zwischen einzelnen Nachrichtenübertragungen werden im Rahmen von SOAP gar nicht beachtet. Auf Grund der weiten Verbreitung gibt SOAP jedoch eine Möglich zur Realisierung synchroner Request-Response Interaktionen vor, um damit den Aufruf von RPC zu ermöglichen. Dies geschieht durch Bindung der Nachrichtenübertragung an ein Transportprotokoll mit entsprechenden Eigenschaften. Für die synchrone Interaktion schreibt SOAP die Übertragung von Nachrichten (z. B. RPC-Aufrufe) im Rahmen eines HTTP-Requests vor. Der synchrone HTTP-Reply enthält dann eine Nachricht als direkte Antwort (z. B. RPC-Rückgabewert). Für die asynchrone Interaktion ist daneben der Versand von Nachrichten mithilfe von SMTP vorgesehen. Ein weiterer wichtiger Aspekt der Bindung ist die Adressierung des Empfängers. Diese wird in SOAP zurzeit nicht direkt behandelt, sondern durch das Transportprotokoll bestimmt.[123]

[122] Dies ist nützlich um z. B. zu verhindern, dass eine sicherheitsrelevante Nachricht unverschlüsselt übertragen wird, weil ein Knoten den entsprechenden Verschlüsselungsmechanismus nicht beherrscht.

[123] Durch diese Praxis entstehen zum Teil erhebliche Probleme für Service-orient. AS. Daher wird zurzeit mit *WS-Addressing* ein Standard entwickelt, der die Adressierung einheitlich regelt [BCC+04]. Dieser ist jedoch zur Zeit der Erarbeitung noch nicht verfügbar.

Web Service-Beschreibung mit WSDL Ziel der Web Service Definition Language (WSDL)[124] ist es, eine Beschreibung von Web Services zu ermöglichen, die es Clients erlaubt, das dahinterstehende Service-Angebot zu verstehen und in die eigene AS-Architektur zu integrieren. Dazu umfasst WSDL Mittel zur Beschreibung von Web Service-Funktionen in Bezug auf operationale Schnittstellen, deren Bindung an konkrete Zugriffsmechanismen, die Gruppierung dieser konkreten Schnittstellen zu Services und die Endpunktadressen des Service-Providers im Internet. Dabei dient wiederum XML als Basissprache. Ein entsprechendes WSDL-Dokument kann von Clients ganz ähnlich wie eine IDL-Schnittstelle konventioneller Middleware verwendet werden.

WSDL-Dokumente fokussieren im Wesentlichen die operationalen Schnittstellen eines Web Service. Als deren Grundlage wird zunächst die Bestimmung eines Typsystems ermöglicht. Hierfür wird in der Regel das Typsystem von XML-Schema verwendet. Dieses erlaubt die Deklaration aller gängigen Formen einfacher und komplexer *Datentypen*. Typinformationen erlauben den Interaktionspartnern die Konstruktion und Interpretation gesendeter und empfangener *Nachrichten*. Entsprechend werden die deklarierten Datentypen zur Definition von Nachrichten verwendet; diese können aus beliebig vielen *Nachrichtenteilen* mit individuellen Typen bestehen. Nachrichten bilden wiederum die wesentlichen Bestandteile von *Web Service-Operationen*. WSDL unterscheidet dabei zwischen vier verschiedenen Interaktionsmustern: *Request-Response*, *Solicit-Response*, *One-Way* und *Notification* repräsentieren Interaktionen, bei denen der Client aufruft und der Provider synchron antwortet, der Provider aufruft und der Client synchron antwortet, der Client asynchron aufruft und der Provider asynchron aufruft. Dies unterstreicht die Vielfalt WS-basierter Interaktionen und eröffnet vielfältige Möglichkeiten für Integrationskonzepte. Je nach Interaktionsmuster enthält eine Operation einen oder zwei Nachrichtentypen, die bei der Interaktion zu verwenden sind. Verschiedene solcher Operationen werden schließlich durch einen *Port Type* zusammengefasst. Port Types bilden somit ein Äquivalent zu den IDL-Schnittstellen konventioneller Middleware.

An dieser Stelle endet dann die Ähnlichkeit. Die weiteren WSDL-Bestandteile dienen nämlich im Wesentlichen dem Zweck, das Fehlen einer zentralen Middleware zu kompensieren. Dementsprechend müssen Informationen, die sich bei konventioneller Middleware implizit ergeben, nun explizit ausgewiesen werden. Dies betrifft zunächst den Zugriffsmechanismus für die Operationen. Da hier keine einheitliche Methode festgeschrieben werden soll, muss dies individuell geschehen. Dazu wird ein so genanntes *Binding* festgelegt. Dieses gibt zunächst an, ob Operationen als RPC (Parameteraustausch) oder Dokumentenaustausch realisiert sind. Entsprechend wird dann das Transportprotokoll zum Nachrichtenaustausch festgelegt. Eine gängige Variante ist hier die Verwendung von SOAP, z. B. in Form von RPC auf Basis von

[124] Die folgenden Betrachtungen der vorliegenden Arbeit beziehen sich auf den zur Zeit der Erstellung aktuellen Standard WSDL 1.1 [CCM+01].

HTTP. Des weiteren spezifiziert das Binding die Kodierungen von Nachrichten. Dies geschieht für alle Operationen eines Port Type.

Auf Basis konkreter Bindings wird am Ende der eigentliche *Service* definiert. Genauer basiert dieser auf einer Menge von *Ports*. Ports verbinden ein Binding mit der Adresse eines *Endpoint*. Der Endpoint gibt an, an welchem Ort (im Internet) auf das Binding zugegriffen werden kann. Insgesamt stellt ein Port somit die Konkretisierung eines Port Type bezüglich Zugriffsmechanismen und -ort dar. Verschiedene solche Ports ergeben die gesamte Funktionalität eines Service.

Als Ganzes betrachtet, können WSDL-Beschreibungen generell in einen abstrakten und einen konkreten Teil untergliedert werden. Port Types mitsamt ihren Operationen, Nachrichten und Typen beschreiben die Funktionalität eines Service auf abstrakter Ebene. Services mitsamt ihren Ports und Bindings beschreiben hingegen eine konkrete Form der technischen Realisierung von Service-Interaktion durch einen konkreten Provider. Es ist dabei durchaus üblich, die beiden Teile getrennt voneinander zu spezifizieren und ganz generell verschiedene Elemente wiederzuverwenden. Z. B. kommt es häufig vor, dass die abstrakte Funktionalität von Service-Schnittstellen durch Standardisierungsgremien vorgegeben wird. Provider verwenden dann mithilfe der WSDL Import-Funktion die Beschreibung des Standards wieder und ergänzen die konkreten Teile ihrer Services.

Die Be- und Verarbeitung von WSDL-Dokumenten erfolgt in überwiegender Weise automatisiert. Provider generieren oft WSDL-Beschreibungen aus ihren internen Systemen; z. B. aus den Klassen eines Objektsystems. Clients und Provider generieren dann aus diesen WSDL-Beschreibungen wiederum die Zugriffsmechanismen über das Web, z. B. in Form so genannter *Stubs*, die eine Kodierung bzw. Dekodierung gemäß SOAP vornehmen und die Service-Funktionen als lokale programmiersprachliche Abstraktionen repräsentieren.

Web Service-Vermittlung mit UDDI Durch den UDDI-Standard[125] (Universal Description, Discovery and Integration) soll ein einheitlicher Rahmen zur inhaltlichen und technischen Integration von Unternehmen geschaffen werden. Das zentrale Konzept dazu ist die *Universal Business Registry (UBR)*. Dieses Verzeichnis soll zunächst eine zentrale Anlaufstelle zur Vermittlung von Unternehmen und ihrer Dienstleistungen bieten. Darüber hinaus soll ein UDDI-Verzeichnis aber auch die technische Implementierung der Dienstleistungen erfassen und potenziellen Kunden deren Integration im Sinne von B2Bi ermöglichen. Obwohl keineswegs darauf beschränkt, fokussiert UDDI dabei die Web Service-Techniken. Aus dieser Perspektive kann ein UDDI-Verzeichnis also als erweiterter Namens- und Verzeichnisdienst für Web Services aufgefasst werden. Die duale Natur der Verzeichnisinhalte spiegelt sich dann auch in der Verwendung von UDDI wider. Hier liegt das Ziel zum einen in der Unterstützung interaktiver

[125] Die folgenden Betrachtungen der vorliegenden Arbeit beziehen sich auf den zur Zeit der Erstellung aktuellen Standard UDDI 3.0.1 [ABC+02].

Nutzungszenarien. Z. B. soll es Unternehmen ermöglicht werden, potenzielle Partner zu suchen und deren technische Integration zu planen. Zum anderen soll UDDI als externer Middleware Service dienen. Dieser soll vor allem eine dynamische Bindung von Web Services unterstützen.

Um das breite inhaltliche und funktionale Spektrum abzudecken, gibt UDDI einen generischen Rahmen verschiedener Datenstrukturen vor. Dieser strukturiert Informationen in Analogie zu den gängigen Formen von Telefonbüchern nach ihrer Verwendung in Form von *White*, *Yellow* und *Green Pages*. Diese dienen alle der Suche nach Dienstleistungen, wobei die Ausgangsbasis entweder ein bekanntes Unternehmen (z. B. UPS), eine inhaltliche Kategorie/Branche (z. B. Paketversand) oder eine bestimmte Technik (z. B. WSDL-Schnittstelle eines Logistik-Standards) ist. Die entsprechenden Informationen werden in UDDI durch die vier Datenstrukturen *businessEntity*, *businessService*, *bindingTemplate* und *tModel* repräsentiert. BusinessEntities beinhalten jeweils die informale Beschreibung eines Unternehmens. Jede ist mit einem oder mehreren businessServices verknüpft. Diese beschreiben – ebenfalls auf informale Weise – jeweils eine Dienstleistung. Beide Datenstrukturen können mit passenden Kategorien einer beliebigen anwendungsspezifischen Ontologie verknüpft werden. Ein businessService referenziert des Weiteren eine Menge von bindingTemplates. Letztere erfassen die notwendigen Informationen, um eine Dienstleistung technisch zu integrieren. Dazu gehört der Typ der Implementierung, die Adresse an der das Unternehmen eine solche Implementierung vorhält und deren konkrete Eigenschaften. UDDI macht hierbei jedoch explizit keine Vorgaben für die Beschreibung von technischen Details. Diese können beliebig ausfallen. Es muss nur auf ein tModel verwiesen werden, das den konkreten Hintergrund definiert. tModels repräsentieren ein technisches Modell (z. B. einen Standard), das beliebig oft referenziert werden kann. Für die Beschreibung des Modells wird dabei auf eine externe Quelle verwiesen (z. B. ein Standard-Dokument). Diese indirekte Form der Beschreibung vermeidet die Festlegung auf bestimmte Standards und erlaubt eine flexible Anpassung an deren ständige Weiterentwicklung. Sie wird nicht nur für die Techniken verwendet, auf die innerhalb von bindingTemplates Bezug genommen wird, sondern auch für die Ontologien, auf die sich die Kategorien von BusinessEntities und -Services beziehen.

Der Zugriff auf das UDDI-Verzeichnis erfolgt über eine Reihe von APIs, die selbst als Web Services realisiert sind. Die Schnittstellen der *Publishers API*, *Inquiry API*, *Subscription API*, *Security API*, *Custody and Ownership Transfer API* sowie der *Replication API* liegen als abstrakte WSDL-Beschreibungen vor und können konkret über SOAP zugegriffen werden. Ihre Funktionen sollen hier jedoch nicht im einzelnen vertieft werden, da sie für die vorliegende Arbeit keine konkrete Relevanz besitzen. Es sei nur allgemein gesagt, dass sie die Publikation von Daten, die Suche im Verzeichnis, die Beobachtung von Änderungen, die Authentifizierung von Benutzern sowie die Verschiebung und Replikation von Einträgen zwischen verschiedenen Verzeichnissen ermöglichen.

3.4.2.2. Regelung zusammenhängender WS-Interaktionen

Die Web Service-Basisarchitektur realisiert eine Grundform des Service-orient. Modells. Hierbei werden Web Service-Komponenten mittels WSDL als operationale Schnittstellen beschrieben und in UDDI-Verzeichnissen publiziert. Die Interaktion von Partnern im Sinne einzelner Operationen kann dann durch synchrone oder asynchrone Kommunikation von SOAP-Nachrichten erfolgen. Mit diesen Mechanismen ist prinzipiell der organisationsübergreifende Zugriff auf einzelne Anwendungsfunktionen möglich.

Die Mittel der Web Service-Basisarchitektur reichen jedoch noch nicht aus, um mehrfache zusammenhängende Interaktionen zu unterstützen. Auf Anwendungsebene findet nämlich in aller Regel ein wechselseitiger Zugriff auf Anwendungsfunktionen verschiedener Partner statt. Zum einen ergibt sich die gesamte Anwendungsfunktionalität einer Web Service-Komponente erst aus der zusammengenommenen Menge aller ihrer Operationen. Dabei kann dann aber die Sequenz der Operationsaufrufe entscheidend sein. Zum anderen kann der Aufruf einer Operation durch einen Client bei einem Provider den Aufruf einer anderen Operation durch den Provider bei dem Client (Notification oder Solicit-Response) oder bei einem anderen Provider zur Folge haben. Insgesamt zeigt sich also die Bedeutung von Aufruf- bzw. Nachrichtensequenzen an den Schnittstellen eines oder mehrerer Web Services. Die Sequenz von ein- und ausgehenden Nachrichten an der Schnittstelle eines Web Service wird als *Konversation* bezeichnet. Für Abfolgen von Nachrichtentransfers verschiedener Web Services ist hingegen der Begriff der *Choreografie* gebräuchlich.

Konversationen und Choreografien sind in den meisten Fällen nicht in beliebiger Weise möglich oder sinnvoll. Dies kann z. B. an der technischen Realisierung eines Service liegen oder durch Reglementierungen des Anwendungskontextes bedingt sein. D. h. auf der einen Seite, dass zusammenhängende Operationsaufrufe bei einem oder mehreren Web Services nur in bestimmter Reihenfolge erfolgen dürfen. Auf der anderen Seite müssen Clients ihre Aufrufe in eben dieser Reihenfolge durchführen. Die beiden Seiten stehen für zwei Perspektiven auf Konversationen und Choreografien. Die eine Perspektive nimmt einen externen Blickwinkel ein und fokussiert die *Koordination* von Web Services bzw. Operationsaufrufen. Die andere Perspektive nimmt einen internen Blickwinkel ein und fokussiert die *Komposition* von Web Services bzw. Operationsaufrufen.

Koordination bezeichnet im Allgemeinen das Management von Beziehungen. Im Falle von Web Service-Techniken sind dies Beziehungen verschiedener Operationsaufrufe. *Web Service-Koordination* zielt zunächst darauf ab, erlaubte Konversationen und Choreografien als Koordinationsprotokolle zu modellieren. Damit soll dann eine Übereinkunft verschiedener Partner in Bezug auf ihre Choreografie erzielt werden. Schließlich sollen einzelne Partner in die Lage versetzt werden, Operationsaufrufe zu Konversationen zuzuordnen und deren Konformität zu den vereinbarten Koordinationsprotokollen zu überprüfen.

Komposition bezeichnet im Allgemeinen ein Implementierungsprinzip, das auf der Kombination vorhandener Komponenten beruht. Im Falle von WS-Techniken sind die Komponenten dabei durch Web Services gegeben und deren Kombination erfolgt auf Basis von Workflow-Techniken. Das Ziel der *Web Service-Komposition* liegt in einer Unterstützung von Clients bei der Implementierung von komponierten Anwendungssystemen oder Web Services aus WS-Komponenten. Dies beinhaltet Modelle und Werkzeuge zur Entwicklung von Web Service-Kompositionen als Software-Prozess-Schemata. Des Weiteren sollen Laufzeitsysteme eine automatische Ausführung von Kompositionen ermöglichen.

Koordination und Komposition bilden zwei grundsätzlich eigenständige Aspekte von WS-Techniken. Koordinationsprotokolle sind öffentliche Dokumente zur Abstimmung aller Partner. Software-Prozesse zur Komposition sind dagegen in der Regel nicht öffentlich, sondern nur zur Implementierung des Verhaltens einzelner Partner gedacht. Hierbei sind jedoch gewisse Zusammenhänge zu beachten. Kompositionen beeinflussen nämlich unmittelbar die Konversationen und Choreografien der Partner. Daher müssen Kompositionen auch die generellen Vorgaben von Koordinationsprotokollen berücksichtigen. Dies geht so weit, dass Koordinationsprotokolle als direkte Vorgabe für Kompositionen dienen können. Diese sind dann im Sinne eines ausführbaren Prozessschemas zu konkretisieren.

Für die Steuerung von Dienstleistungsprozessen in virt. Unternehmen spielen diese Techniken eine Schlüsselrolle. Mit ihrer Hilfe lassen sich die organisationsübergreifenden Produktionsprozesse als Protokolle darstellen und als Software-Prozesse automatisieren. Dabei erlauben die speziellen Bedingungen der virt. Dienstleistungsunternehmen eine noch direktere Beziehung von Koordination und Komposition, die später noch ausführlich hergeleitet wird.

Zunächst sollen die beiden Aspekte jedoch im generischen Rahmen allgemeiner Web Service-Techniken getrennt eingeführt werden. Der Rest dieser Sektion beschreibt die externe Perspektive der Koordination. Zunächst werden dazu die einzelnen Aspekte der WS-Koordination besprochen. Im Anschluss werden Protokolle, Szenarien und Mechanismen der Koordination im Einzelnen dargestellt. Die interne Perspektive der Komposition wird dann in der nächsten Sektion ausführlich behandelt.

Aspekte der WS-Koordination Um geschäftliche Interaktionen autonomer Unternehmen durch elektronischen Austausch von Nachrichten zu automatisieren, muss der Interaktionsverlauf – also die Abfolge der wechselseitig auszutauschenden Nachrichten – allen Teilnehmern bekannt sein. Dies ist notwendig, um den technischen und geschäftlichen Anforderungen und Regeln gerecht zu werden, die der Transaktion und ihrer elektronischen Implementierung zugrunde liegen. In diesem Sinne wurden im generelleren B2Bi-Kontext schon B2B-Protokolle eingeführt.[126] B2B-Protokolle ermöglichen es, die Menge der erlaubten Nachrichtensequenzen einer Interaktion

[126] Siehe Sektion 3.3.2.4.

auszudrücken. Interaktionspartner können sich dann vor der Interaktion auf ein solches Protokoll einigen, dessen Vorgaben bei der Implementierung des Nachrichtenaustauschs berücksichtigen und während der Interaktion den Interaktionsverlauf auf seine Konformität zum Protokoll prüfen.

Dieses Prinzip kann und muss grundsätzlich auf die Web Service-Architektur übertragen werden. Dabei wird zunächst vom Anwendungskontext abstrahiert. Man spricht hier allgemein von *Koordinationsprotokollen*, die den Verlauf generischer Nachrichtenübertragungen vorgeben. Zur *Web Service-Koordination* muss eine Übereinkunft in Bezug auf Koordinationsprotokolle getroffen werden. Hierfür ist der Inhalt des Protokolls entscheidend: Bestimmte Protokolle sind generisch und brauchen nur einmalig standardisiert zu werden. Ein Beispiel für so ein *horizontales Protokoll* ist etwa der Interaktionsverlauf zur Sicherung von Transaktionalität. Andere Protokolle beziehen sich nur auf die Anwendungsfunktionen spezifischer Web Services und müssen mit diesen zusammen veröffentlicht werden. So ein *vertikales Protokoll* beschreibt z. B. die Interaktion mit dem Web Service einer Transportversicherung.

Aus Koordinationsprotokollen lassen sich die Konversationen der verschiedenen Rollen einer Interaktion ableiten. Zur Realisierung der Interaktion müssen die Partner dann jeweils eine interne Konversationslogik implementieren, um den mit ihrer Rolle verbundenen Nachrichtenaustausch zu steuern. Die Implementierung erfolgt zumeist durch klassische Programmierung. Es können jedoch auch spezifische Techniken der WS-Komposition verwendet werden, die weiter unten noch genauer beschrieben werden. Besonders für vertikale Protokolle ist meist eine individuelle Implementierung durch die Interaktionspartner notwendig. Dazu stellt WS-Middleware in der Regel verschiedene Unterstützungsmechanismen bereit. Die *Konversationskontrolle* (realisiert durch so genannte *Conversation Controller*) erlaubt es einem WS, parallele Konversationen zu führen. Dazu werden eingehende Nachrichten automatisch den laufenden Konversationen zugeordnet und an die entsprechenden Instanzen der Konversationslogik weitergeleitet. Des Weiteren können durch Konversationskontrolle die Nachrichten einer laufenden Konversation auf Konformität zu einem Koordinationsprotokoll geprüft und ggf. Fehler gemeldet werden. Die generischen horizontalen Protokolle sind hingegen oft direkt als Komponenten einer WS-Middleware realisiert. Diese *Protokollabwicklung* (realisiert durch so genannte *Protocol Handler*) implementiert Standards zur Koordination von Partnern in Bezug auf generische externe Middleware-Mechanismen. Hierdurch werden Eigenschaften und Abstraktionen der WS-Middleware auf Anwendungsebene in transparenter Weise realisiert. Auf dem aktuellen Stand der Technik gilt das vor allem für Konversationskontrolle und Transaktionalität.

WS-Koordinationsprotokolle Das erste Problem der Web Service-Koordination besteht darin, Koordinationsprotokolle spezifizieren zu können. Dazu sind entsprechende Modelle und Sprachen notwendig. Gewisse Eingaben kommen hier von bestehenden

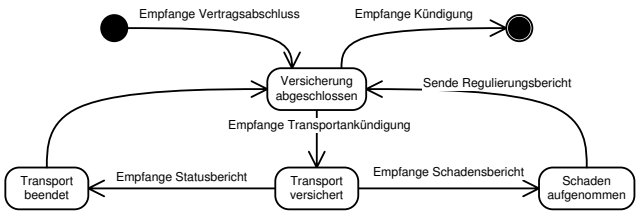

Abbildung 3.25.: Protokoll einer Konversation als UML-Zustandsdiagramm

Ansätzen des B2Bi. Inzwischen sind auch im direkten Kontext der Web Service-Architektur verschiedene Ansätze entwickelt worden. Obwohl daraus schon Vorschläge zur Standardisierung hervorgegangen sind, hat sich bislang kein Ansatz durchsetzen können. Allgemein kann festgestellt werden, dass die Ansätze im Wesentlichen auf den bekannten Modellen zur Verhaltensbeschreibung von Software-Systemen beruhen. Vor allem finden sich hier spezielle Formen der UML-Verhaltensdiagramme: Es kommen Zustands-, Sequenz- und Aktivitätsdiagramme vor.

Modelle auf Basis endlicher Automaten sind schon länger zur Verhaltensbeschreibung von Software-Komponenten bekannt [GZMW01]. In Übertragung auf den WS-Kontext dienen sie meist zur Beschreibung der zulässigen Konversationen mit einzelnen Web Services. Derartige Modelle heben insbesondere die zustandsbehaftete Natur von Konversationen hervor. Alle möglichen Zustände im Laufe einer Konversation werden explizit modelliert. Jede Interaktion zwischen Web Services führt zu einem Zustandsübergang. Durch Start- und Endzustände wird der Gesamtverlauf kenntlich gemacht. Abbildung 3.25 macht das Prinzip anhand der Konversation eines beispielhaften Web Service zur Versicherung von Transporten deutlich. Zustandsübergänge sind dabei jeweils mit dem Austausch einer Nachricht zwischen Versicherer und Client verbunden. Es wird hier ersichtlich, dass zunächst ein Versicherungsvertrag abgeschlossen werden muss und dann jeder einzelne Transport zu registrieren ist, um im Problemfall eine Schadensregulierung zu veranlassen, bis die Versicherung schließlich beendet wird. Auf Basis dieses Grundkonzepts wurden diverse Erweiterungen und Spezialisierungen vorgeschlagen. Ein Standardisierungsvorschlag dieser Art ist die Web Service Conversation Language (WSCL) [BBB+02], eine XML-Sprache zur Erweiterung von WSDL-Beschreibungen durch Automaten, deren Zustandsübergänge mit WSDL-Operationen beschriftet werden [KLK+01]. Allen solchen Ansätzen gemeinsam ist die gute Eignung zur Überwachung von Konversationen. Monitoring-Werkzeuge können leicht den momentanen Zustand einer Konversation erkennen.

Auch Sequenzdiagramme können zur Beschreibung von WS-Konversationen eingesetzt werden (Ein solches Diagramm ist in Abb. 3.11 rechts auf S. 120 zu sehen). Diese Diagramme stellen die sequenzielle Abfolge von Interaktionen zwischen bestimmten Klassen/Rollen dar und berücksichtigen dabei unterschiedliche Kommunikationsformen. Zudem können hier auch Choreografien mit mehr als zwei Partnern in anschaulicher Weise erfasst werden. In der Praxis werden Sequenzdiagramme zur

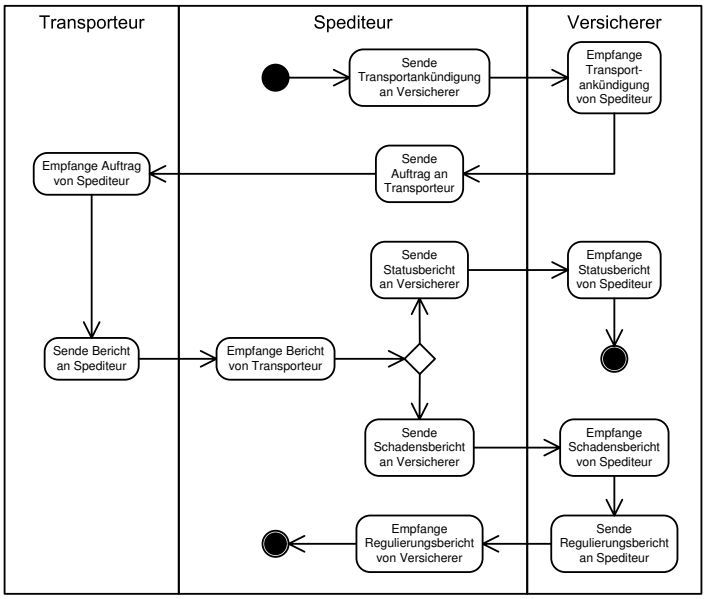

Abbildung 3.26.: Protokoll einer Choreografie als UML-Aktivitätsdiagramm

Spezifikation von einfachen Protokollen eingesetzt. Zur Darstellung komplexer Interaktionsverläufe mit Schleifen und Alternativen werden jedoch mehrere Diagramme benötigt. Solche Interaktionsprozesse lassen sich mit Aktivitätsdiagrammen intuitiver darstellen. Hierbei werden Interaktionen als einzelne Blöcke repräsentiert. Der Prozessverlauf ergibt sich durch Kanten, die insbesondere auch alternative und nebenläufige Verzweigungen beinhalten können. Abbildung 3.26 zeigt dies anhand der Choreografie zwischen der schon bekannten Transportversicherung und einem Klienten, der in diesem Fall ein Spediteur ist und seinerseits mit dem Web Service eines Transporteurs interagiert.

An dem Beispiel zeigt sich, wie intuitiv die Prozessdarstellung des Aktivitätsdiagramms die globale Perspektive einer Choreografie zu zeigen vermag. Äquivalent dazu kann der Sachverhalt auch als eine Menge von Konversationen aus den jeweiligen Perspektiven der einzelnen Rollen dargestellt werden. Bei Verwendung von Aktivitätsdiagrammen stellen sich solche Konversationen als so genannte *öffentliche Prozesse* dar. Sie zeigen nämlich den Prozess einer Rolle, wie er von außen sichtbar ist. Derartige Prozesse sind in der Regel nichtdeterministisch, da sich die konkrete Logik einzelner Schritte und Entscheidungen von außen nicht erschließt. Aus diesem Grund werden öffentliche Prozesse auch als *abstrakt* bezeichnet. Für Koordinationsprotokolle auf Basis abstrakter Prozesse sind zurzeit zwei konkurrierende Standards vorgeschlagen worden. Dies ist zum einen die Business Process Execution Language (BPEL), die eigentlich der Web Service-Komposition dient und die Spezifikation von Koordinationsprotokollen nur als Option vorsieht [ACD+03]. Der zweite Standard ist das Web Service Choreography Interface (WSCI) [Ark02]. Beide Ansätze erweitern das

Grundprinzip der Aktivitätsdiagramme in erheblichem Maße durch Abstraktionen wie Transaktionen und Fehlerbehandlung. Welche Spezifikation sich durchsetzen wird, ist zurzeit nicht absehbar.

Szenarien der WS-Koordination Der Nutzen von Koordinationsprotokollen für die Entwicklung und den Betrieb von Service-orient. Anwendungssystemen hängt von der Art des Protokolls ab. Vertikale Koordinationsprotokolle beschreiben Interaktionen auf Anwendungsebene. Mit ihrer Hilfe können Provider die geschäftliche Interaktion mit ihren Web Services als Teil der Schnittstelle beschreiben. Dadurch ermöglichen sie Clients die fehlerfreie Service-Nutzung und sich selbst deren automatisierte Überwachungen. Zum Teil werden vertikalen Protokolle in Standards einzelner Branchen vorgegeben. Dies ist z. B. beim RosettaNet-Standard für die IT-Industrie der Fall [Ros04].

Vertikale Koordinationsprotokolle können sowohl zum Auffinden als auch zur dynamischen Bindung von Web Services herangezogen werden. Dazu müssen sie als Teil der Service-Spezifikation (z. B. in UDDI-Verzeichnissen) publiziert werden. Entwickler können dann die Vorgaben des Protokolls bei der Interaktionslogik eines WS-Clients einfließen lassen. Zur dynamischen Bindung kann dann eine Suche nach konkreten Web Services durch das Protokoll eingeschränkt werden. Für diese Art der Verwendung sollten generell nur Protokolle von Konversationen und nicht von Choreografien verwendet werden. Für den Client ist nämlich nur die direkte Interaktion mit einer Rolle von Bedeutung. Die Interaktionen anderer Rollen, die unter Umständen noch Teil einer Choreografie sein können, sind unerheblich und sollten die Auswahl nicht beeinflussen.

Im Gegensatz zu den vertikalen Protokollen sind die horizontalen Protokolle auf der Anwendungsebene weitgehend transparent. Sie dienen der Vorgabe von Interaktionen externer WS-Middleware-Komponenten der Interaktionspartner. Solche Interaktionen dienen zum einen als Metaprotokolle zur Herstellung einer grundsätzlichen Konnektivität verschiedener Plattformen. Zum anderen realisieren sie Middleware-Abstraktionen wie verlässliche Kommunikation oder transaktionale Garantien. Diese Abstraktionen werden dann auf der Anwendungsebene, z. B. bei der Implementierung der geschäftlichen Interaktionslogik, verwendet. Dass dann zur Laufzeit entsprechende vertikale Protokolle zwischen den Middleware-Plattformen abgewickelt werden, bleibt auf der Anwendungsebene in der Regel verborgen. Konkret bedeutet das, dass zur Laufzeit vertikale und horizontale Protokolle gleichzeitig abgewickelt werden: Während über die vertikalen Protokolle die geschäftliche Interaktion vollzogen wird, laufen zu deren Unterstützung Interaktionen der Middleware-Plattformen ab. Wenn dann z. B. auf geschäftlicher Ebene der Abbruch einer Transaktion beschlossen wird, regeln die Middleware-Plattformen die Wiederherstellung eines konsistenten Zustands in transparenter und automatischer Weise durch Abwicklung z. B. eines 2PC-Protokolls.

Web Service Coordination Middleware Unabhängig von der Art des Koordinationsprotokolls erschwert die Berücksichtigung zusammenhängender Aufrufe die Implementierung von Web Services. Durch Konversationen entsteht eine zeitlich andauernde Beziehung zwischen Provider und Client, deren Zustand gespeichert werden muss. Da die Ports eines Web Service über jeweils eine Adresse angesprochen werden, ist zur Unterscheidung verschiedener Clients, deren Konversationen sich überschneiden, eine zusätzliche Logik erforderlich. Diese immer wiederkehrende Aufgabe kann zur Erleichterung der Web Service-Entwicklung auf der Ebene der WS-Middleware durch eine Konversationskontrolle realisiert werden [KL01]. Das Prinzip besteht darin, dass die Implementierung des Web Service pro Konversation eine eigene Instanz der Konversationslogik erzeugt. Die Konversationskontrolle ergänzt dann die versendeten Nachrichten mit einer Kennung der Konversation. Anhand dieser Kennung werden Nachrichten an die zuständigen Instanzen der Konversationslogik weitergeleitet. Eine weitere Unterstützungsfunktion der Konversationskontrolle kann darin bestehen, das Koordinationsprotokoll zu interpretieren und mit der laufenden Konversation zu vergleichen. Bei Abweichungen wird dann ein Fehler gemeldet. Abbildung 3.27 zeigt, wie sich die Konversationskontrolle *(Conversation Controller)* in die WS-Middleware eingliedert. Es sei noch bemerkt, dass Konversationskontrolle grundsätzlich eine Standardisierung notwendig macht. Nur so können die beteiligten Partner die Kennzeichnung von Konversationen auf einheitliche Weise durchführen und das Koordinationsprotokoll interpretieren.

Als weitere Middleware-Komponente zeigt die Abbildung Komponenten zur automatischen Protokollabwicklung (Protocol Handler). Konzeptionell können diese mit der Konversationskontrolle verbunden werden, da die Implementierung der Konversationslogik für Letztere unerheblich ist. Protokollabwicklung ist prinzipiell für immer wiederkehrende horizontale Protokolle sinnvoll, da diese für beliebige Web Services immer wieder gleich ablaufen. Aktuell sind hier Metaprotokolle vorgeschlagen worden, die eine Infrastruktur zur Ausführung horizontaler Middleware-Protokolle realisieren. Ein grundsätzliches Problem besteht nämlich in der anfänglichen Einigung von Partnern auf das auszuführende Protokoll, ihre jeweiligen Rollen darin, sowie die Adressen ihrer Ports. *WS-Coordination* beschreibt in diesem Sinne eine einheitliche Vorgehensweise [CCF+05c]. Zudem wurden Middleware-Protokolle zur Gewährleistung transaktionaler Garantien entwickelt, die auf dem Metaprotokoll aufsetzen. Im Rahmen von *WS-Transaction* sind in diesem Zusammenhang verschiedene Protokolle enthalten. Diese beinhalten zunächst die Realisierung klassischer Transaktionen [CCF+05a] mit den bekannten ACID-Eigenschaften durch das 2PC-Protokoll. Darüber hinaus werden aber auch längerfristige geschäftliche Aktivitäten berücksichtigt, bei denen die Isolationseigenschaften eine abgeschwächte Form aufweisen [CCF+05b]. Da in der vorliegenden Arbeit die Unterstützung geschäftlicher Prozesse auf Anwendungsebene untersucht wird, soll die Betrachtung horizontaler Koordinationsprotokolle mit diesem kurzen Einblick enden.

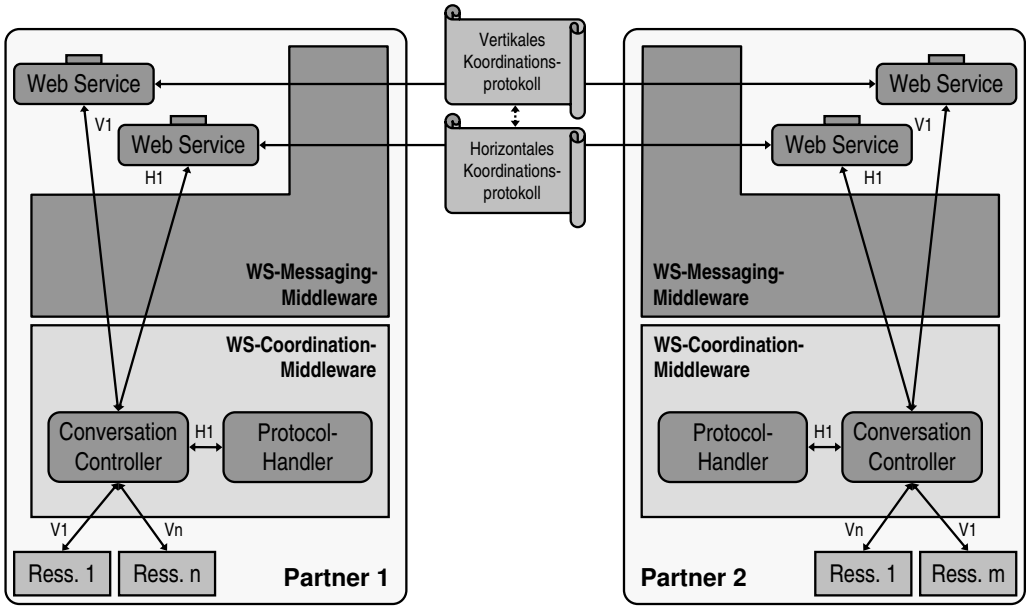

Abbildung 3.27.: Middleware-Mechanismen zur Koordination von Web Services

3.4.2.3. Implementierung geregelter WS-Interaktionen

Im letzten Abschnitt wurde dargestellt, dass bei der Interaktion mit Web Services die Reihenfolge der wechselseitigen Nachrichtenübertragungen bzw. Operationsaufrufe von entscheidender Bedeutung ist. Nur bestimmte Sequenzen sind sinnvoll und erlaubt. Valide Choreografien bzw. die damit einhergehenden Konversationen einzelner Partner werden durch Koordinationsprotokolle öffentlich vorgegeben. Das hat zur Folge, dass die Interaktionslogik einzelner Partner komplex werden kann – besonders, wenn ein Partner gleichzeitig verschiedene Konversationen führt. Die Implementierung verschiedener Konversationen durch einen Client wird im Allgemeinen als *Web Service-Komposition* bezeichnet. Die Partner der einzelnen Konversationen steuern dabei *Web Service-Komponenten* bei. Der Client führt mit diesen simultane Konversationen und macht das Ergebnis in der Regel wiederum als *komponierten Web Service* verfügbar. Für den Client ist es dabei unerheblich, ob die WS-Komponenten selber komponiert sind. Die WS-Komposition kann daher mehrstufig erfolgen und in jeder Stufe die Komplexität der jeweiligen Konversationen verbergen. Die Implementierung kann dadurch Schritt für Schritt erfolgen und die Komplexität bleibt unter Kontrolle. Insgesamt bezeichnet WS-Komposition also eine Implementierungsstrategie für WS-Clients.[127]

[127]Es ist zu beachten, dass Service-Komposition in verschiedener Hinsicht wesentliche Unterschiede zum Kompositionsbegriff der komponentenorientierten Software-Technik aufweist [YHP02]. Zunächst handelt es sich nicht um eine physische Komposition. Die WS-Komponenten bleiben in räumlicher Hinsicht stets vom komponierten Web Service getrennt. Zudem kann sich

Der unmittelbare Ansatz zur Realisierung von WS-Kompositionen besteht in deren Programmierung mithilfe einer meist objektorientierten Programmiersprache.[128] Dieser Weg entspricht der üblichen Vorgehensweise klassischer Middleware und EAI-Plattformen. Tatsächlich stehen eine Reihe von Werkzeugen und API bereit, um die Abstraktionen der grundlegenden WS-Architektur programmatisch zu verwenden. Die Kombination verschiedener Web Services durch ein Programm ist daher aus technischer Sicht kein Problem. Allerdings geht dies auch mit erheblichen Nachteilen einher. Der überwiegende Teil solcher Programme beschäftigt sich nämlich mit der technischen Einbindung und Verwendung der WS-Middleware-Mechanismen, was sich in der Regel sehr komplex gestaltet. Die geschäftliche Logik der Komposition nimmt nur einen vergleichsweise geringen Anteil ein und ist über den sonstigen Code verteilt. Dementsprechend ist der Kernaspekt der Komposition umständlich zu implementieren, schwierig zu erkennen und aufwendig zu warten. Hinzu kommt, dass klassische Programmiersprachen keine spezifischen Abstraktionen (z. B. Rollenbegriff, Interaktionsmuster, geschäftliche Transaktionen etc.) bieten, die bei der Implementierung von Kompositionslogik helfen könnten.

Eine mögliche Lösung dieses Problems besteht darin, die spezifischen Aspekte der WS-Komposition auf eine höhere Abstraktionsebene anzuheben. Auf dieser Ebene wird dann die Kompositionslogik explizit und exklusiv behandelt. Spezifische Sprachen zur Spezifikation komponierter Web Services bilden dabei die Basis zur systemtechnischen Unterstützung, etwa beim Auffinden und Einbinden geeigneter Provider und Clients sowie bei der Steuerung und Kontrolle von Interaktionen. Die Repräsentation komponierter Web Services basiert dabei auf dem Konzept der *Orchestrierung*: Hierbei werden die Präzedenzen von WS-Interaktionen sowie die Abhängigkeiten der dabei verwendeten Parameter als Prozess spezifiziert. Diese Prozessbeschreibung repräsentiert die Kompositionslogik und dient im Weiteren als Instruktion zur automatisierten Ausführung. Die Konzepte der hier verwendeten Orchestrierungssprachen basieren dabei meist auf denen von Workflows, was eine Unterstützung durch entsprechend angepasste WfMS ermöglicht [GL02].

Web Service Composition Middleware Aus technischer Sicht wird die Ebene der Web Service-Komposition durch eine spezifische Klasse von Middleware realisiert: *Web Service Composition Middleware* stellt eine Infrastruktur bereit, die mit spezifischen Abstraktionen und Mechanismen die Implementierung von Web Service-Clients im Allgemeinen und von komponierten Web Services im Speziellen unterstützt. Es handelt sich hierbei also um eine Implementierungstechnik, die im internen Bereich eines Clients oder Aggregators zum Einsatz kommt. Konkret bildet Web Service

sowohl das Angebot an WS-Komponenten als auch die Nachfrage nach und die Anforderung an komponierte Web Services rasch ändern. WS-Komposition muss also grundsätzlich flexibel sein und einen gewissen Grad an Dynamik beinhalten.

[128] Hierzu werden z. B. oft die Sprachen C# oder Java verwendet.

Composition Middleware eine Plattform, die in drei Bereiche unterteilt werden kann [ACK+04, S. 249 f.]:

- *Kompositionsmodell* – erfasst das konzeptionelle Rahmenwerk einer Kompositionsplattform. Hierin werden die spezifischen Abstraktionen festgelegt, mit denen komponierte Web Services modelliert und konstruiert werden können. Insbesondere wird festgelegt, welche Anforderungen an WS-Komponenten gestellt werden, wie die Aufrufreihenfolge und -parameter (bzw. Nachrichten) dieser WS-Komponenten definiert werden können und wie das Binden von Providern der WS-Komponenten erfolgt. Eine *Kompositionssprache* stellt Konstrukte bereit, um komponierte WS im Sinne des Kompositionsmodells zu spezifizieren. Diese Spezifikationen werden als Kompositionsschemata bezeichnet. Sie werden mithilfe einer Entwicklungsumgebung erstellt und können in einer Laufzeitumgebung ausgeführt werden.

- *Entwicklungsumgebung* – beinhaltet meist interaktive Werkzeuge zur Unterstützung des Lebenszyklusses komponierter WS. Dazu gehören meist interaktive Modellierungswerkzeuge zum Entwurf der geschäftlichen Logik von Kompositionen. Zu deren Validierung sind oft *Analysewerkzeuge* enthalten. Darüber hinaus finden sich von Fall zu Fall unterschiedliche Mechanismen zur *Konstruktion*, zur *Optimierung*, zum *Testen* und zum *Einsatz (Deployment)* ausführbarer Kompositionsschemata, zur *Bereitstellung (Provisioning)* der dadurch realisierten (komponierten) Web Services und schließlich zur *Wartung (Maintenance)* wie auch zur *Weiterentwicklung* von Kompositionen im Anschluss an die Ausführung.

- *Laufzeitumgebung* – stellt Mechanismen zur Verfügung, um komponierte WS auszuführen. Der wichtigste Bestandteil sind *Composition Engines*. Diese werten bei der Aktivierung eines komponierten WS dessen Kompositionsschema aus, führen die dort spezifizierte geschäftliche Logik aus und rufen die WS-Komponenten auf. Für jeden Client, der den komponierten WS nutzt, wird eine eigene *Kompositionsinstanz* vorgehalten.

Während Programmiersprachen-basierte Lösungen zur WS-Komposition auf der Ebene klassischer Middleware operieren, bildet Web Service Composition Middleware eine Erweiterung der Web Service Middleware. Sie bildet dort ein Komplement zur Web Service Coordination Middleware und basiert zusammen mit dieser auf den Mechanismen der grundlegenden WS-Architektur (siehe Abb. 3.28). Die einzelnen Bereiche können so optimal aufeinander abgestimmt werden und sich wechselseitig ergänzen. Dies ist Z. B. am Zusammenspiel der Laufzeitumgebung für Kompositionen und der allgemeinen Konversationskontrolle ersichtlich: Während die Composition Engine die Konversation mit verschiedenen Partnern antreibt, kann die Konversationskontrolle die Einhaltung der verschiedenen Koordinationsprotokolle überwachen und

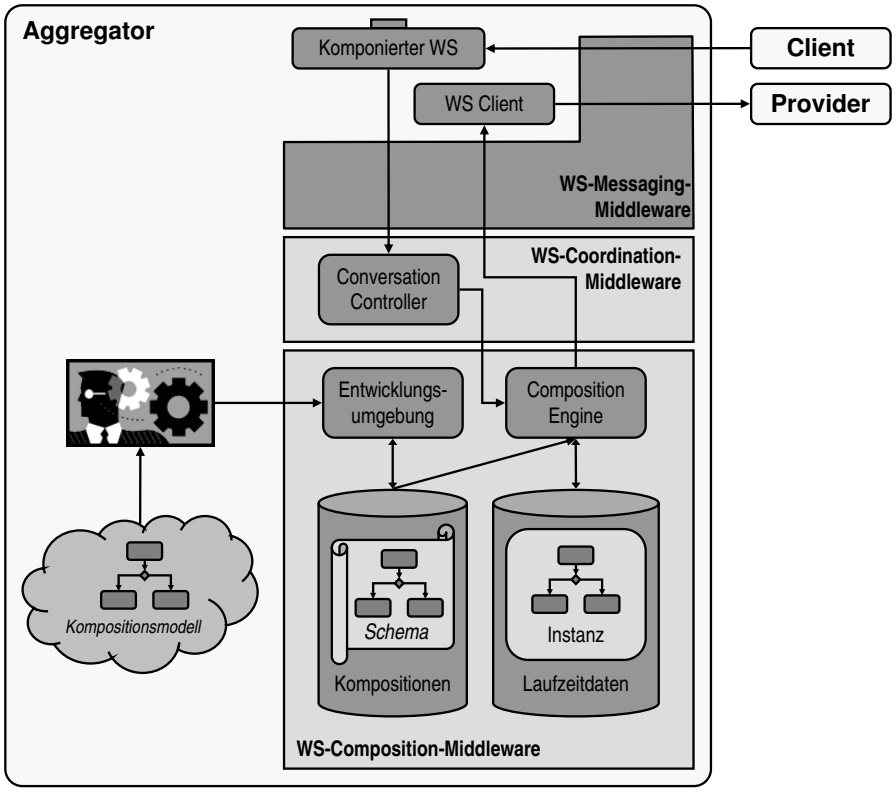

Abbildung 3.28.: Middleware-Mechanismen zur Komposition von Web Services

die Engine durch das Routen von Konversationen bei der Verwaltung verschiedener Kompositionsinstanzen unterstützen.

Aus übergeordneter Perspektive genereller Middleware-Techniken stellt sich das Prinzip der Web Service-Komposition als Übertragung der Workflow-basierten Systemintegration dar. Dieser Ansatz wurde schon wesentlich früher im EAI-Kontext vorgeschlagen und ist dort nur bedingt erfolgreich gewesen. Der Grund dafür ist u. a. darin zu suchen, dass im EAI-Kontext weder ein einheitliches Komponentenmodell noch ein dominantes Kompositionsmodell vorhanden ist. Zur Maximierung der Flexibilität in Bezug auf die Integration beliebiger Anwendungssysteme wurde absichtlich kein einheitliches Komponentenmodell vorgeschrieben. Stattdessen wird die Interaktion sehr unterschiedlicher Software-Elemente über stets spezifische Adapter geregelt und durch Message Broker nur unwesentlich erleichtert. Auch das Kompositionsmodell ist bei den meisten WfMS individuell. Spezifische Integrationslösungen sind daher komplex, aufwendig zu realisieren und zu warten sowie praktisch nicht zu portieren. Versuche, hier durch Standardisierung Abhilfe zu schaffen, haben bislang nur wenig Besserung gebracht.[129] Web Service-Komposition hat das Potenzial,

[129] Z. B. geht das weit verbreitete und akzeptierte Komponentenmodell von CORBA [OMG04] mit einer für EAI unpassenden programmatischen Komposition einher. Die Workflow-Standards

die Schwachstellen der Workflow-basierten Integration zum Teil zu kompensieren. Web Services bilden dabei zunächst ein passendes, einheitliches und standardisiertes Komponentenmodell. Zudem zeichnet sich eine Fokussierung auf wenige Kompositionsmodelle ab.[130] Die Konzentration auf wenige einfache Techniken lässt auf eine schnellere Entstehung und Reifung komfortabler Middleware-Techniken hoffen. Diese könnten dann den Aufwand bei der (Weiter-)Entwicklung von Kompositionen und der Einbeziehung neuer Komponenten nachhaltig verringern. Hierbei spielt nicht nur die Verbreitung von WS, sondern auch das schnellere Erlernen weniger Kompositionsmodelle zur Implementierung portierbarer Lösungen auf einer Vielzahl verschiedener Plattformen eine Rolle.

WS-Kompositionsmodelle Die konkreten Fähigkeiten und Eigenschaften einer Web Service Composition Middleware werden maßgeblich durch ihr Kompositionsmodell bestimmt. Zur differenzierten Betrachtung der dabei festgelegten Aspekte können Kompositionsmodelle in eine Reihe von Dimensionen unterteilt werden [ACK+04, S. 256 ff.]:

- *Komponentenmodell* – legt fest, was die Bausteine der Komposition sind und welche Voraussetzungen diese erfüllen müssen.
- *Koordinationsmodelle* – bestimmen einen Rahmen zur Spezifikation von Präzedenzen zwischen einzelnen Interaktionen mit Komponenten.
  - *Orchestrierungsmodell* – regelt d. Spezifikation v. Interaktionssequenzen.
  - *Datenmodell* – gibt den Rahmen zur Datenmodellierung vor und regelt den gemeinsamen Speicherzugriff verschiedener Interaktionen.
- *Aggregationsmodell* – bestimmt den Mechanismus zum Binden von Providern.

Das *Komponentenmodell* der WS-Komposition wird im Allgemeinen eng am Begriff des Web Service angelehnt. Der WS-Begriff umfasst jedoch (im technischen Sinne) ein Kontinuum möglicher Eigenschaften. Dabei kann aus einer Vielzahl ergänzender und konkurrierender Standards gewählt werden. Konkrete Web Services unterscheiden sich in der Menge der von ihnen unterstützten Standards und in deren Anwendung. Jedes Software-Element mit beliebiger Möglichkeit zur Nachrichtenkommunikation kann durch Beschreibung mittels einer WSDL-Variante schon als Web Service gelten. Die potenzielle Heterogenität solcher Web Services macht deren Komposition jedoch schwierig. Zur Vereinfachung können zusätzliche WS-Standards wie z. B. SOAP

der WfMC [WfM04] haben sich hingegen in der Praxis nicht durchsetzen können.

[130] Der Fokus liegt zurzeit auf der Business Process Execution Language (BPEL). Es sei aber bemerkt, dass die momentan diskutierten Kompositionsmodelle nicht unumstritten sind [Aal03a, ADH03].

zur Nachrichtenkommunikation und WS-Coordination/-Atomic Transaction als horizontale Koordinationsprotokolle vorausgesetzt werden. Hierdurch verringert sich jedoch die Universalität des Kompositionsmodells, da solche Voraussetzungen von wenigen Web Services erfüllt werden. Bei der Gestaltung des Komponentenmodells besteht die Aufgabe nun darin, Voraussetzungen an WS-Komponenten zu definieren. Dabei ist ein geeigneter Kompromiss zwischen Spezialisierung und Universalität der WS-Komponenten zu finden. Dieser Kompromiss muss dem geplanten Einsatz des Kompositionsmodells gerecht werden.

Der zentrale Aspekt des Kompositionsmodells liegt nach Festlegung der Komponenten im Konzept für deren Zusammensetzung. Bildlich gesprochen wird hier der Klebstoff bestimmt, der komponierte Web Services zusammenhält.[131] Der Fokus liegt aber weniger auf der (Wieder-)Verwendung vorgegebene Komponenten – wie dies bei der komponentenbasierten Software-Konstruktion der Fall ist. Vielmehr steht – Analog zu den Konzepten des Workflow – der Vorgang der Zusammensetzung selbst im Mittelpunkt. Dieser soll einen vorhandenen (geschäftlichen) Prozess widerspiegeln und die WS-Komponenten entsprechend koordinieren. Das bedeutet, dass die Beziehungen zwischen deren Aktivitäten zu regeln sind. Auf Aktivitäten der WS-Komponenten kann dabei nur indirekt über atomare nachrichtenbasierte Interaktionen (d. h. durch Aufruf von WSDL-Operationen) Einfluss genommen werden. Entsprechend sind für komponierte Web Services die Beziehungen zwischen Aufrufen von deren WSDL-Operationen zu regeln. Hierbei werden meist allgemeine Präzedenzen (Kontrollfluss) und Datenbeziehungen (Datenfluss) in zwei zusammenhängenden *Koordinationsmodellen* behandelt: Ein Orchestrierungsmodell erlaubt die Spezifikation von Aufrufsequenzen für WSDL-Operationen. Ein Datenmodell bestimmt den Rahmen zur Modellierung von Daten und zum gemeinsamen Zugriff darauf bei verschiedenen Operationsaufrufen.

Das *Orchestrierungsmodell* gibt vor, wie die Ausführungsreihenfolge von WSDL-Operationsaufrufen bei den verschiedenen WS-Komponenten eines komponierten Web Service spezifiziert werden kann. Eine verbreitete Klasse von diesbezüglichen Ansätzen beruht auf Varianten von Flussdiagrammen im Allgemeinen und speziell von Aktivitätsdiagrammen. In Anspielung auf die Struktur werden solche Ansätze zum Teil auch als *Graph-basiert* bezeichnet. Zur Veranschaulichung soll das in Abb. 3.29 gezeigte Beispiel dienen. Es zeigt das Kompositionsschema eines komponierten Web Service, der sich gemäß der schon im letzten Abschnitt verwendeten Choreografie verhält (siehe Abb. 3.26) und dort die Rolle des Spediteurs implementiert. Aktivitäten sind in diesem Fall Interaktionen mit Web Services, die andere Rollen der Choreografie realisieren. Hierbei wird entweder eine Nachricht gesendet, eine Nachricht empfangen oder eine Operation synchron aufgerufen. Das Senden von Nachrichten erfolgt asynchron und blockiert die Ausführung nicht. Beim Empfangen einer Nachricht oder dem synchronen Aufruf einer Operation blockiert die Ausführung hingegen

[131] Das Bild des *Klebstoffes (Glue)* entstammt dem Bereich der Software-Komponenten [NL97].

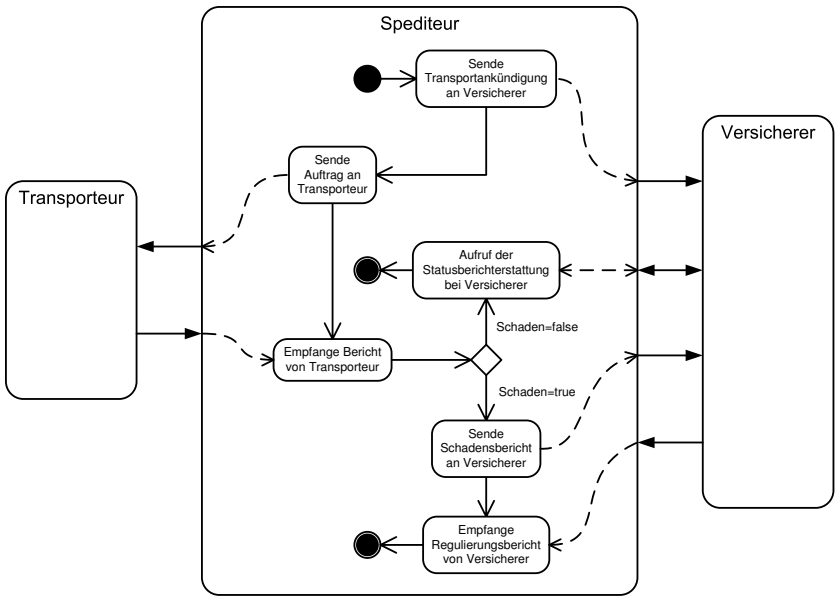

Abbildung 3.29.: Kompositionsschema als UML-Aktivitätsdiagramm

bis zum Empfangsereignis. Falls das Senden einer Nachricht einen asynchronen Operationsaufruf einleitet, dann folgt im weiteren Verlauf auch ein entsprechender Empfang der Antwortnachricht. Je nach konkretem Orchestrierungsmodell können zudem weitere Aktivitäten vorhanden sein, die z. B. Daten manipulieren oder die Bindung von Providern beeinflussen (s. u.). Im direkten Vergleich ähnelt das Kompositionsschema dem Koordinationsprotokoll.[132] Tatsächlich liegt der Unterschied zum öffentlichen Prozess des Spediteurs nur im Detaillierungsgrad der Spezifikation. Da das Kompositionsschema ausführbar sein soll, muss die Spezifikation im Gegensatz zum Koordinationsprotokoll deterministisch sein. Die Verwendung von Flussdiagrammen bildet hierzu ein intuitives Instrument, das sich schon bei diversen WfMS bewährt hat.

Weitere Orchestrierungsmodelle basieren u. a. auf Zustandsdiagrammen, Petrinetzen, dem π-Calculus und regelbasierten Ansätzen, um nur einige der wichtigsten Möglichkeiten zu nennen.[133] Erweiterte *Zustandsdiagramme* finden sich z. B. in der UML. In entsprechenden Orchestrierungsmodellen kann das Eintreten, Verlassen und der Übergang zwischen den Zuständen mit Aktivitäten annotiert werden, die im Wesentlichen Interaktionen sind. Zudem können Bedingungen oder auslösende Ereignisse für den Übergang zwischen den Zuständen definiert werden. In Analogie zu entsprechenden Koordinationsprotokollen erleichtert die explizite Modellierung

[132] Dies ist schon im Vergleich zur Choreografie zu erkennen. Eine noch größere Ähnlichkeit besteht zum öffentlichen Prozess des Spediteurs (hier nicht gezeigt).

[133] Die Ausführung lehnt sich an Alonso et al. an [ACK+04]. Dort und auch bei Milanovic [MM04] finden sich noch weitere Möglichkeiten für Orchestrierungsmodelle.

von Zuständen das Monitoring komponierter Web Services.[134] *Petrinetze* bilden ganz allgemein ein äußerst vielseitiges Instrument zum Umgang mit Prozessen, das die Begriffe von Aktivitäten und Zuständen verbindet. Die Vorteile liegen in einer formalen Basis mit eindeutiger Semantik und einem darauf begründeten immensen Methodenapparat. Vielfältige Erweiterungen erlauben die Anwendung in fast beliebigen Kontexten praktischer und angewandter Art. Petrinetz-basierte Orchestrierungsmodelle erlauben die exakte Modellierung von Kompositionsschemata und bieten vielfältige Möglichkeiten der Validierung [HB03, MPP02]. Hierbei hat sich auch die Verwendung von Referenznetzen als vorteilhaft erwiesen [Off03]. Für diese existieren leistungsfähige Engines zur Ausführung von Kompositionsschemata.[135] Ein weiteres formales Instrument, das sich als Basis für Orchestrierungsmodelle eignet, ist Prozessalgebra. Hierbei wurde besonders der π-Calculus berücksichtigt, der auf *Communicating Sequential Processes (CSP)*, *Algebra of Communicating Processes with Abstractions (ACP)* und *Calculus of Concurrent Systems (CCS)* zurückgeht [SW01].[136] Vorteil ist – wie auch bei Petrinetzen – der Hintergrund theoretischen Wissens und die damit einhergehenden Möglichkeiten zur Validierung von Ergebnissen. In Bezug auf Orchestrierung bietet das Kalkül verschiedene Operatoren an, mit denen die sequenzielle parallele oder konditionale Ausführung von Interaktionen spezifiziert werden kann und die sich beliebig komplex verschachteln lassen. Schließlich sind auch regelbasierte Ansätze möglich. Das Konzept betrachtet eine Composition Engine als reaktives System, das Ereignisse erkennt (z. B. der Eingang einer Nachricht) anhand der aktuellen Situation verschiedene Bedingungen auswertet und daraufhin ggf. Aktivitäten ausführt (z. B. eine Nachricht versenden). Das Verhalten wird dabei durch eine Menge von Event-Condition-Action (ECA)-Regeln bestimmt. Da diese unabhängig voneinander Aktivitäten nur auf Basis ihrer Bedingungen ausführen, eignet sich der Ansatz gut zur Behandlung von Ereignissen und Ausnahmesituationen. Da die Regeln meist aber auch unstrukturiert sind, können derartige Systeme leicht zu unübersichtlichen Lösungen führen. Sie sind daher nur wenig verbreitet.[137]

Im Rahmen von Orchestrierungsmodellen müssen stets auch Daten berücksichtigt werden. Dabei ist zum einen von Interesse, wie diese Daten im Rahmen der Kompositionsschemata modelliert werden können. Zum anderen hat die Art des gemeinsamen Zugriffs auf Daten Konsequenzen für das Verhalten des komponierten WS. Beide Aspekte werden im *Datenmodell* bestimmt.

In Bezug auf die Datenmodellierung kann man zunächst grob *Anwendungsdaten* und ablaufrelevante Daten (bzw. *Prozessvariablen*) unterscheiden. Anwendungsdaten sind diejenigen, die den Inhalt von Nachrichten zwischen den WS-Komponenten

[134] Ein bekanntes Beispiel ist Mentor [MWW+98] (siehe Sektion 3.3.3).

[135] Eine Engine für Referenznetze wird z. B. durch das Renew-System realisiert [KWD+04].

[136] Hierauf basieren z. B. die Kompositionsmodelle XLANG [Tha01] und BPEL [ACD+03].

[137] Eine deklarative Sprache auf Basis von ECA-Regeln wird z. B. in *WEBBIS (WebBase of Internet-accessible Services)* [BMB+00] zur Service-Komposition verwendet. Die meisten Beispiele finden sich aber in Form von DB-Erweiterungen für WfMS (siehe z. B. Kappel et al. [KRSR98]).

bilden. Prozessvariablen sind hingegen vor allem für die Ausführung des Orchestrierungsmodells von Bedeutung, um dort z. B. Ablaufentscheidungen bei konditionalen Verzweigungen zu treffen. Da Prozessvariablen in der Regel nur einfache Daten umfassen und Anwendungsdaten meist ungleich komplexer sind, werden zu deren Repräsentation im Rahmen des Kompositionsmodells zwei Vorgehensweisen unterschieden: Zum einen können beide Arten von Daten explizit modelliert werden, was unter Umständen zu Kapazitätsengpässen der Composition Engine führen kann. Zum anderen besteht die Möglichkeit, Anwendungsdaten lediglich als Referenzen zu berücksichtigen und deren Transfer außerhalb der Web Service Composition Middleware zu realisieren. Dann müssen aber unter Umständen noch zusätzliche Adapter für die Komponenten erstellt werden, die den Datentransfer durchführen. In jedem Fall muss aber die Möglichkeit bestehen, die im Kompositionsschema verwendeten Daten durch *Typisierung* zu kennzeichnen. Dazu hat sich in den meisten Fällen das Typsystem von XML-Schema durchgesetzt.

Für die Daten des Kompositionsschemas ist im Datenmodell zudem die Art des Zugriffs durch verschiedene Aktivitäten zu bestimmen. Hierbei sind die beiden Varianten des *Blackboards* und des *Datenflusses* zu unterscheiden. Beim Blackboard-Ansatz werden alle verwendeten Variablen explizit deklariert und können dann von allen Aktivitäten verwendet werden (unter Umständen mit differenzierten Zugriffsrechten). Dies entspricht den Variablen in klassischen Programmiersprachen. Ansätze auf Basis von Datenfluss erlauben hingegen eine differenziertere Modellierung der Verwendung von Daten. Hierbei wird durch spezifische Kanten explizit spezifiziert, welche Aktivität die Quelle und welche die Senke für Daten bilden. Dieses aus dem Workflow-Bereich übernommene Konzept erlaubt zwar eine genaue Modellierung von Datenbeziehungen, es birgt aber auch Probleme durch implizite Abhängigkeiten der Sequenz sowie die Gefahr von Race Conditions.

Ein weiterer wesentlicher Aspekt des Kompositionsmodells besteht schließlich in der Bestimmung und Bindung konkreter Provider für die WS-Komponenten. Die hierzu möglichen Mechanismen werden im Rahmen des *Aggregationsmodells* festgelegt. Grundsätzlich lassen sich hier statische und dynamische Varianten der Bindung unterscheiden. Bei statischer Bindung werden die Ports konkreter Provider bereits im Kompositionsschema fest vorgegeben und sind dementsprechend für alle Kompositionsinstanzen immer gleich. Im Falle dynamischer Bindung werden im Kompositionsschema lediglich Port Types angegeben. Deren Bindings und Ports werden erst später bestimmt. Um dies zu erreichen, kann die dynamische Bindung zum einen per Referenz bestimmt werden. Bei diesem Konzept des *Dynamic Binding by Reference* wird der konkrete Port zur Laufzeit aus einer Prozessvariable abgeleitet. Die Bestimmung des Ports bleibt dabei unbestimmt. Diese Aufgabe kann konkret im Rahmen des Deployments oder auch durch eine vorgelagerte Aktivität erledigt werden. Daneben berücksichtigen manche Aggregationsmodelle das Auffinden von Providern auch explizit. Bei derartigen Konzepten des *Dynamic Binding by Lookup* wird die

Spezifikation einer Anfrage an ein WS-Verzeichnis als Teil des Kompositionsschemas ermöglicht. Es sei erwähnt, das manche Aggregationsmodelle darüber hinaus sogar die Art der Interaktion bzw. die konkrete WSDL-Operation dynamisch zur Laufzeit ermitteln. Bei diesem Grad der Dynamik ist die Entwicklung robuster Systeme jedoch zurzeit noch nicht hinreichend erforscht. Die Variante ist daher eher theoretischer Natur.

Zur praktischen Verwendung sollte das Kompositionsmodell noch einige weitere Aspekte berücksichtigen, die im Rahmen der vorliegenden Arbeit nur am Rande erwähnt werden sollen. Dazu gehören insbesondere ein *Transaktionsmodell* und ein Modell zur *Ausnahmenbehandlung*. Ein weiterer wichtiger Aspekt ist die Berücksichtigung der zusammengesetzten Service-Qualität komponierter Web Services auf Basis eines *QoS-Modells*.

Die Business Process Execution Language (BPEL) Trotz des relativ frühen Entwicklungsstadiums des Service Oriented Computing sind aus Industrie und Forschung bereits eine Reihe verschiedener WS-Kompositionsmodelle hervorgegangen: Einige Beispiele hierfür sind im Rahmen von XLANG [Tha01], WSFL [Ley01], E-Flow [CS01], Self-Serv [BSD03], Web Components [YP02] oder DAML-S [ABH+01] zu finden. Der zurzeit gängigste Ansatz zur WS-Komposition ist jedoch die Business Process Execution Language (BPEL) [ACD+03].[138] Hierbei handelt es sich um eine Konsolidierung verschiedener früherer Ansätze aus dem industriellen Bereich. Genauer wurden hierbei die Kompositionsmodelle von XLANG und WSFL zusammengefasst und vereinheitlicht. Die Entwicklung war dabei zunächst Industrie-getrieben, wurde aber mittlerweile von OASIS übernommen [OAS05c]. Da BPEL im weiteren Verlauf der vorliegenden Arbeit noch eine wesentliche Rolle spielen wird, erfolgt nun ein kurzer Überblick über die wichtigsten Eigenschaften.

BPEL ist eine XML-Sprache zur Spezifikation von Web Service-Kompositions=schemata und Koordinationsprotokollen. Grundsätzlich repräsentiert eine BPEL-Spezifikation den Interaktionsprozess eines komponierten Web Service. Diese Spezifikation kann so detailliert erfolgen, dass der Prozess automatisch ausgeführt werden kann. In diesem Fall stellt die Spezifikation ein Kompositionsschema dar, das Konversationen mit verschiedenen Providern von WS-Komponenten und ggf. Clients des komponierten Web Service implementiert. Die Prozessspezifikation kann aber auch in nicht-deterministischer Weise erfolgen: Hierbei werden nur die möglichen Konversationen sichtbar und die dahinterstehende Logik bleibt verborgen. In diesem Fall wird dadurch das Koordinationsprotokoll des komponierten Web Service in Service-zentrierter Weise (d. h. als öffentlicher Prozess) beschrieben. Konkret beinhaltet eine BPEL-Spezifikation die folgenden Teile: Zunächst werden für die Partner, die in Konversation mit

[138] Die folgenden Betrachtungen beziehen sich auf die zur Zeit der Erstellung aktuelle Version 1.1.

dem komponierten WS stehen, *Rollen* definiert. Für jede dieser Rollen und für den komponierten Web Service werden dann die *WSDL Port Types* festgelegt. Operationen und Nachrichten der Port Types bilden die Basis des *BPEL-Prozesses*, in dem sich die einzelnen Dimensionen eines Kompositionsmodells widerspiegeln. Darüber hinaus erfolgt noch eine Beschreibung der *Korrelation* von eingehenden Nachrichten und Kompositionsinstanzen. Im Folgenden soll das durch diese Bestandteile realisierte Kompositionsmodell von BPEL dargestellt werden.

In Bezug auf das Komponentenmodell macht BPEL nur wenige Vorgaben. WS-Komponenten sind hier beliebige Software-Einheiten, die eine abstrakte WSDL-Beschreibung besitzen. Sie werden dabei immer im Kontext einer Beziehung zu einem abstrakten Konversationspartner betrachtet. Hierzu werden durch *Service Link Types*[139] Rollen für Konversationspartner und komponierte WS spezifiziert und mit je einem Port Type assoziiert. Ein *BPEL-Partner*[140] füllt eine der Rollen eines Service-Link-Types aus. Im Rahmen des BPEL-Prozesses werden Interaktionen dann in Bezug auf WSDL-Operationen von BPEL-Partnern spezifiziert. Weitere Anforderungen an WS-Komponenten können sich im Rahmen einer konkreten Anwendungssituation ergeben. Hierbei ist besonders die Unterstützung von WS-Coordination und WS-Transaction sinnvoll. Durch ein solches horizontales Koordinationsprotokoll für Transaktionen können nämlich die in BPEL enthaltenen Möglichkeiten zur Spezifikation von Transaktionseigenschaften (s. u.) genutzt werden.

Das Orchestrierungsmodell spiegelt sich in den *Aktivitäten* von BPEL-Prozessen wieder. Durch die Einflüsse der zwei unterschiedlichen Vorgängermodelle ergibt sich hier ein dualer Charakter: Es finden sich sowohl Konzepte von Prozessalgebra (π-Calculus) als auch von Aktivitätsdiagrammen wieder. Dies zeigt sich in den unterschiedlichen Arten von Aktivitäten. Zunächst werden atomare von strukturierten Aktivitäten unterschieden. Atomare Aktivitäten repräsentieren entweder eine Interaktion (d. h. asynchrones Senden einer Nachricht, asynchrones Empfangen einer Nachricht oder synchroner Aufruf einer Operation) oder prozessinterne Funktionen, z. B. zur Datenmanipulation (s. u.). Strukturierte Aktivitäten sind hingegen durch die Operatoren des π-Calculus inspiriert: Sie bestimmen die Ausführungsreihenfolge der darin enthaltenen Aktivitäten und lassen sich dabei beliebig verschachteln. Das Ergebnis ist eine so genannte *Blockstruktur*. BPEL beinhaltet dabei die strukturierten Aktivitäten *Sequence* (sequenzielle Ausführung), *Switch* (Konditionale Ausführung), *While* (Wiederholte Ausführung), *Pick* (Ausführung nach Ereignis) und *Flow* (Nebenläufige Ausführung). Im Rahmen von Flow-Aktivitäten kann durch Definition von *Links* eine Graph-basierte Struktur wie bei Aktivitätsdiagrammen erzeugt werden. Ein Link verbindet dabei immer genau zwei Aktivitäten und kann mit einer Bedingung verknüpft werden. Aktivitäten können mehrere aus- oder eingehende Links besitzen, die mit weiteren Bedingungen zur Zusammenführung *(Join Conditions)* verknüpft

[139] Service Link Types werden in neueren Versionen als Partner Link Types bezeichnet.
[140] BPEL-Partner werden in neueren Versionen als Partner-Links bezeichnet.

werden können. Zyklische Strukturen sind dabei nicht erlaubt. Insgesamt besteht in vielen Fällen die Möglichkeit, die Orchestrierung auf zwei verschiedene Arten (Block- oder Graph-Struktur) zu spezifizieren.

Im BPEL-Datenmodell werden alle Daten – sowohl Anwendungsdaten als auch Prozessvariablen – explizit modelliert. Dazu werden in einem BPEL-Prozess Variablen deklariert, die entweder als WSDL Messages, XML-Schema-Basistypen oder XML-Schema-Elemente typisiert werden können. Variablen können innerhalb des Prozesses durch *Assign-* oder *Copy-*Aktivitäten manipuliert werden. Dabei ist es möglich, durch eine XML Query Sprache spezifische Teile der Daten zu selektieren. Im Prozess werden die Variablen zum einen in den Bedingungen der Ablaufsteuerung verwendet. Zum anderen werden damit die zu versendenden Nachrichten konstruiert bzw. Daten aus den empfangenen Nachrichten übernommen. Innerhalb des Prozesses greifen die Aktivitäten über ein Blackboard auf die Variablen zu. Hierbei besteht die Möglichkeit zur Synchronisation von Zugriffen.

Das Aggregationsmodell von BPEL erlaubt eine statische Bindung sowie auch dynamische Bindung per Referenz. Bei der Bindung werden für die Port Types der BPEL-Partner konkrete WSDL-Beschreibungen ergänzt, die als *Endpoint Reference* bezeichnet werden. Dies kann zum einen im Rahmen des Deployments auf statische Weise erfolgen. Die genaue Funktion hängt dabei von der Implementierung der Composition Engine ab. Zum anderen ist auch eine dynamische Zuweisung während der Ausführung von BPEL-Prozessen möglich. Dies geschieht durch Zuweisung einer Endpoint Reference zu einem BPEL-Partner im Rahmen einer Assign-Aktivität. Die Endpoint Reference muss dazu in Form einer Variable vorliegen. Sie kann dieser entweder statisch zugewiesen werden oder aus einer eingehenden Nachricht hervorgehen. Letztere kann z. B. die Antwort auf eine Anfrage bei einem Web Service-Verzeichnis sein.

Des Weiteren definiert BPEL eine kombinierte Methode zur Behandlung von Ausnahmefällen *(Exceptions)*, Ereignissen *(Events)* und der Kompensation von Transaktionen *(Compensation)*. Grundlage ist jeweils eine Menge von Aktivitäten, die einen zusammenhängenden Bereich *(Scope)* oder eine Transaktion *(Transactional Region)* bilden. Dafür wird jeweils eine Aktivität definiert, die im Falle von Exceptions/Events/Compensations ausgeführt werden soll *(Handler)*. Wie beim Exception Handling von Programmiersprachen können auch hier die auslösenden Faktoren auf die nächsthöhere Schachtelungsebene weitergeleitet werden.

Schließlich ist noch eine Besonderheit von BPEL zu erwähnen: die Unterstützung des Conversation Routings. Wie schon weiter oben beschrieben, kann diese Aufgabe prinzipiell auch durch eine separate Konversationskontrolle der Web Service Coordination Middleware übernommen werden. So ein Mechanismus ist aber nicht immer verfügbar. In diesem Fall bietet BPEL die Möglichkeit, die Korrelation von Nachrichten und Kompositionsinstanzen anhand von Nachrichteninhalten festzulegen. Hierzu werden so genannte *Correlation Sets* definiert, die Teile einer Nachricht als Indikator für die Korrelation ausweisen. Correlation Sets können dann verschiedenen

Interaktionen zugewiesen werden, z. B. kann beim Senden und Empfangen von Nachrichten eines asynchronen Aufrufs eine eindeutige Kennung (z. B. eine Kunden- oder Auftragsnummer) für die Korrelation bestimmt werden. Anhand dieser Kennung wird dann die Antwortnachricht an diejenige Kompositionsinstanz weitergeleitet, von der auch die Anfrage stammte.

3.4.3. SOC zur (Geschäfts-)Prozessintegration

In den letzten Sektionen wurde das SOC-Paradigma eingeführt und anhand praktischer Web Service-Techniken konkretisiert. Auf dieser Basis kann nun ein spezifischer Aspekt des SOC beleuchtet werden, der Service-orient. Techniken in den Fokus der vorliegenden Arbeit gerückt hat: Service-orient. Ansätze zur Prozessintegration.

Zunächst soll der konzeptionelle Hintergrund der Prozessintegration im VDU noch einmal kurz zusammengefasst werden. Zudem soll noch einmal eine Rekapitulation der bisherigen Ansätze für Systemtechniken zur Unterstützung der Prozessintegration erfolgen. Hieran lassen sich dann die *Potenziale* von SOC und Web Services aufzeigen.

Weder SOC-Konzepte noch Web Service-Techniken bieten jedoch eine pauschale Lösung aller Integrationsfragen. Systemintegration ist stets ein individuelles Problem mit vielen möglichen Lösungsansätzen. Dementsprechend sollen verschiedene Forschungsansätze zur Service-orient. Prozessintegration vorgestellt werden. Diese Ansätze geben wichtige Hinweise auf den Lösungsweg in Bezug auf die Fragestellung der vorliegenden Arbeit. Die Betrachtung gliedert sich hier in zwei wesentliche Aspekte der Prozessintegration im SOC. Zunächst werden Ansätze zur *Regelung zwischenbetrieblicher Interaktionsprozesse* durch Service-Koordination dargestellt. Dann schwenkt die Betrachtung zur entsprechenden *Implementierung von Integrationsprozessen* durch Service-Komposition.

3.4.3.1. SOC-Potenziale zur Prozessintegration

Der Begriff der Prozessintegration zieht sich als roter Faden durch das gesamte Kapitel. Zunächst wurde er im Zusammenhang der integrierten Informationsverarbeitung als Perspektive der betrieblichen Systemintegration eingeführt:[141] Die Perspektive der Funktions- und Prozessintegration betrachtet Integration auf Basis der Anwendungsfunktionen einzelner Systeme und fokussiert die entstehenden Prozesse auf Anwendungsebene. Eng damit verbunden sind die Perspektiven der *Interoperabilität* und *Koordination*. Interoperabilität befähigt die Systemkomponenten zur Interaktion und schafft einen Zugang zu deren Anwendungsfunktionen. Koordination dient der Regelung von Beziehungen zwischen Anwendungsfunktionen und als Basis zur entsprechenden Steuerung von Interaktionen. Die Beziehungen der Anwendungsfunktionen geben dabei u. a. den (geschäftlichen) Anwendungsprozess wieder. In diesem Zusammenhang wurde herausgestellt, dass die Koordination in der Regel nicht Teil

[141] Vgl. Sektion 3.2.1.2, S. 93.

der ursprünglichen Systemkomponenten ist. Sie muss explizit als übergeordnete Koordinationsebene etabliert werden. Auf dieser Ebene gibt ein *Koordinationsprozess* (als Ergebnis der anwendungsbezogenen Prozessintegration) die Regeln für einen *Kompositionsprozess*[142] vor. Dieser Kompositionsprozess implementiert dabei die Steuerung des letztendlichen *Interaktionsprozesses*. Eine Infrastruktur der Systemintegration muss in diesem Sinne gleichzeitig die Interaktion und Kooperation von Systemkomponenten unterstützen. Je nach Integrationsreichweite (inner-/zwischenbetrieblich) sind dabei verschiedenartige Anforderungen zu berücksichtigen.

Auch in Bezug auf die Anforderungen an eine IV-Infrastruktur für virt. Dienstleistungsunternehmen tritt die Prozessintegration ganz explizit hervor.[143] In diesem Fall müssen Dienstleistungsprozesse in besonders flexibler Weise zwischen verschiedenen und häufig wechselnden Organisationen in Einklang gebracht werden. Dies betrifft zum einen die Kollaboration von Netzwerkunternehmen eines stets temporär ausgelegten virt. Unternehmens. Zum anderen betrifft das die Erbringung von Dienstleistungen für verschiedene Kunden im Rahmen eines interaktiven Kernleistungsprozesses (mehr dazu im nächsten Kapitel). In diesem Sinne sind Aspekte der *Kooperationsunterstützung* durch *Koordination* interoperabler AS zur *Kollaboration* von Netzwerkunternehmen und Kunden integrale Bestandteile der horizontalen Basisinfrastruktur von VU. Dies gilt noch mehr in Hinblick auf den Fokus der vorliegenden Arbeit: E-Services können als explizite Koordinationsebene im spezifischen Anwendungskontext der Dienstleistungserbringung im VDU aufgefasst werden. Sie fokussieren dabei die Integration des Dienstleistungsprozesses.

Unterstützung der Prozessintegration durch Middleware Die technische Infrastruktur zur System- und insbesondere Prozessintegration basiert in den meisten Fällen auf Middleware. Das Middleware-Konzept beruht dabei auf dem Prinzip einer systemübergreifenden Ebene, auf der einheitliche Abstraktionen und Mechanismen zur Interaktion verschiedenartiger (Software-)Komponenten bereitgestellt werden.[144] Durch diese Vorgaben wird zunächst die Interoperabilität ansonsten potenziell heterogener Komponenten hergestellt. Aufbauend auf deren grundlegender Interaktionsfähigkeit, wurden im Kontext von Middleware verschiedene erweiterte Unterstützungsmechanismen entwickelt, die für die Prozessintegration im VDU von Interesse sind. Dazu gehören vor allem Entwicklungen im Kontext von Integrationsplattformen. A2Ai-Plattformen unterstützen die lose Kopplung kompletter Legacy-Anwendungssysteme auf Basis von Message Brokern, die eine nachrichtenbasierte, asynchrone und vor allem indirekte Interaktion von Komponenten (bzw. Adaptern/Wrappern) realisieren. Komponenten interagieren hierbei nicht untereinander,

[142] Der Kompositionsprozess wird zum Teil auch als *Integrationsprozess* oder *Orchestrierungsprozess* bezeichnet [CMA03]. Hier soll im Folgenden einheitlich der Begriff des Kompositionsprozesses verwendet werden, der zugleich generell ist und der SOC Terminologie entspricht.

[143] Vgl. Sektion 3.2.2.1.

[144] Vgl. Sektion 3.3.2.

sondern jeweils mit dem Message Broker, der die Nachrichten entsprechend einer individuellen Logik weiterleitet. Hierdurch realisieren Message Broker eine übergeordnete Koordinationsebene. In einigen Fällen basiert die Logik der Koordination direkt auf einem Prozessmodell. Die Koordination wird dabei in der Regel an ein Workflow-Management-System delegiert. Auf Basis solcher Plattformen kann ein Anwendungsprozess sehr direkt als Workflow-Prozess implementiert werden, der den Interaktionsprozess beteiligter AS steuert.

A2Ai-Plattformen sind in Bezug auf die Integrationsreichweite auf den innerbetrieblichen Bereich ausgelegt. Wenn bei der Systemintegration die Grenze zwischen organisatorischen Domänen überschritten wird, entstehen hingegen grundsätzliche Probleme.[145] Dies liegt nicht so sehr daran, dass in diesem Fall die Vernetzung in der Regel über WAN erfolgt. Vielmehr bietet die verteilte Organisationsstruktur keinen offensichtlichen Ort für die überwiegend zentralisierten Middleware-Mechanismen. Dass hierfür nicht einfach eines der beteiligten Partnerunternehmen gewählt werden kann, liegt an dem meist fehlenden Vertrauenskontext. Unter diesen Umständen will in der Regel keine Organisation einer anderen die Kontrolle über und Einsicht in kritischer Mechanismen gewähren. In der Folge können nicht einmal die grundlegenden Interoperabilitätsmechanismen der Middleware vorausgesetzt werden. Das Fehlen eines gemeinsamen Kontextes führt darüber hinaus auch zu Problemen auf höheren Abstraktionsebenen. So ist ein einheitliches Verständnis in Bezug auf Struktur und Semantik von Daten oder Geschäftsprozessen in vielen Fällen nicht a priori vorhanden. Als Konsequenz sind auf den Ebenen der Kommunikation, Information und Geschäftsprozesse organisationsübergreifende Standards notwendig. Für deren Umsetzung sind B2Bi-Plattformen verantwortlich. Diese setzen auf konventioneller Middleware auf und vermitteln zwischen internen und externen Interaktionen. Für die Prozessintegration leiten sich daraus wesentliche Konsequenzen ab:

1. Der Koordinationsprozess muss zwischen den beteiligten Organisationen explizit geregelt werden.

2. Es existieren getrennte Kompositionsprozesse in den beteiligten Organisationen, die gemeinsam den Koordinationsprozess implementieren.

Die Perspektive der Prozessintegration wird im Rahmen von Integrationsplattformen überwiegend durch Workflow-Techniken[146] eröffnet. Diese sind explizit darauf ausgelegt die Koordination von Akteuren in Bezug auf den Ablauf ihrer Aktivitäten zu realisieren. Bei der Systemintegration ist das die Koordination von Systemkomponenten in Bezug auf den Ablauf ihrer Anwendungsfunktionen. Ein Workflow repräsentiert also den Kompositionsprozess von Systemkomponenten. Auf dieser Basis steuert die Engine eines WfMS den Interaktionsprozess zwischen ihnen. Dies kann allerdings

[145] Vgl. Sektion 3.3.2.4.
[146] Vgl. Sektion 3.3.3.

nur innerhalb genau einer Workflow-Domäne geschehen. Im zwischenbetrieblichen Kontext muss ein WfMS zum Inter-Enterprise Workflow-System erweitert werden. Dort kommen nämlich die oben beschriebenen Konsequenzen zum Tragen: Beteiligte Partner besitzen getrennte WfMS mit eigenständigen Workflows zur Integration von internen Komponenten. Diese sind in der Regel privat und werden den Partnern nicht offengelegt. Daneben existiert ein *öffentlicher (kollaborativer) Geschäftsprozess* mit den Partnern. Aus der Sicht einzelner WfMS sind diese Partner lediglich Komponenten, die mit einem Kompositionsprozess gesteuert werden. Um die Interoperabilität dieser Workflows herzustellen, muss eine Regelung des gemeinsamen Koordinationsprozesses und dessen Umsetzung auf einzelne Kompositionsprozesse bei den Partnern erfolgen. Der Koordinationsprozess wird dabei als *Protokoll* für alle Partner vorgegeben. Aus diesem Protokoll wird der Kompositionsprozess als Workflow abgeleitet, partitioniert und auf die WfMS der Partner aufgeteilt. Je nach Ansatz können dabei Schemata oder Instanzen partitioniert werden. Beim verbreiteten Modell der losen Kopplung erfolgt eine Partitionierung und Verteilung des Schemas. Jeder Partner führt dann einen individuellen Teil des gesamten Kompositionsprozesses aus, wobei sich insgesamt der im Protokoll vorgegebene Koordinationsprozess ergibt. Je nach Ansatz müssen die einzelnen Workflows (bzw. WfMS) dann auf verschiedene Weise gekoppelt werden. Das WfMC-RM sieht z. B. ein spezifisches API für WfMS (WAPI 4) vor. Abbildung 3.30 fasst den Einsatz integrativer Middleware zur organisationsinternen und -übergreifenden Prozessintegration zusammen und zeigt deren Unterschiede auf.

In der Praxis ist die Verbreitung Workflow-basierter Unterstützungsmechanismen zur Prozessintegration sowohl im A2A- als auch im B2B-Bereich eher hinter den Erwartungen zurückgeblieben. Als Mittel der Systemintegration zur prozessbasierten Komposition von AS-Komponenten fehlen hier vor allem einheitliche Modelle. In der Regel wird zur Maximierung der Generalität kein einheitliches Komponentenmodell festgelegt, d. h. dass für jedes neue AS ein Adapter realisiert werden muss. Zudem sind die meisten Workflow-Modelle unterschiedlich. Dies führt zu einer steilen Lernkurve bei der Schulung von Integrationstechniken. Zugleich tendiert die Forschung und Entwicklung in diesem Gebiet zur Zergliederung. Schließlich wird aus praktischer Sicht die Portierbarkeit von Kompositionsprozessen erschwert. Bei der zwischenbetrieblichen Integration kommen weitere Probleme hinzu. Durch die uneinheitlichen Modelle ist Workflow-Interoperabilität hier nur sehr schwer zu realisieren. Einheitliche Standards wie das WfMC-RM mit Workflow-Modellen wie XPDL und WfMS-Protokollen wie Wf-XML sind kaum umgesetzt worden. Ohne diese ist aber weder die Regelung von B2B-Protokollen noch deren Implementierung als IEWS möglich.

Prozessintegration mit zwischenbetrieblicher WS-Middleware Service Oriented Computing generell und Web Services konkret geben der Systemintegration neue Impulse. Insbesondere wird hierdurch die Realisierung der zwischenbetrieblichen Systemintegration gefördert. Dieser Umstand geht auf drei Faktoren zurück: die Service-

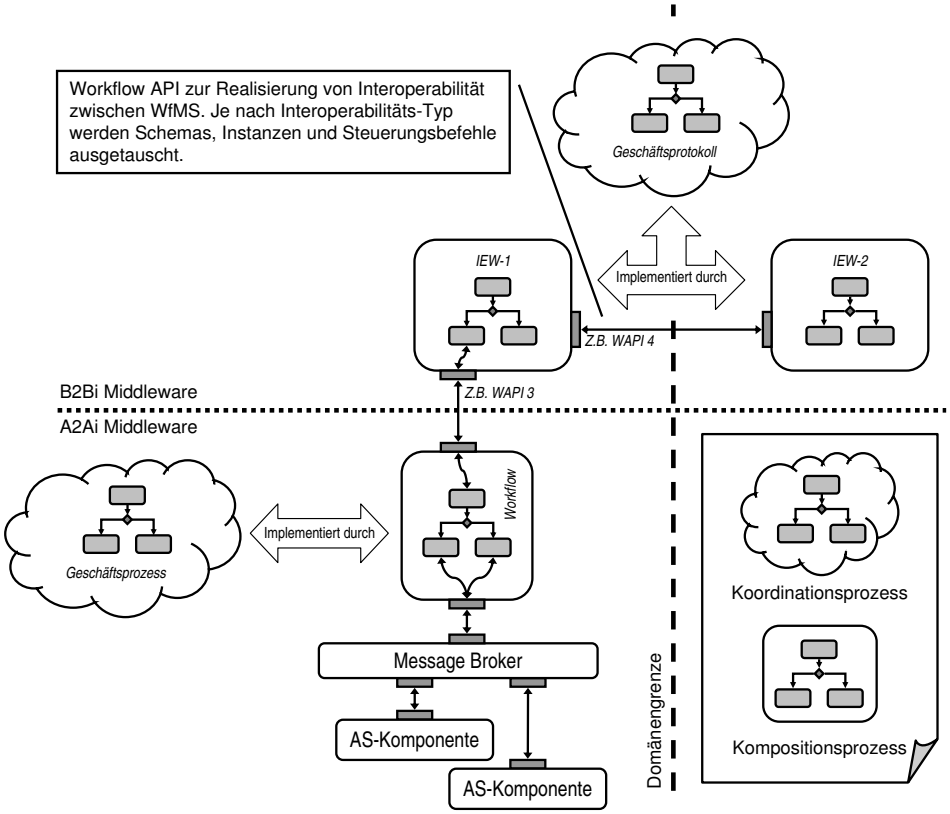

Abbildung 3.30.: Prozessintegration in und zwischen Organisationen am Beispiel von Workflow-basierten Mechanismen integrativer Middleware

Abstraktion, organisationsübergreifende Middleware-Protokolle und erfolgreiche Standardisierung. Web Services bieten eine Abstraktion zur Repräsentation von Anwendungsfunktionen zwischen Organisationen und bilden damit die Basis für einheitliche Komponentenmodelle. Die Mechanismen zur systemtechnischen Unterstützung dieser und weiterer Abstraktionen sind dabei explizit für den organisationsübergreifenden Einsatz konzipiert. Dazu sind mittlerweile für viele grundlegende Mechanismen einer klassischen Middleware XML-basierte Protokolle entworfen worden. Diese werden stets den Anforderungen des organisationsübergreifenden Kontextes gerecht. Wenn nötig sind dabei für klassischerweise zentral realisierte Mechanismen dezentralisierte Varianten entwickelt worden. Hierbei arbeitet die Middleware verschiedener Organisationen als Peer-to-Peer-System zusammen. Praktisch gesehen kann dadurch aber nur dann eine zwischenbetriebliche Integration realisiert werden, wenn die SOC-Techniken von den entsprechenden Partnerunternehmen auch in einheitlicher Weise umgesetzt werden. Dies wird durch zwei Aspekte begünstigt: Zum einen wurde die Standardisierung frühzeitig von maßgeblichen Institutionen des Internets (u. a. W3C und OASIS) übernommen. Zum anderen wurde diese Standardisierung in konzertierter Weise von

den führenden Software-Herstellern (u. a. IBM und Microsoft) getrieben und rasch in Produkte umgesetzt. Alles in allem bilden SOC-Techniken mit standardisierten Abstraktionen und Mechanismen also eine Middleware für den zwischenbetrieblichen Einsatz. Dies heißt jedoch nicht, dass die Techniken nicht auch bei der A2A-Integration einsetzbar wären. In der momentanen betrieblichen Praxis liegt hier sogar ihr Anwendungsschwerpunkt.

Der Prozessbegriff wird im Service Oriented Computing erst durch das erweiterte Service-orient. Modell thematisiert. Dort wechselt die Perspektive von einem auf mehrere Services. Der Fokus verschiebt sich dabei von der einzelnen Service-Interaktion zum Interaktionsablauf. In diesem Zusammenhang ist die *Konversation* und *Choreografie* von Interesse: also der Ablauf einer multilateralen Interaktion verschiedener Services (a) aus der eingeschränkten Service-zentrierten Sicht eines einzelnen Teilnehmers und (b) aus der ganzheitlichen neutralen Sicht eines externen Beobachters. Konversationen und Choreografien sind deshalb interessant, weil alle beteiligten Services darin übereinstimmen müssen, um sinnvoll zu interagieren. Eine entsprechende *Service-Koordination* basiert auf der Festlegung erlaubter Interaktionsabläufe als Koordinationsprotokoll. SOC-Mechanismen erlauben es den Services, die zusammenhängenden Interaktionen einer Konversation zu erkennen und mit dem Koordinationsprotokoll abzugleichen. Auf der anderen Seite muss sich jeder Service auch in seinem eigenen Verhalten an das Koordinationsprotokoll halten. Besonders bei multilateralen Interaktionsabläufen kann dies eine komplexe Logik erfordern. Durch *Service-Komposition* kann diese Logik als Prozess ausgedrückt und ausgeführt werden. Dieses *Kompositionsschema* implementiert in Analogie zu Workflows die Koordination von Service-Komponenten im Sinne der Abfolge ihrer Service-Operationen.[147] Genauer gesagt, lassen sich durch ein Kompositionsschema die Konversationen einzelner individueller Services einer übergeordneten Choreografie implementieren. Abbildung 3.31 zeigt die prozessorientierten Mechanismen der erweiterten WS-Architektur im Zusammenhang.[148]

Die Mechanismen der Service-Koordination und -Komposition erlauben eine prozessorientierte Integration sowohl im A2A- als auch B2B-Kontext. In Bezug auf A2Ai können Geschäftsprozesse direkt durch Service-Kompositionsschemata implementiert werden. In Bezug auf B2Bi erfolgt eine Regelung der organisationsübergreifenden Geschäftsprozesse durch Service-Koordination und deren Implementierung durch

[147] Solche Interaktionsabläufe, bei denen ein Service mit mehreren anderen Konversationen führt, decken auch den Fall ab, dass einige der Konversationspartner Leistungen erbringen, die zusammen an einem anderen Konversationspartner weitergegeben werden. In diesem Sinne komponiert der Service die Leistungen anderer Services und erbringt auf dieser Basis selbst einen komponierten Service. Ob es sich bei multilateraler Konversation tatsächlich um eine Komposition von Leistungen handelt, und wer in so einer Konstellation die Rolle des Clients oder Providers inne hat, kann in heutigen Kompositionsmodellen allerdings selten explizit ausgedrückt werden.

[148] Prozessorientierte Erweiterungen der WS-Architektur (und deren Nutzung zur flexiblen Komposition) wurden vom Autor dieser Arbeit in [ZLB04] beschrieben.

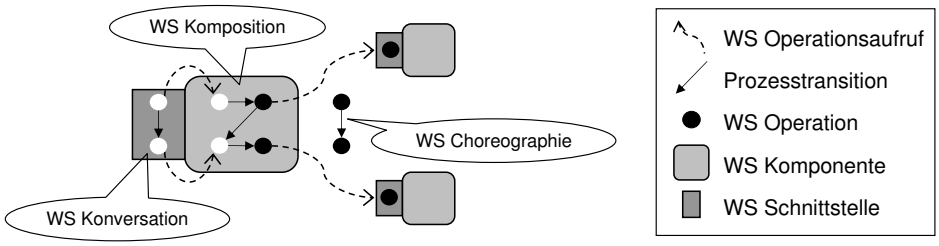

Abbildung 3.31.: Prozessorientierte Erweiterungen der WS-Architektur [ZLB04]

partitionierte Service-Komposition. Damit gehen die Konzepte des SOC also zunächst grundsätzlich konform zu denjenigen Workflow-basierter integrativer Middleware. Der Unterschied liegt in den oben schon genannten Faktoren: die WS-Architektur spezifiziert einfache und einheitliche Standards sowohl für ein Komponentenmodell als auch für organisationsübergreifende Middleware-Protokolle. Dies hat entscheidende Vorteile für A2A- und noch mehr für B2B-Integration.

Im A2A-Kontext ist im Wesentlichen das einheitliche Komponentenmodell von Vorteil. Hier liegen viele AS-Komponenten schon in dieser Form vor. Aber auch Altsysteme profitieren, denn das einheitliche Komponentenmodell fördert die Entstehung effizienterer Adapter-Techniken. Des Weiteren können auf Basis von WS vielfältige Kompositionsmodelle realisiert werden. Zurzeit konzentriert sich die Forschung und Entwicklung auf eine relativ kleine Zahl, wobei der Fokus klar auf BPEL liegt. Diese Sprache setzt den Standard in Bezug auf Abstraktionen und Mechanismen der Komposition. Dazu gehört u. a. die Berücksichtigung von Ereignissen, Ausnahmen, Kompensation und Korrelation. Damit lassen sich Geschäftsprozesse oft besser implementieren als mit Workflow-Modellen, die zum Teil nicht originär auf die Koordination von Software-Komponenten ausgelegt waren. Durch Vereinheitlichung grundlegender Kompositionskonzepte und Verbreitung einzelner Kompositionsmodelle wird die Entwicklung und Portierung von Kompositionsschemata auf unterschiedlichen WS-Middleware-Plattformen erheblich vereinfacht.

Noch größere Vorteile ergeben sich im B2B-Bereich: WS liefern hier ein organisationsübergreifendes Komponentenmodell. Dieses garantiert die Interoperabilität bei der Durchführung einzelner Interaktionen zwischen Kollaborationspartnern. Es vereinheitlicht auch die Regelung und Umsetzung von Kollaborationsprozessen. Konkret wird das gleiche Komponentenmodell sowohl zur Regelung geschäftlicher Kollaborationen durch Koordinationsprozesse (bzw. Protokolle) als auch zur Implementierung dieser Protokolle durch Kompositionsprozesse verwendet. Dadurch stehen die beiden Ebenen in enger Beziehung zueinander. Dies erleichtert die wechselseitige Überführung sowohl in manueller als zum Teil auch in automatisierter Form. Workflow-basierten B2Bi-Plattformen liegt hingegen meist nicht das gleiche Komponentenmodell zugrunde wie den B2Bi-Protokollen, z. B. von EDIFACT, ebXML oder RosettaNet. In diesen Fällen ist eine Implementierung der Protokolle schwieriger.

Ein weiterer, äußerst positiver Aspekt ergibt sich durch die P2P-Middleware-Protokolle der WS-Architektur. Auf Basis der WS-Protokollinfrastruktur lässt sich Interoperabilität der prozessorientierten Laufzeitumgebungen verschiedener Organisationen sehr viel leichter erreichen als bei Workflow-basierten B2Bi-Plattformen. Der Hintergrund wird wiederum durch Web Services gebildet, die bei B2B-Kollaborationen den öffentlichen Geschäftsprozess eines jeden Kollaborationspartners abstrahieren und dessen Implementierung durch einen jeweils spezifischen Kompositionsprozess nach außen kapseln. Aus globaler Sicht wird dadurch eine horizontale Partitionierung des DBP mit *loser Kopplung* der einzelnen Fragmente vorgegeben.[149] Durch diese Vorgabe reicht ein relativ einfaches horizontales Koordinationsprotokoll zur Abstimmung der Kollaborationspartner in Bezug auf das (vertikale) Geschäftsprotokoll und die Vergabe der darin spezifizierten Rollen. Ein entsprechendes horizontales Protokoll kann z. B. auf der Protokollinfrastruktur von WS-Coordination basieren.

Die Ausführungen zur Nutzung von prozessorientierten SOC-Techniken im B2Bi zeigten die Vorteile der Service-Koordination und Komposition gegenüber entsprechenden Workflow-Techniken auf. Aus diesen Vorteilen ergeben sich Potenziale für die Prozessintegration u. a. in Bezug auf zwei Aspekte: *Regelung kollaborativer Beziehungen* und *Implementierung kollaborativer Regelungen*.

Kollaborative Beziehungen liegen direkt im Fokus der Prozessintegration. Hier drückt sich aus, wie verschiedene Organisationen ihre lokalen Prozesse im Rahmen einer Kollaboration miteinander verbinden. Es ist unabdingbar, dass hierüber Einigkeit besteht. Wenn das Verhalten der Partner – das sich in ihren wechselseitigen Interaktionen ausdrückt – hierin voneinander abweicht, wird die Kollaboration scheitern. Spätestens wenn eine erwartete Nachricht ausbleibt oder eine unerwartete Nachricht nicht gedeutet werden kann, bricht der Interaktionsfluss ab. Aus diesem Grund muss die kollaborative Beziehung zwischen potenziellen Partnern explizit geregelt werden. Die Regelung erfolgt dabei durch Spezifikation eines gemeinsamen Koordinationsprozesses. In Service-orient. Ansätzen werden die Interaktionen eines Koordinationsprozesses durch WS-Operationen ausgedrückt. Auf diese Weise wird die Ausdrucksmächtigkeit erheblich ausgeweitet. Koordinationsprotokolle gliedern sich dann nämlich in eine Familie vielfältiger WS-Beschreibungssprachen auf unterschiedlichen Abstraktionsebenen ein und können von deren Inhalten profitieren. Dies gilt z. B. für die technische Realisierung der Interaktion (WSDL) oder deren Semantik (DAML-S). Diese Aspekte können also bei der Regelung einer Kollaboration mitberücksichtigt werden. So eine Regelung kann z. B. im Zuge des Neuentwurfs einer Kollaborationsform frei erfolgen. Es kann aber auch sein, dass vorhandene Geschäftsprozesse bei der Regelung berücksichtigt werden müssen und die Regelung also davon abhängig ist. Vom Vorgang her besteht die Regelung einer Kollaboration oft im manuellen Entwurf eines Koordinationsprotokolls auf Basis ausdrucksstarker Modelle und Sprachen. Besonders bei der freien Regelung ist dies der Normalfall.

[149] Vgl. Formen der Workflow-Interoperation in Tab. 3.4, Sektion 3.3.3.2, S. 146.

Aspekte	Varianten	exemplarische Ansätze
Regelung von Kollaborationen durch Koordinationsprotokolle	freie Regelung	*WSCL, WSCI, BPEL, BPSS, DySCo, SELF-SERV*
	abhängige Regelung	*Reconcilation*
Implementierung von Koordinationsregeln durch Kompositionsschemas	manuelle Implementierung	*BPEL, WSFL, XLANG, eFlow*
	automatische Implementierung	*Templates, DySCo, SELF-SERV*

Tabelle 3.5.: Unterstützung der Prozessintegration durch SOC-Techniken

In anderen Fällen kann der Regelungsvorgang auch zum Teil automatisiert werden. Hierbei kann wiederum das Netz verschiedener Service-Beschreibungen hilfreich sein. So ist es z. B. möglich, die Koordinationsprotokolle einzelner Services voneinander abzuleiten und aneinander anzupassen. Solche Mechanismen liefern eine Unterstützung für die abhängige Regelung. Tabelle 3.5 fasst diese Varianten im oberen Teil zusammen und nennt einige entsprechende Ansätze.

Die Perspektive der integrierten Anwendungsprozesse ist für die Gestaltung von Kollaborationen erstrangig. Liegt aber eine Regelung der Kollaboration in Form eines Koordinationsprozesses vor, muss dieser schließlich auch implementiert werden. Der klassische Weg ist Programmierung nach Vorgabe der koordinativen Regelungen. WS-Kompositionssprachen erlauben hingegen die Entwicklung einer Integrationslogik für die Web Services von Kollaborationspartnern unter Beibehaltung der Prozess-Perspektive. Hierbei werden meist spezifische Abstraktionen bereitgestellt. Mit diesen wird zum einen die Entwicklung voll automatisierter Prozesse unterstützt (z. B. Events, Exceptions). Zum anderen werden organisationsübergreifende Kollaborationen berücksichtigt (z. B. dynamische Komposition oder nicht-isolierte Transaktionen). Diese Mittel bieten Entwicklern vielfältige Möglichkeiten zur Implementierung von Kollaborationen. Neben der manuellen Entwicklung kann aber auch die schon angesprochene enge Verknüpfung von WS-Beschreibungen als Ansatzpunkt zur Automatisierung von Teilen des Implementierungsvorgangs herangezogen werden. Bei dieser Variante nutzen einige Ansätze die enge Beziehung von Koordinationsprotokoll und Kompositionsschema aus. Aus den kollaborativen Regelungen werden dann entweder Teile oder sogar die kompletten Kompositionsschematas der Kollaborationspartner generiert. Die beiden Varianten der Implementierung werden im unteren Teil von Tabelle 3.5 mit einigen entsprechenden Ansätzen in Verbindung gebracht.

3.4.3.2. Ansätze SOC-basierter Prozessintegration

SOC-basierte Ansätze für die Regelung und Implementierung von Kollaborationen wurden mittlerweile in verschiedenen Arbeiten thematisiert. In den folgenden Sektionen sollen einige davon kurz vorgestellt werden. Die Strukturierung der Darstellung erfolgt dabei nach diesen beiden Aspekten der Prozessintegration zunächst für die *Regelung* und dann für die *Implementierung*.

Ansätze zur kollaborativen Regelung Services bieten eine passende Abstraktion für (geschäftliche) Anwendungsfunktionen im organisationsübergreifenden Kontext. Dadurch bieten sie eine gute Basis zur Regelung von Kollaborationen. Grundsätzlich kann hier zwischen einer *freien Regelung* und einer *abhängigen Regelung* unterschieden werden. Für beide Fälle existieren entsprechende Ansätze zur Service-Koordination. Dazu gehören sowohl Ansätze für den manuellen Entwurf als zum Teil auch für die automatisierte Herleitung von Koordinationsprotokollen.

Eine freie Regelung von Kollaborationsprozessen soll hier bedeuten, dass vertikale Koordinationsprotokolle ohne Berücksichtigung spezifische Geschäftsprozesse einzelner Partner von Grund auf neu entwickelt werden. Dabei sind die Regelungen im Allgemeinen auf (geschäftlicher Anwendungsebene) z. B. in juristischer Form schon vorhanden. Diese sind dann auf die Ebene Service-orient. Systeme zu übertragen. Dort dienen sie dann potenziellen Kollaborationspartnern als Vorgabe für die Implementierung ihrer Services. Grundsätzlich können koordinative Regelungen als *Konversationen* oder *Choreografien* ausgeführt werden. Konversationen werden dabei meist zur Regelung bilateralen Kollaborationen verwendet. Zur Regelung multilateraler Kollaborationen sind hingegen die Konversationen aller Teilnehmer zu spezifizieren. In diesem Fall ist es günstiger, den Gesamtprozess der Kollaboration zu betrachten. Dieser wird durch Koordinationsprotokolle in Form von Choreografien repräsentiert.

In Bezug auf Konversationen existieren verschiedene Sprachen, die eine freie Spezifikation von Regelungen erlauben: **WSCL** (Web Service Conversation Language) war einer der ersten Ansätze zur Spezifikation derartiger Koordinationsprotokolle [BBB+02]. Hierbei werden die möglichen Konversationen eines Web Services in Form von Zustandstabellen beschrieben, wobei Zustandsübergänge mit Web Service-Operationen markiert sind. Die Konversationen werden in einer XML-Sprache formuliert, die sich direkt auf eine WSDL-Beschreibung bezieht. Auch auf Basis von **BPEL** [ACD+03] ist die Spezifikation erlaubter Konversationen eines Service möglich. In diesem Fall spricht man von so genannten *abstrakten Prozessen*. Diese entsprechen den öffentlichen Geschäftsprozessen von Kollaborationspartnern. Der wesentliche Unterschied zu Kompositionsschemata besteht dabei im Nicht-Determinismus. Dieser gründet sich vor allem auf die beschränkte Spezifikation von Prozessvariablen. Einen ähnlichen Ansatz verfolgt auch **WSCI** (Web Service Choreography Interface) [Ark02]. Hierbei werden einer WSDL-Beschreibung WSCI-Interfaces hinzugefügt. Das hierin beschriebene Protokoll basiert auf Aktivitätsdiagrammen und nimmt eine rollenspezifische Perspektive ein. Jede Aktivität repräsentiert also eine WS-Interaktion zwischen der Rolle des Service und einer anderen Rolle. In diesem Rahmen bietet WSCI verschiedene Abstraktionen u. a. zur Repräsentation von Exceptions, Transaktionen und Korrelationsinformationen an.

In verschiedenen Ansätzen wird zur Regelung der Kollaboration eine übergeordnete Perspektive eingenommen. Dabei wird der *Distributed Business Process* fokussiert,

der die öffentlichen Geschäftsprozesse aller Kollaborationspartner miteinander in Beziehung setzt. In diesem Sinne kann die Perspektive der Prozessintegration bei der Regelung besser berücksichtigt werden. Das Ergebnis sind Koordinationsprotokolle in Form von WS-Choreografien. Diese Protokolle regeln alle Interaktionsvorgänge zwischen Web Services verschiedener Rollen einer Kollaboration. Eine Implementierung des Protokolls – z. B. durch Kompositionsschemata – muss diesen Regeln entsprechen.

Die Definition von Kollaborationen als Distributed Business Process wurden schon 2001 in ebXML thematisiert: Hier bietet **BPSS** (Business Process Specification Schema) die entsprechenden Mittel [Lev01]. Im Kontext der WS-Architektur wurden solche Choreografien zunächst wenig beachtet und stattdessen Konversationen fokussiert (s. o.).[150] In einem neueren Ansatz wurde dies aber nachgeholt: **WS-CDL** (Web Services Choreography Description Language) ist eine XML-basierte Sprache zur Beschreibung von Peer-to-Peer-Kollaborationen zwischen Partnern auf Basis der Definition ihres üblichen und komplementär wahrnehmbaren Verhaltens aus globaler Perspektive, wobei der geordnete Nachrichtenaustausch im Erreichen eines gemeinsamen geschäftlichen Ziels resultiert [KBR04]. Die Spezifikationen der Web Service-Architektur dienen dabei als Interoperabilitätsplattform; sie werden jedoch nicht zwingend vorausgesetzt. WS-CDL enthält verschiedene innovative Abstraktionen sowohl in Bezug auf Kollaborationen als auch auf Koordinationsprotokolle, die u. a. die Korrelation und den Abgleich von Zuständen unabhängiger Partner betreffen. Das in WS-CDL verwendete Prozessmodell basiert auf einer Variante des π-Calculus (Solos Calculus). Diese formale Basis birgt erhebliches Potenzial für Verifikation und Simulation. Zurzeit befindet sich WS-CDL im Frühstadium.

Es kann festgestellt werden, dass zwischen Service-Koordination und Service-Komposition ein fließender Übergang vorliegt. Der Unterschied beruht dabei vielmehr auf der Perspektive der Anwendung als auf der Frage des Determinismus. Z. B. wird in **DySCo** die Choreografie von Web Services durch eine Workflow-Sprache auf Basis erweiterter Prozessalgebra beschrieben [PW03a]. Ein DySCo-Workflow dient dabei der Regelung einer Kollaboration zwischen Rollen. Gleichzeitig beinhaltet er alle notwendigen Informationen, um daraus Workflows zur Implementierung der einzelnen Web Services zu generieren. Ähnlich diesem Ansatz werden in **SELF-SERV** die Beziehungen zwischen Providern eines komponierten Services in Form von Zustandsdiagrammen geregelt. Auch hier dient die Spezifikation der Regelungen gleichzeitig als Vorschrift zur Steuerung der Web Service-Interaktionen zwischen den Providern. An dieser Stelle ist zunächst vor allem der Aspekt relevant, dass mit beiden Ansätzen Regelungen in Form von WS-Choreografien spezifiziert werden können. Eine genauere Beschreibung findet sich in der nächsten Sektion, in der es um Möglichkeiten zur Implementierung von Regelungen geht.

[150] Bei Bernauer et al. findet sich ein Vergleich von WS- und ebXML-basierten Möglichkeiten zur Protokollspezifikation [BKK03a].

In Ergänzung der freien Regelung soll die abhängige Regelung ausdrücken, dass Koordinationsprotokolle unter Berücksichtigung schon vorhandener öffentlicher Geschäftsprozesse der Kollaborationspartner entworfen oder geändert werden müssen, d. h. dass für einen oder alle Partner schon Spezifikationen ihrer Konversationen vorliegen. Die Aufgabe der Regelung besteht in diesem Fall darin, die Konversationen aller Partner im Sinne einer gemeinsamen Kollaboration in Einklang zu bringen. Je nachdem, für wie viele Partner Konversationen vorliegen, sind dabei unterschiedliche Ansätze zur Regelung nötig.

Liegt nur die Konversation einer einzelnen Organisation als Basis vor, besteht der verbreitetste Ansatz darin, diese Organisation aus Service-orient. Sicht als „Provider" aufzufassen und deren Konversation direkt als Regelung zu übernehmen. In diesem Fall muss ein potenzieller Kollaborationspartner (in diesem Sinne „Client") sich an diese Konversation anpassen und eine kompatible Konversation implementieren [ACK+04, S. 276 ff.].

Falls für alle Partner Konversationen spezifiziert sind, schlagen Piccinelli und Emmerich zusammen mit dem Autor der vorliegenden Arbeit **Reconciliation** als Ansatz zur dynamischen Regelung vor [PEZ+02]. Hierbei wird die Regelung unmittelbar vor der Kollaboration vorgenommen. Dazu werden die Konversationen der Kollaborationspartner miteinander verglichen. Ist keine unmittelbare Kompatibilität der Konversationen feststellbar, wird versucht, mittelbare Kompatibilität durch geringfügige Modifikationen herzustellen. Dies geschieht derart, dass die geänderten Konversationen stets im Einklang mit den ursprünglichen Implementierungen bleiben. Es wird also geprüft, ob mit den bestehenden Implementierungen eine leicht abgeänderte Kollaboration durchgeführt werden könnte. Hierbei muss dann aber auf Anwendungsebene geprüft werden, ob die geänderte Kollaboration im Sinne der Prozessintegration noch akzeptabel ist.

Ansätze zur Implementierung von Kollaborationsregelungen Wie vorangehend dargestellt wurde, können Kollaborationsregelungen durch Service-orient. Ansätze in ganz verschiedener Art und Weise erfolgen. Das Ergebnis ist jedoch stets ein Koordinationsprotokoll, das das Verhalten von Services der Kollaborationspartner spezifiziert. Die Implementierung derartiger Kollaborationsregelungen beinhaltet dann die Implementierung der daran beteiligten Services nach den Vorgaben der Koordinationsprotokolle. Hierbei können ganz generell manuelle und automatische Ansätze unterschieden werden.

Bei der manuellen Implementierung von Interaktionsprozessen ist entweder kein Koordinationsprotokoll für die Kollaboration vorhanden oder Letzteres steht in keiner formalen Beziehung zum Kompositionsmodell. Entsprechende Ansätze bieten meist sehr vielseitige Kompositionsmodelle an. Mit ihnen ist es möglich, für ein breites Spektrum verschiedener Koordinationsmodelle konforme Kompositionen zu realisieren. Die Implementierung und Validierung erfolgt dabei überwiegend manuell.

In diese Kategorie fallen die meisten und verbreitetsten Standards/Spezifikationen für WS-Kompositionssprachen wie z. B. **XLANG** [Tha01], **WSFL** [Ley01] und **BPEL** [ACD+03]. Neben der komplett manuellen Implementierung kann hier eine gewisse Teilautomatisierung durch *Template-basierte Ansätze* erreicht werden. Ausgangspunkt sind Service-Koordinationsprotokolle, die entweder direkt den abstrakten Prozess eines Partners spezifizieren oder dessen Ableitung aus einer Choreografie erlauben. Bei geeignetem Koordinationsprotokoll lässt sich die Spezifikation des abstrakten Prozesses als Template für ein Service-Kompositionsschema verwenden [ACK+04, S. 276 ff.]. Dieses Template ist dann so zu erweitern, dass alle nicht-deterministischen Aspekte des abstrakten Prozesses konkretisiert werden, so dass ein ausführbares Schema entsteht. Für derartige Ansätze kann z. B. BPEL verwendet werden, das gleichzeitig die Spezifikation von abstrakten Prozessen und Kompositionsschemata erlaubt.

Ein früher und viel beachteter Ansatz soll hier noch etwas ausführlicher betrachtet werden. Gemeint ist das ***eFlow-Framework*** von Casati, Shan u. a. [CS01, CIJ+00b, CIJ+00a, CSS01]. Konkret bildet eFlow eine Web Service Composition Middleware mit Unterstützungsmechanismen zur Spezifikation, Realisierung und zum Management komponierter Services. Ein Schwerpunkt liegt dabei auf dem vielseitigen Kompositionsmodell. Dieses umfasst zunächst ein generisches Komponenten- und Orchestrierungsmodell auf Basis von Aktivitätsdiagrammen. Darauf sind verschiedene innovative Konzepte für dynamische Aggregation, Transaktionskontrolle und Ereignisbehandlung aufgesetzt.

Ein komponierter Service wird durch ein Prozessschema beschrieben, das atomare und komponierte Services kombiniert. Die Modellierung des Schemas geschieht in Form eines Graphen, welcher die Ausführungsreihenfolge der Prozessaktivitäten an seinen Knotenpunkten definiert. Es wird zwischen Service-, Entscheidungs- und Ereignisknoten unterschieden. Service-Knoten repräsentieren den Aufruf eines atomaren oder komponierten Service und enthalten eine Suchanfrage, die bei Aufruf des Knotens ausgeführt wird, um die Referenz eines passenden Service zu ermitteln. Entscheidungsknoten regeln den Kontrollfluss, während Ereignisknoten diverse Ereignisse senden und empfangen können.

Eine Service-Prozessinstanz ist die Realisierung eines Prozessschemas. Komponierte Services werden durch einen *Service Process Composer* spezifiziert und mittels einer *Service Process Engine* ausgeführt. Die Service Process Engine besteht aus den Teilen *Planer (Scheduler)*, *Ereignis-Manager (Event Manager)* und *Transaktions-Manager (Transaction Manager)*. Der Scheduler erhält Nachrichten von Service-Knoten. Er kontaktiert daraufhin einen *Service Process Broker*, um einen Provider zu finden, der die im Service-Knoten geforderten Leistungen erbringen kann. eFlow stellt einen Standard Broker zur Verfügung. Der Benutzer kann aber auch eigene Broker verwenden. Der Ereignis-Manager überwacht das Auftreten verschiedener durch Nachrichten übermittelter Ereignisse und reagiert entsprechend darauf. Der Transaktions-Manager ermöglicht die atomare Ausführung von Teilen des Prozesses. Im Falle

abgebrochener Transaktionen werden spezifische Kompensationen durchgeführt.

Im Gegensatz zu den bislang betrachteten Implementierungsansätzen richten sich Ansätze zur automatischen Implementierung an einem bestimmten Koordinationsmodell aus. Hierbei können Kompositionsschemata aus Koordinationsprotokollen abgeleitet werden. Anders herum ist zum Teil auch eine Validierung von Kompositionsschemata in Bezug auf Koordinationsprotokolle möglich.

Ein automatischer Ansatz zur Implementierung von Kollaborationen anhand einer vorgegebenen Regelung findet sich im ***DySCo-Framework*** *(DYnamic Service COmposition)*, das von Piccinelli, Williams u. a. in den HP-Labs Bristol erarbeitet wurde [PW03a, PFW03, PDVM01]. Das charakteristische Merkmal dieses Ansatzes liegt in der Erweiterung des Ressourcenmodells von Workflow-Konzepten. Bei so genannten Service-orient. Workflows repräsentieren einzelne Aktivitäten nicht die Interaktion des Workflow (also des komponierten Service) mit einer einzelnen Service-Komponente, sondern die Interaktion beliebiger Service-Komponenten untereinander. Daher repräsentiert ein solcher Workflow keinen komponierten Service, sondern eine Choreografie.

Ein DySCo-Workflow (in DySCo-Terminologie als Komposition bezeichnet) modelliert ein deterministisches Koordinationsprotokoll zwischen Service-Komponenten. Das Orchestrierungsmodell basiert dabei auf einer Prozessalgebra im Sinne von CSP. Hier werden jedoch Ressourcen durch Rollen interagierender Partner ersetzt. Jede Aktivität des DySCo-Workflows spezifiziert beliebige P2P Interaktionen zwischen diesen Rollen.

Die Implementierung einer derart spezifizierten Choreografie (in DySCo-Terminologie als Koordination bezeichnet) erfolgt durch automatische Generierung von Kompositionsschemata. Zunächst erfolgt dazu eine (dynamische) Aggregation der Rollen durch konkrete Provider. Danach wird aus dem DySCo-Workflow für jeden Provider genau ein Kompositionsschema generiert. Das Orchestrierungsmodell ist hierbei eine Prozessalgebra des WfMC-RM. Je nach Zuweisung von Rollen (einer oder mehrerer) zu Providern ändern sich deren Kompositionsschemata. Die dadurch bewirkten P2P-Interaktionen sind jedoch stets mit denen der Choreografie identisch. Zur Ausführung verfügt jeder Provider über eine eigenständige Composition Engine: In diesem Fall werden diese durch WfMC-konforme WfMS realisiert.

Einen weiteren Ansatz dieser Art liefert ***SELF-SERV*** *(compoSing wEb accessibLe inFormation and buSiness sERvices)*. In diesem Rahmen schlagen Benatallah, Dumas u. a. einen verteilten Ansatz für Web Service Composition Middleware vor [BSD03, BDM02]. Das Kompositionsmodell geht dabei von speziellen Service-Komponenten aus, die Web Services um generische Zugriffsoperationen erweitern. Das Orchestrierungsmodell spezifiziert die Aufrufreihenfolge dieser Komponenten durch Zustandstabellen. Darauf basiert eine Peer-to-Peer-Laufzeitumgebung, in der die Ausführung des Orchestrierungsprozesses über eine Reihe von Peer-Engines (*Koordinatoren*) verteilt ist.

Ein SELF-SERV-Koordinator ist eine Komponente zur zeitlichen Planung (Scheduling) der Service-Ausführung. Hierbei wird entschieden, ob ein Zustand innerhalb einer Zustandstabelle betreten/verlassen werden soll und was in diesem Fall geschieht. Das Wissen, welches ein Koordinator zur Beantwortung dieser Fragen zur Laufzeit benötigt, wird statisch aus der Zustandstabelle, welche die Operationen des zusammengesetzten Dienstes beschreibt, extrahiert und in Form von Routing-Tabellen repräsentiert. Ein Koordinator ist jeweils für eine bestimmte Anzahl von Services zuständig.

Jeder in SELF-SERV zum Einsatz kommende Service, egal ob elementar oder zusammengesetzt, wird durch eine Wrapper-Klasse des SELF-SERV-Framework umschlossen. Hierdurch erhalten Services eine generische Schnittstelle mit Operationen wie *instantiate*, *start*, *cancel*. Hierüber nimmt der Wrapper Anfragen zur Ausführung des Service entgegen. Dabei regelt er zugleich die Konvertierungen zwischen dem Datenmodell der Service-Schnittstelle und seiner Implementierung.

Die Ausführung eines zusammengesetzten Service beinhaltet dann hauptsächlich Peer-to-Peer Interaktionen zwischen den einzelnen Koordinatoren, basierend auf der Zustandstabelle [BDF+01]. Der Koordinator für den ersten Zustand ruft dabei den ersten Service auf, indem er mit dessen Wrapper kommuniziert. Sobald die Ausführung des Service beendet ist, benachrichtigt der Koordinator alle Koordinatoren von Zuständen der Zustandstabelle, die als nächstes betreten werden müssen, über die erfolgreiche Ausführung. Dieser Vorgang wird so lange fortgesetzt, bis der Endzustand erreicht und damit alle nötigen Schritte ausgeführt wurden.

Der Vorteil von SELF-SERV ist, dass hier gegenüber Architekturen mit einer zentralen Prozess-Engine zur koordinierten Ausführung aller Service-Instanzen die potenzielle Gefahr eines Flaschenhalses minimiert wird. Zudem ist die Architektur von vornherein auf das Verfolgen des Service-Status zugeschnitten [DFB+01], was der Analyse des Systems und der darauf ausgeführten Prozesse entgegenkommt. Ein Nachteil ist, dass durch die proprietären Zustandstabellen gegenüber der Verwendung von Workflows eine Umsetzung von meist durch Workflows modellierten Unternehmensprozessen erschwert wird. Zudem handelt es sich beim hier verwendeten Service-Begriff wie bei den schon beschriebenen Frameworks um einen eher technisch orientierten Systembegriff.

3.5. Zusammenfassung

In den vorangegangenen Sektionen wurde eine Überleitung von der ökonomischen auf die technische Sicht der Dienstleistungsproduktion in virt. Unternehmen vollzogen. Diese Überleitung wurde als allgemeine Betrachtung integrierter IV im VDU begonnen, dann im Sinne der Zielsetzung auf den Aspekt generischer Integrationsmechanismen eingegrenzt und schließlich in Bezug auf den Anwendungskontext für die Technikfamilie des Service Oriented Computing konkret ausgeführt.

Zunächst wurde das Gebiet der betrieblichen Informationsverarbeitung (Englisch "Information Systems") als Rahmen der weiteren Betrachtungen abgesteckt. Hierbei wurde der Aspekt der Systemintegration als wichtiger Faktor bei der Entwicklung und Evolution von AS in dynamisch-metamorphen Systemlandschaften hervorgehoben. Im Anschluss wurde die Bandbreite informationstechnischer Unterstützungspotenziale im VDU abgesteckt. Dabei zeigte sich, dass die Anforderungen sich hier u. a. auf eben den Aspekt der Integration konzentrieren. In diesem Bereich führen die Rahmenbedingungen der virtuellen Organisation und Dienstleistungsproduktion zu hohen Erwartungen an die Fähigkeiten und Eigenschaften der IT-Infrastruktur. Hierfür wurden verschiedene Kategorien identifiziert. Von denen haben sich besonders Kategorien der horizontalen Basisinfrastruktur für die Zielsetzung der vorliegenden Arbeit als relevant erwiesen. Techniken zur Virtualisierung von Dienstleistungen lassen sich hierin nämlich als spezifische Integrationstechniken einordnen. Genauer bilden sie eine koordinative Ebene zur Integration von Dienstleistungsprozessen. In diesem Zusammenhang wurde die Wahl des Service-Paradigmas zur Abstraktion partieller Vorgänge der Dienstleistungsproduktion untermauert.

Nach Einordnung und Abgrenzung verschiedener E-Service-Techniken innerhalb der ganzheitlichen IT-Infrastruktur von VDU konzentrierte sich die Untersuchung auf eine Vertiefung der identifizierten Klasse koordinativer Integrationstechniken für DLP. Hieraus wurden drei grundlegende Techniken beleuchtet, die verschiedene komplementäre Aspekte einer konkreten E-Service-Technologie abdecken: *Enterprise-Architektur* und insbesondere das GERAM Enterprise Architecture Framework liefern ein Referenzmodell u. a. für die methodische Eingliederung generischer und fachlicher IT-Mechanismen in den betrieblichen Kontext. *Cl/Srv-basierte Middleware* beinhaltet allgemeine praktische Konzepte zur systemtechnischen Unterstützung der Interoperabilität und Integration verteilter System-Komponenten in und zwischen Unternehmen. *WfMS und IEWS* liefern spezifische Teilkonzepte einer Middleware zur prozessbasierten Integration im Allgemeinen und für DBP im Speziellen. Im Zuge der Betrachtungen erschienen die abstrakten Konzepte von GERAM als ausreichend generisch, um damit auch das geplante Vorgehensmodell zur Anwendung von E-Services in einen übergreifenden betrieblichen Kontext einzubetten. Hingegen zeigten sich die sehr viel praktischeren Konzepte konventioneller Middleware für die zwischenbetriebliche Prozessintegration im VDU als suboptimal. Dies liegt vor allem daran, dass Workflow-basierte Integrationsplattformen im Allgemeinen weder Abstraktionen noch Middleware-Mechanismen bereitstellen, die in ausreichender Weise den organisationsübergreifenden Kontext berücksichtigen und erfolgreich standardisiert sind. Gleichwohl bilden sie die konzeptionelle und praktische Basis für entsprechende Erweiterungen.

Als derartige Erweiterung wurde das Paradigma des Service Oriented Computing vorgestellt. Dieses basiert auf der weithin anerkannten Web Service-Abstraktion. Rund um Web Services wurden die wesentlichen Mechanismen konventioneller Midd-

leware für den zwischenbetrieblichen Bereich angepasst und erfolgreich standardisiert. Die Basis dazu bilden SOM, die die wesentlichen Konzepte beschreiben und SOAs, die deren Realisierung in Bezug auf das Zusammenspiel mit organisationsinternen und -externen Komponenten darstellen. Dabei fügt sich die WS-Middleware verschiedener Unternehmen zu einer organisationsübergreifenden Middleware zusammen. Neben grundlegenden Interoperabilitätsmechanismen der WS-Architektur beinhaltet dies insbesondere auch prozessorientierte Erweiterungen zur Integration verschiedener Services. Hier wurden die fundamentalen Konzepte der Service-Koordination und -Komposition vorgestellt. Auf Basis dieser SOC-Techniken wurde gezeigt, wie verschiedene grundsätzliche Probleme Workflow-basierter Integrationsplattformen überwunden werden können. Daneben wurden vielfältige Möglichkeiten zur Unterstützung zwischenbetrieblicher Prozessintegration bei der Regelung und Implementierung von Kollaborationen aufgezeigt. Diese werden im weiteren Verlauf zur Integration von DLP im VDU angewendet.

4. Konzeption eines Service-orient. Rahmenwerks für VDL

4.1. Zielsetzung

Nach der ausführlichen Darstellung ökonomischer und technischer Grundlagen in den beiden vorangegangenen Kapiteln soll in diesem nun der wesentliche konzeptionelle Anteil der Arbeit folgen. Dies betrifft vor allem die Konzeption eines Service-orient. Rahmenwerks für virt. Dienstleistungen. In Bezug auf dieses konzeptionelle Rahmenwerk sollen in den Sektionen dieses Kapitels drei inhaltlichen Schwerpunkte untersucht werden:

- *konzeptionelle Modelle*, die den Begriff der virtuellen Dienstleistung definieren,
- *Service-orient. Architektur* als Grundlage für den Entwurf von Service-orient. Anwendungssystemen (E-Services) zur Produktionssteuerung virt. Dienstleistungen,
- *Service-orient. Vorgehensmodelle* zur Regelung virt. Dienstleistungsproduktion durch Entwicklung von E-Services.

In diesen drei Bereichen sollen jeweils sowohl abstrakte Überlegungen als auch exemplarische Konkretisierungen thematisiert werden. Für die Konkretisierungen liefert das Forschungsprojekt FRESCO[1] einen durchgehenden Rahmen, der die Betrachtungen der vorliegenden Arbeit von hier an begleiten soll.

Die Untersuchung der konzeptionellen Aspekte beginnt in Sektion 4.2 mit dem fundamentalen Ziel eines interaktionsbasierten konzeptionellen Modells virt. Dienstleistungen. In diesem Modell sollen Dienstleistungen durch informationstechnische Substitution der Interaktionsbeziehungen im Dienstleistungsprozess virtualisiert werden. Der Nutzeneffekt dieser Substitution soll wiederum die virt. Dienstleistungsproduktion begünstigen. Konkret beruht dies auf einer Regelung der Koordination kooperativer Vorgänge autonomer Dienstleistungsunternehmen durch Planung und Steuerung ihrer Interaktionsvorgänge und -abläufe in Form von virt. Dienstleistungsprozessen. Dieses Konzept soll zunächst in abstrakter Form ausgearbeitet und dann anhand des exemplarischen FRESCO-Dienstleistungsmodells in verfeinerter Form veranschaulicht werden.

[1] Die Einführung des FRESCO-Projekts erfolgt in Sektion 4.2.2.

Aufbauend auf dem konzeptionellen VDL-Modell wird als nächster Schritt die Realisierung virt. Dienstleistungsprozesse auf Basis Service-orient. Techniken angestrebt. Hierzu sollen in Sektion 4.3 zunächst die grundsätzlichen Voraussetzungen zur Realisierung virt. Dienstleistungen im Kontext der integrativen IT-Infrastruktur von VDU untersucht werden. Das Ziel besteht in der Abgrenzung einer entsprechenden Klasse von Integrationstechniken als E-Services und deren Nutzung im Kontext virt. Dienstleistungsproduktion als E-Service-Management. Diese Grundlagen sollen dazu verwendet werden, die Eignung von SOC-Techniken als E-Service-Grundlage zu evaluieren. Als konkretes Beispiel hierfür soll dann die Entwicklung des Service-orient. Modell für E-Services in FRESCO betrachtet werden. Dies schließt insbesondere das FRESCO E-Service-Metamodell ein, das eine konkrete SOA zum Entwurf Service-orient. Realisierungen für E-Services vorgibt.

Am Ende des Kapitels soll in Sektion 4.4 untersucht werden, wie Dienstleistungsmodell und E-Service-SOA im Rahmen eines virt. Dienstleistungsunternehmen konkret zum Einsatz gebracht werden können. Zur Orientierung soll hier zunächst eine Einordnung in den ganzheitlichen Kontext eines Enterprise Architecture Framework erfolgen. In diesem Rahmen ist ein Vorgehensmodell für das E-Service-Management herzuleiten. Die Grundlage dazu soll die Herleitung eines allgemeinen Service-orient. Entwicklungslebenszyklus für E-Services bilden, der dem VDU-Lebenszyklus angepasst ist. Dieser Entwicklungslebenszyklus ist dann exemplarisch in Form des FRESCO E-Service-Management-Vorgehensmodell zu konkretisieren.

4.2. Konzeptionelle Modellierung virt. Dienstleistungen (VDLs)

Auf dem Weg hin zu einer IV-Unterstützung der Dienstleistungsproduktion in VDU steht an erster Stelle die Analyse und Modellierung der Anwendungsebene. Durch die Analyse sollen Struktur und Mechanik des Dienstleistungsbegriffs und der Dienstleistungsproduktion im Kontext virtueller Organisation als informale Konzepte erarbeitet werden. Im Rahmen der Modellierung gilt es dann, die komplexe Realität virtueller Dienstleistungsproduktion derart zu vereinfachen, einzuschränken und zu konkretisieren, dass eine wirkungsvolle Unterstützung mittels IT-Mechanismen möglich wird. In der vorliegenden Arbeit wird diese Vorgehensweise in zwei Schritten zunächst für einen allgemeinen konzeptionellen Rahmen und dann für einen konkreten Ansatz angewandt.

Der erste Schritt analysiert und modelliert die grundsätzlichen Zusammenhänge zwischen der Virtualisierung von Dienstleistungen und ihrer Produktion unter dem speziellen Gesichtspunkt interaktionsbasierter Koordination. Hierbei müssen zunächst die verschiedenen Möglichkeiten zur Virtualisierung von Dienstleistungen grundsätzlich analysiert werden. Dabei ist deren potenzieller Nutzen für eine virtuelle Organisation der Dienstleistungsproduktion herauszustellen. Aufbauend kann dann eine konkretisierte Betrachtung in Hinblick auf die Regelung der Koordination von

VDU-Teilnehmern bei der Produktion virt. Dienstleistungen erfolgen. Die Analyse fokussiert dabei entsprechende Aspekte des Produktionsmanagements von virt. Dienstleistungsproduktionsnetzwerken (VDPN). Am Ende resultiert ein konzeptionelles Modell, das den Rahmen für die Beantwortung der in der vorliegenden Arbeit betrachteten Fragestellung bietet.

In diesem konzeptionellen Rahmen beschreibt und spezifiziert der zweite Schritt einen exemplarischen Ansatz zur Virtualisierung von Dienstleistungen im Rahmen von VDPN. Dieser Ansatz beruht auf einer Restrukturierung des DLP durch Unterscheidung von Dienstleistungsinhalten und ihrer Erbringung. Die Virtualisierung erfolgt dann durch Substitution koordinativer Interaktionsprozesse der Dienstleistungserbringung. Das Konzept geht auf ein konkretes Dienstleistungsmodell zurück, das im Rahmen des Forschungsprojektes FRESCO entstanden ist. Dieses Projekt soll hier und im weiteren Verlauf der Betrachtung zur Veranschaulichung eines durchgehenden konkreten Lösungsansatzes der in der vorliegenden Arbeit betrachteten Fragestellung dienen.

Die nachfolgenden Sektionen beschreiben die Analyse und Modellierung im Rahmen der beiden Schritte im Detail. Sektion 4.2.1 beinhaltet dabei die Basiskonzepte, während Sektion 4.2.2 sich dem spezifischen FRESCO-Ansatz widmet.

4.2.1. Virtualisierung von Dienstleistungen im VDU

Hinter dem Konzept virt. Dienstleistungsunternehmen steht die grundsätzliche Idee, die Vorteile einer virtuellen Organisationsstruktur auf die Produktion von Dienstleistungen anzuwenden. Dies wird zum einen dadurch motiviert, dass sich virt. Organisation (im instrumentalen Sinn) gerade im Fall der Dienstleistungsproduktion besonders positiv auswirkt. Zum anderen eignet sich die Dienstleistungsproduktion auch in besonderem Maße zur Anwendung dieser Organisationsprinzipien.[2]

Bislang wurden diese Zusammenhänge auf allgemeiner Ebene behandelt. Bei konkreterer Betrachtung stellt sich aber die Frage nach den Möglichkeiten der Ausgestaltung virt. Organisation in Dienstleistungsunternehmen. Hier ist zu klären, auf welche Weise die Virtualisierung der Organisationsstrukturen bei der Dienstleistungsproduktion bewirkt werden soll. Dabei muss vor allem die Art und Weise konkretisiert werden, mit der sich die spezifischen Eigenschaften von Dienstleistungen optimal einsetzen lassen.

Eine mögliche Antwort soll nun durch verschiedene Konzepte zur Virtualisierung von Dienstleistungen und zu deren konkreter Produktion in VDU gegeben werden. Hierbei werden in Sektion 4.2.1.1 Nutzen und Anwendung von Produktvirtualisierung für virt. Unternehmen dargestellt und auf Dienstleistungsvirtualisierung übertragen. Danach erfolgt in Sektion 4.2.1.2 eine Diskussion alternativer Virtualisierungsstrategien für Dienstleistungen. Schließlich wird in Sektion 4.2.1.3 ein konzeptionelles Basismodell der Dienstleistungsproduktion in VDU auf Basis von UML formuliert.

[2] Dies wurde in Sektion 2.3.2.2, S. 51 ausführlich dargestellt.

4.2.1.1. Dienstleistungsvirtualisierung als VDU-Faktor

In strategischen Netzwerken bilden die Geschäftsprozesse teilnehmender Netzwerkunternehmen einen gemeinsamen Pool von Kernkompetenzen im Sinne funktionaler Spezialisierung. Diese Kernkompetenzen werden je nach Anforderungen von Kunden und Märkten zu maßgeschneiderten Lösungen integriert. Innerhalb des Netzwerks wird es durch wechselseitiges Vertrauen und die Integration von Informationssystemen (im weiteren Sinn) ermöglicht, die Wertschöpfungsketten zu restrukturieren, wann immer sich die internen oder externen Anforderungen ändern. Im Extremfall virt. Unternehmen ist jede denkbare organisatorische Struktur innerhalb des Netzwerks eine virtuelle Fähigkeit. Die extreme Flexibilität wird dabei durch den Verzicht auf eine Institutionalisierung von Managementfunktionen erreicht. Dies muss jedoch durch IT-gestützte Mechanismen kompensiert werden. Je dynamischer das Unternehmensnetzwerk ausgelegt ist, desto mehr ist es auf eine integrierte IT-Infrastruktur aller Netzwerkunternehmen angewiesen. Ganz allgemein muss eine solche Infrastruktur horizontale und vertikale Mechanismen zur Integration von Netzwerkunternehmen bereitstellen. Horizontale Mechanismen erbringen generische Funktionen, die die Netzwerkunternehmen zur Kommunikation und Kooperation befähigen. Vertikale Mechanismen erbringen domänenspezifische Fachfunktionen zur Unterstützung des VU-Lebenszyklus.

Effektivität und Effizienz der informationstechnischen VU-Infrastruktur stehen in direktem Zusammenhang zur Charakteristik des virt. Unternehmens selbst. Z. B. ist es intuitiv unmittelbar ersichtlich, dass IT-gestützte Integrationsmechanismen grundsätzlich effektiver im Publikationswesen als in der Textilindustrie zur Anwendung gebracht werden können. Hierbei ist u. a. das Potenzial zur Virtualisierung der Produkte von Bedeutung. Die informationstechnische Substitution von Produkteigenschaften wirkt sich nämlich auf das informationstechnische Potenzial zur Integration von Netzwerkunternehmen eines entsprechenden virt. Unternehmens aus. Dabei hängt die Effektivität auch von der Art der Virtualisierung ab. Z. B. ist es im Publikationswesen möglich, die Manuskripte von Autoren in digitaler anstatt in physikalischer Form vorzuhalten. Dadurch können verschiedene Anteile der Produktion wie Lektorat, Layout und Druck sehr effizient durch IT-Mechanismen integriert werden. Bei der Textilproduktion können z. B. digitale Produktmuster eingesetzt werden, um verschiedene Funktionen aus Planung, Entwurf, Beschaffung oder Distribution zusammenzuführen. Diese Variante der Virtualisierung wirkt sich jedoch deutlich geringer auf die Produktion aus.

Entsprechend wird das Potenzial virtueller Organisation im Falle genereller Dienstleistungsproduktion ebenfalls stark vom Charakter des Dienstleistungsprodukts bestimmt. Ganz allgemein ergeben sich die konstitutiven Merkmale von Dienstleistungen aus den drei möglichen Betrachtungsweisen als Potenzial, Prozess und Wirkung. Die prozessorientierte Perspektive charakterisiert den Dienstleistungsprozess selbst als nutzenstiftend. Der Wert liegt dabei in der Wirkung der Prozessaktivitäten und wird

daher im Verlauf des DLP kontinuierlich vom Dienstleistungsunternehmen an den Kunden übergeben. In diesem Sinne sind Dienstleistungen weitgehend immateriell und flüchtig. Diese Eigenschaft wird dahingehend ausgelegt, dass Dienstleistungen grundsätzlich ein hohes Potenzial zur Virtualisierung zugesprochen wird [PN98]. Dies betrifft den Dienstleistungsprozess sowie die darin enthaltenen Aktivitäten. Darüber hinaus tragen die Kunden eines Dienstleistungsunternehmens auch selbst als externe Produktionsfaktoren zur Dienstleistungserstellung bei. Der Dienstleistungsprozess ist dementsprechend durch einen hohen Grad an Interaktivität gekennzeichnet. Diese Eigenschaft führt dazu, dass sich die Virtualisierung in hohem Maße auf Effektivität und Effizienz der Dienstleistungsproduktion auswirkt. Die informationstechnische Abbildung des DLP hat nämlich direkten Einfluss auf die Interaktion von Netzwerkunternehmen und Kunden.

Das Potenzial zur Virtualisierung von Dienstleistungen bildet also einen wesentlichen Ansatzpunkt für die Gestaltung virt. Dienstleistungsunternehmen. Die informationstechnische Realisierung nutzenstiftender Dienstleistungsvorgänge begünstigt deren Integration in Produktionsnetzwerke. Darüber hinaus birgt eine informationstechnische Repräsentation des Dienstleistungsprozesses nicht nur Potenziale zur Effektivitätssteigerung der Dienstleistungsproduktion durch Erhöhung der Interaktivität. Ein in diesem Sinne *virt. Dienstleistungsprozess* bildet darüber hinaus eine Möglichkeit zur flexiblen Integration von Netzwerkunternehmen sowie von Kunden. Entsprechende *virt. Dienstleistungen* bilden somit nicht nur die Voraussetzungen für informationstechnische Infrastrukturen virt. Dienstleistungsunternehmen, sondern verkörpern selbst Teile einer solchen.

4.2.1.2. Ansatzpunkte der Dienstleistungsvirtualisierung

Das grundlegende Merkmal der Virtualisierung besteht in der Ersetzung von physikalischen Objektmerkmalen durch nicht-physikalische Zusatzmerkmale. In diesem Sinne können Dienstleistungen auf verschiedene Weise virtualisiert werden. Als Basis des nachfolgenden Entwurfs eines VDL-Modells sollen in dieser Sektion verschiedene Virtualisierungskonzepte für Dienstleistungen analysiert werden.

Abb. 4.1 knüpft an die Darstellung der grundlegenden DLP-Struktur an.[3] Die Darstellung zeigt verschiedene Ansatzpunkte der Virtualisierung. Fundamental kann die Virtualisierung entweder auf Basis der Prozessaktivitäten oder der dazwischenliegenden Interaktionen erfolgen. Aufbauend sind Mischformen denkbar. Hier erscheint besonders die informationstechnische Abbildung der Aktivitäten und Interaktionen im Steuerprozess sinnvoll. Im Folgenden werden die Varianten einzeln diskutiert.

Im Kontext des relativ abstrakten Konzepts des Dienstleistungsprozesses sind die Prozessaktivitäten intuitiv am besten zu erfassen. Es handelt sich dabei um interne Prozesse des DLU oder Kunden. Auf der Ebene des DLP treten diese als

[3] Die DLP-Struktur wurde in Sektion 2.3.1.2, S. 41 ausführlich beschrieben.

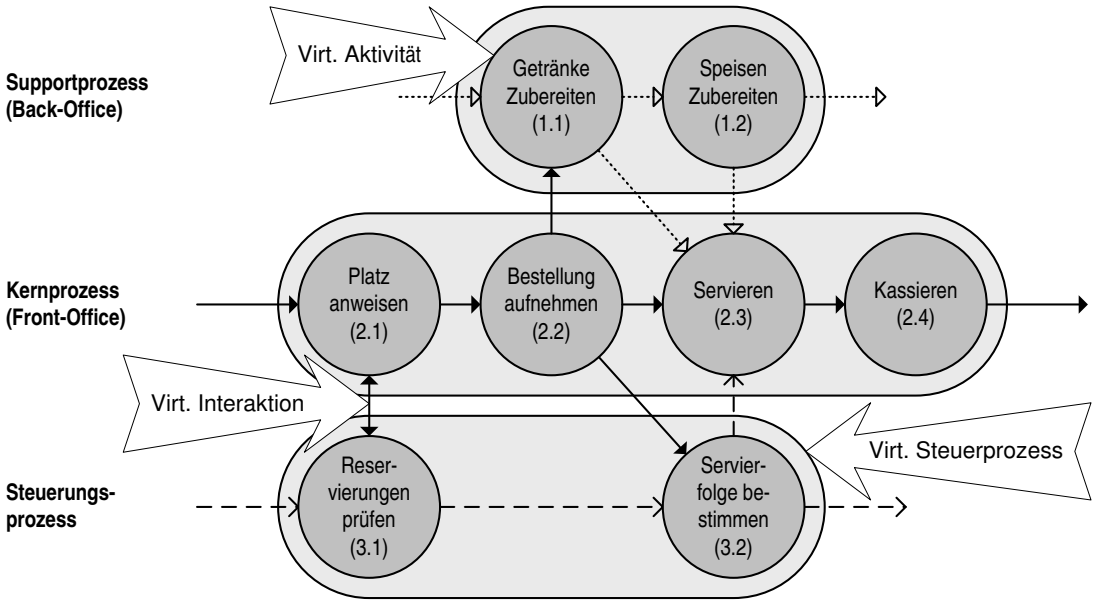

Abbildung 4.1.: Verschiedene Ansatzpunkte zur Virtualisierung des DLP

abgeschlossene Vorgänge auf. Sie bieten sich daher direkt zur Substitution durch IV-Mechanismen an. Freilich ist der Grad der Virtualisierung hier von der Art der Aktivitäten abhängig. Handelt es sich dabei um rein informationelle Primär- oder Sekundärprozesse, so kann der Vorgang unter Umständen vollständig automatisiert werden. Bei materiellen oder Innovationsprozessen ist dies in der Regel nicht möglich bzw. auch nicht wünschenswert. Gleichwohl können solche Vorgänge durch informationstechnische Modellierung sehr wohl unterstützt werden. In diesem Zusammenhang ist auch die Positionierung eines Vorgangs im DLP von Bedeutung. In vielen Fällen sind Vorgänge des Back Office materieller Natur. Aktivitäten im Steuerprozess sind hingegen überwiegend informationell. Diese Inhomogenität der Prozessaktivitäten muss bei einem Virtualisierungskonzept berücksichtigt werden. In Bezug auf die Integrationsaspekte von VDU ist vor allem die informationstechnische Repräsentation von DLP-Aktivitäten durch Modelle und Kommunikationsschnittstellen relevant. Modelle begünstigen die Selektion von Netzwerkunternehmen zur Initialisierung von VDU. Kommunikationsschnittstellen helfen bei der Integration der Vorgänge während der Dienstleistungsproduktion.

Der Dienstleistungsprozess ist durch hohe Interaktivität der darin handelnden Akteure geprägt. Im Rahmen des Kernprozesses finden Interaktionsvorgänge zwischen DLU und Kunden statt. Darüber hinaus sind die Prozessaktivitäten verschiedener Akteure im Dienstleistungsprozess generell durch Interaktionen dieser Akteure verknüpft. Es ist unmittelbar einleuchtend, dass die im Rahmen von Interaktion anfallende Kommunikation sich relativ leicht informationstechnisch abbilden lässt.

Daneben besteht auf übergeordneter Ebene ein entscheidender Aspekt in der Abbildung der Strukturmuster des ganzheitlichen Interaktionsprozesses. Hiermit ist das Schema für den Ablauf der vielfältigen Interaktionen im Verlauf der gesamten Dienstleistung gemeint. Interaktionsmuster sind in realen Dienstleistungsprozessen mehr oder weniger stark ausgeprägt und generell lediglich implizit präsent. Die Virtualisierung des Interaktionsprozesses trägt dazu bei, deren Interaktionsmuster explizit zu machen. Darüber hinaus können verschiedenartige Mechanismen zur Unterstützung der Kooperation zwischen Akteuren zum Einsatz kommen. Dies reicht von der Unterstützung unstrukturierter Interaktionsprozesse durch CSCW bis hin zur Automatisierung hoch strukturierter Interaktionsprozesse durch WfMS. Insgesamt bilden Mechanismen zur Virtualisierung der Dienstleistungsinteraktion durch ihren primären Integrationsaspekt in direkter Weise Teile einer horizontalen VDU-Infrastruktur.

Zur Virtualisierung dienstleistungsspezifischer Interaktionsprozesse ist es hilfreich, sich an der konzeptionellen DLP-Struktur zu orientieren. Hiernach sind verschiedene Teilprozesse mitsamt der darin enthaltenen Aktivitäten und Interaktionen in Kategorien eingeteilt. Diese Kategorien kennzeichnen Teilprozesse in Bezug auf unterschiedliche Aspekte der Dienstleistungsproduktion. Kernprozesse und Supportprozesse bestimmen den Nutzeneffekt der Dienstleistung im funktionalen Sinne. Steuerprozesse koordinieren das Zusammenspiel der funktionalen Anteile und tragen dadurch zur Dienstleistungsqualität im nicht-funktionalen Sinne bei. Sie beinhalten dabei nicht nur überwiegend informationelle Prozessaktivitäten. Sie konzentrieren darüber hinaus auch noch die wesentlichen strukturellen Anteile des Interaktionsprozesses der Dienstleistung. Aus diesem Grund muss die informationstechnische Abbildung der Steuerprozesse als eines der wesentlichen Ziele bei einer Dienstleistungsvirtualisierung in Hinblick auf VDU betrachtet werden.

4.2.1.3. Spezifikation eines konzeptionellen VDL-Basismodells

Als Basis der nachfolgenden Darstellung eines konkreten VDL-Modells sollen in dieser Sektion zunächst einige grundlegende Begriffe der bisherigen Diskussion in expliziterer Form konzeptioneller Modelle erfasst werden. Dies betrifft zunächst den grundlegenden Dienstleistungsbegriff mitsamt des darin enthaltenen Dienstleistungsprozesses und die Organisationsform des Produktionsnetzwerks in allgemeiner und virtualisierter Form (VPN). Schließlich wird ein allgemeines Modell für die Synthese der beiden Aspekte zu *virt. Dienstleistungsproduktionsnetzwerken* formuliert. Zuvor erfolgt noch eine kurze Einführung der verwendeten Methode *konzeptioneller Modellierung* auf Basis von UML.

Konzeptionelle Modellierung mit UML Unter *konzeptioneller Modellierung* werden im Allgemeinen Techniken zur Repräsentation (eines Teils) einer komplexen Situation in abstrakter Weise und mittels präziser Notation verstanden. Entsprechende

Methoden wurden vielfach zur Spezifikation und Analyse von Benutzeranforderungen in Bezug auf Informationssysteme eingesetzt. Andere Anwendungen umfassen generell die Sammlung und Repräsentation von Informationen zur Lösung komplexer Probleme aus dem technischen oder auch organisatorischen Bereich.

Konzeptionelle Modellierung kann im Wesentlichen in funktions-, daten- oder objektorientierter Weise geschehen.[4] Heute wird vor allem die objektorientierte Modellierung betont, wobei diese Methode als evolutionäre Weiterentwicklung auf Basis verschiedener Nachteile der anderen Methoden gesehen werden kann. Objektorientierte Modelle werden allgemein als abstrakter angesehen, woraus sich leichtere Verständlichkeit, Wartung und Wiederverwendung ableiten lassen. In diesem Sinne soll auch das konzeptionelle Dienstleistungsmodell der vorliegenden Arbeit auf einer objektorientierten Methode beruhen.

Ein weit verbreitetes und bewährtes Mittel zur objektorientierten Modellierung stellt die Unified Modelling Language (UML) [Sie03] dar.[5] Die Sprache beinhaltet wesentliche Aspekte der Objektorientierung wie Kapselung und Vererbung von Objekten sowie explizite Beschreibungen ihres interaktiven und koordinierten Verhaltens. Diese Aspekte machen UML zur konzeptionellen Modellierung äußerst geeignet, da die Sprache sich leicht für eine explizite Unterstützung der Spezifikation von Anwendungseigenschaften konfigurieren lässt.

Zur Modellierung der Anwendungsdomäne zwecks Analyse von Anforderungen stellt die UML Anwendungsfalldiagramme bereit. Ein Anwendungsfalldiagramm beschreibt die Beziehungen zwischen Akteuren und Anwendungsfällen.[6] Ziel ist dabei die Beschreibung eines Systems (in abstraktem Sinn) als Menge relevanter Geschäftsvorfälle einer Domäne. Hierbei wird die Perspektive der daran beteiligten Akteure eingenommen. Anwendungsfälle repräsentieren jeweils eine Menge von Aktivitäten des betrachteten Systems mit definiertem Auslöser und Ergebnis. Akteure repräsentieren Rollen, die innerhalb des Systems von beteiligten Menschen und/oder Maschinen eingenommen werden. Die Beteiligung bezieht sich dabei auf Anwendungsfälle. Insgesamt lässt sich auf diese Weise eine aussagekräftige Systembeschreibung auf hohem Abstraktionsniveau erreichen, die sich als Ansatzpunkt für einen konkreten Systementwurf eignet.

Modellierung von Dienstleistungen Als Startpunkt zur Modellierung komplexerer Dienstleistungsmodelle soll zunächst das konzeptionelle Modell des grundlegenden abstrakten Dienstleistungsbegriffs zugrunde gelegt werden. Auf dieser Ebene sollen

[4] Siehe z. B. für funktionsorientierte Modellierung [You89], für datenorientierte Modellierung [HK87] und für objektorientierte Modellierung [Rum91].

[5] UML ist eine Spezifikation der Object Management Group (OMG) [OMG06] und wird zurzeit auf Version 2.0 [OMG05] umgestellt. In der weiteren Darstellung wird jedoch die Version 1.5 [OMG03] zugrunde gelegt, die zur Zeit der Erstellung des Modells die gebräuchlichste und stabilste Variante war.

[6] Siehe z. B. [Oes01, S. 1 ff.].

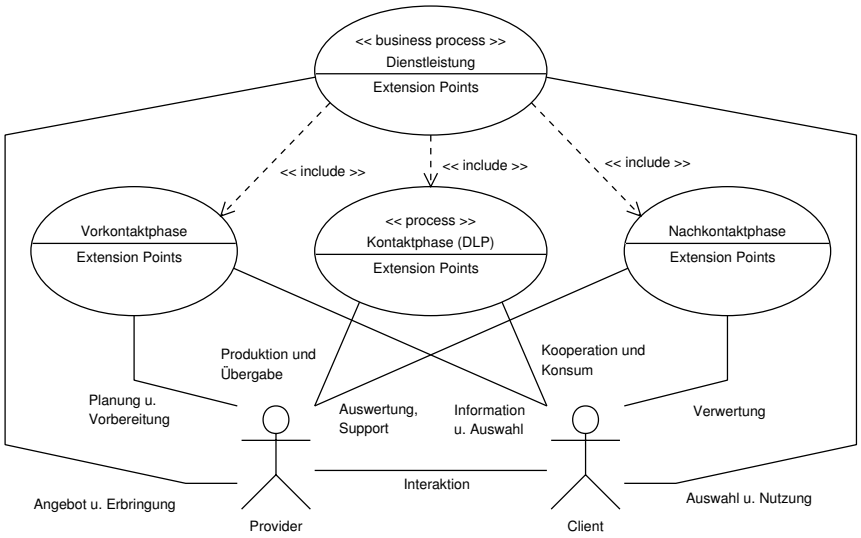

Abbildung 4.2.: Konzeptionelles Modell konventioneller Dienstleistungen

zunächst Anwendungsfalldiagramme benutzt werden. Diese eignen sich gut, um die grundsätzlichen Zusammenhänge im Kontext von Dienstleistungserbringung und Konsum in abstrakter Weise zu erfassen. Die Darstellung steht hier noch in keinem Zusammenhang zur IT. Durch die Konzepte der Modellierungssprache lässt sie sich jedoch im weiteren Verlauf diesbezüglich konkretisieren und als Anforderungsprofil verwenden.[7]

Das verwendete Anwendungsfalldiagramm ist in Abb. 4.2 dargestellt. Die wesentlichen Elemente des Modells sind durch den Anwendungsfall der `Dienstleistung` und die Akteure `Provider` und `Client` gegeben.[8] Die Modellierung von Dienstleistungen als zentrale Anwendungsfall ist im Sinne der UML dadurch gerechtfertigt, dass es sich dabei um einen Geschäftsprozess handelt,[9] der vom `Client` durch Auswahl angestoßen und durch den Nutzeneffekt wahrgenommen wird. Der `Provider` ist aus dieser abstrakten Sicht durch Angebot und Erbringung bzw. Produktion beteiligt. Eine wesentliche Charakteristik dieses Systems ist die Interaktionsbeziehung der beiden Akteure. Diese Beziehung erhält durch die `Dienstleistung` einen Rahmen.

Eine `Dienstleistung` beinhaltet im Sinne der alternativen ökonomischen Definitionsansätze die Anwendungsfälle der `Vorkontaktphase`, `Kontaktphase` und `Nachkontaktphase`. Die `Vorkontaktphase` leitet der `Provider` durch Planung und Potenzialerstellung ein.

[7]Dazu können Anwendungsfälle u. a. erweitert werden. Die Existenz von *Erweiterungspunkten* (`Extension points`) wird in der grafischen Repräsentation gekennzeichnet. Die konkreten Erweiterungen werden dann in den folgenden Diagrammen ausgeführt.

[8]Elemente aus Modellen, Systembeschreibungen, oder Quellcode werden hier und im Folgenden durch Darstellung mittels „`Schreibmaschinenschrift`“ gekennzeichnet.

[9]Dies wird im Modell durch einen Stereotyp ≪`business process`≫ verdeutlicht.

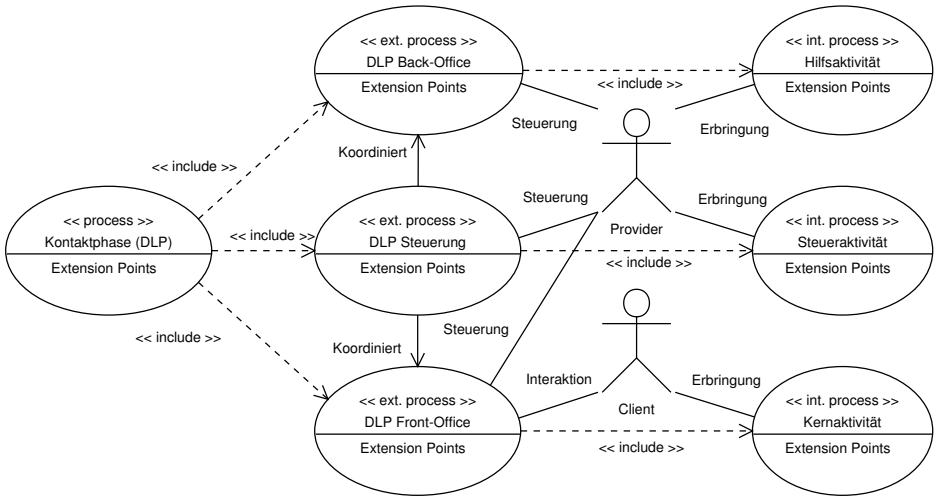

Abbildung 4.3.: Konzeptionelles Modell des DLP

Der `Client` ist hierin durch Information, Auswahl/Anforderung und ggf. Verhandlung aktiv. Die `Kontaktphase` repräsentiert den eigentlichen Dienstleistungsprozess.[10] Der `Provider` betreibt durch Abwicklung der wesentlichen Aktivitäten die Produktion dieses Prozesses. Er übergibt dessen Ergebnisse kontinuierlich an den `Client`, der dabei kooperativ mitwirkt. In der `Nachkontaktphase` tritt der `Provider` nur noch bei außerordentlichem Unterstützungsbedarf mit dem `Client` in Verbindung. Dieser verwertet hier primär die Nutzeffekte des Dienstleistungsprozesses.

Zur weiteren Konkretisierung des ökonomischen Dienstleistungskonzepts im Sinne der vorliegenden Arbeit soll noch ein zweites konzeptionelles Modell des *Dienstleistungsprozesses* betrachtet werden. Abb. 4.3 zeigt das entsprechende Anwendungsfalldiagramm. Für den schon im allg. Dienstleistungsmodell gezeigten Anwendungsfall der `Kontaktphase` werden hier die drei maßgeblichen strukturellen Anteile des DLP herausgestellt. Dies ist zunächst der Fall des Hilfsprozesses (`DLP Back-Office`), in dessen Rahmen der `Provider` entsprechende `Hilfsaktivitäten` zur Unterstützung des Kernprozesses erbringt. Im Kernprozess (`DLP Front-Office`) findet dann die Interaktion mit dem Client in Form von `Kernaktivitäten` statt. Die Steuerung des DLP ist grundsätzlich Aufgabe des `Providers`. Hierzu erbringt er im Rahmen des Steuerprozesses (`DLP Steuerung`) die notwendigen `Steueraktivitäten`. Zusätzlich koordiniert der `Provider` in allen Teilprozessen die enthaltenen Aktivitäten und stimmt diese auch zwischen Prozessen ab.

Als zusätzlicher Aspekt kennzeichnet das Modell noch explizit die Art der Vorgänge im DLP. Hierbei wird zwischen internen und externen Prozessen unterschieden.[11]

[10] Die Prozesseigenschaft wird im Modell durch einen Stereotyp ≪`process`≫ betont.

[11] Organisationsinterne Prozesse werden mit dem Stereotyp ≪`int. process`≫ gekennzeichnet,

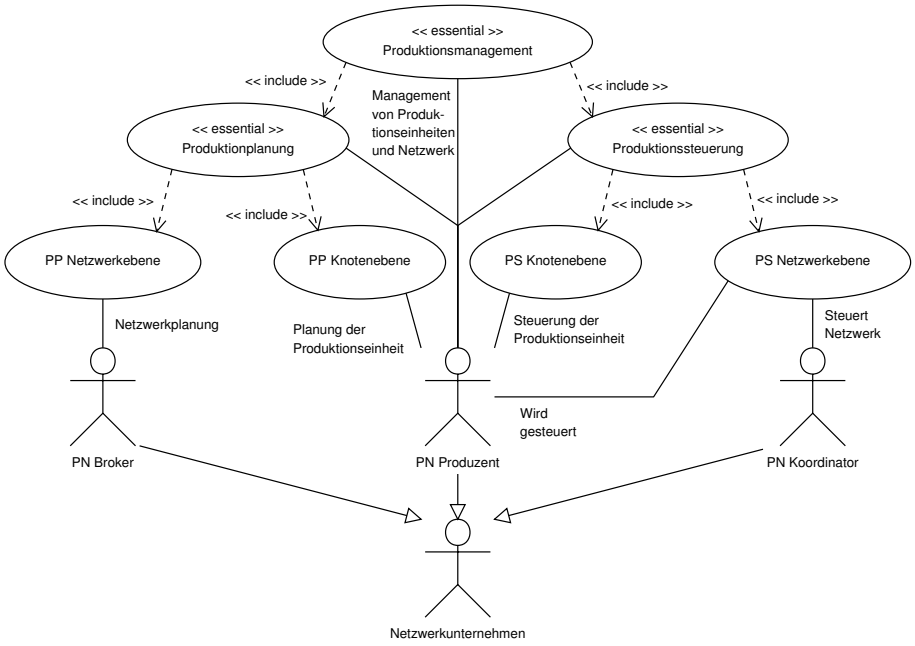

Abbildung 4.4.: Konzeptionelles Modell für Produktionsnetzwerke

Generell werden die Vorgänge im Rahmen des DLP als interne Geschäftsprozesse verstanden. Diese werden entweder vom Provider oder vom Client autonom und eigenverantwortlich erbracht. Aus globaler Perspektive sind die Abläufe innerhalb dieser Geschäftsprozesse nicht ersichtlich. Lediglich das Verhalten an der Schnittstelle dieser Prozesse kann beobachtet werden. Die Prozessanteile des DLP sind hingegen externe Prozesse. Die Abläufe beziehen jeweils beide Akteure ein und sind aus globaler Perspektive sichtbar.

Modellierung virt. Produktionsnetzwerke Da sich die vorliegende Arbeit vornehmlich dem Aspekt der Produktionssteuerung widmet, sollen in einem weiteren konzeptionellen Modell die organisatorischen Grundlagen *virt. Produktionsnetzwerke* beschrieben werden. Dazu wird in einem ersten Schritt das Konzept des generellen *Produktionsnetzwerks* erfasst. Hierauf gründet sich ein Modell der virt. Variante, das auf die Perspektive der vorliegenden Arbeit zugeschnitten ist.

Abb. 4.4 zeigt das Anwendungsfalldiagramm des konzeptionellen Modells für Produktionsnetzwerke. Der Fokus liegt dabei auf dem in Sektion 2.3.3.1 beschriebenen Aspekt des `Produktionemanagements`, das hier den zentralen Anwendungsfall darstellt. Es handelt sich hierbei um einen *essenziellen* Anwendungsfall, der die spezifischeren Aspekte der `Produktionsplanung` und `Produktionssteuerung` zusammenfasst. Letztere beinhalten wiederum jeweils Varianten auf Netzwerkebene und Knotenebene eines

organisationsexterne hingegen mit dem Stereotyp ≪`ext. process`≫.

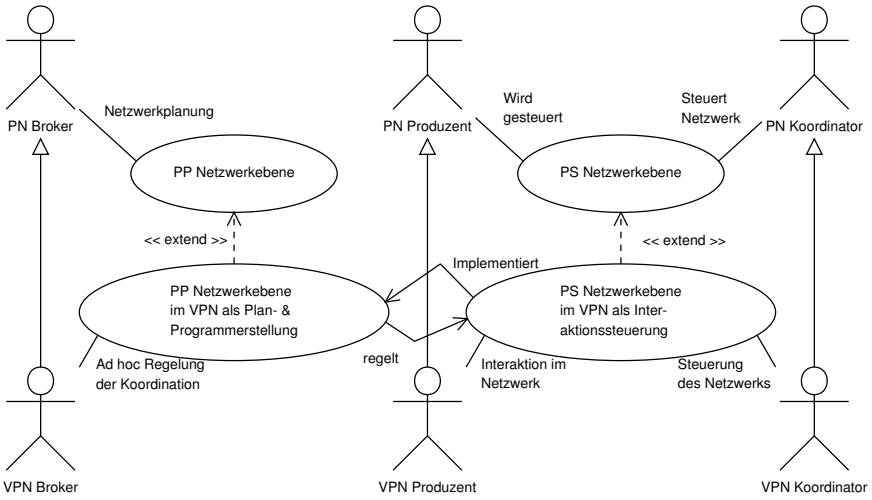

Abbildung 4.5.: Konzeptionelles Modell für virt. Produktionsnetzwerke

Unternehmensnetzwerks. Für die vorliegende Arbeit ist dabei vor allem die Koordinationsproblematik auf Netzwerkebene von Interesse. Diese Perspektive wird durch drei beteiligte Akteure hervorgehoben. `PN Broker` repräsentieren eine übergeordnete vermittelnde Instanz im Produktionsnetzwerk, der die Produktionsplanung auf Netzwerkebene (`PP Netzwerkebene`) zufällt. `PN Produzenten` repräsentieren einzelne Produktionseinheiten. Sie sind mit den Anwendungsfällen der Produktionsplanung (`PP Knotenebene`) und Produktionssteuerung (`PS Knotenebene`) auf Knotenebene befasst. Diese stehen hier für die gemeinhin geforderte Selbstverwaltung von Produktionseinheiten. Für das notwendige Mindestmaß übergeordneter Steuerung auf Netzwerkebene (`PS Knotenebene`) ist schließlich der spezielle Akteur `PN Koordinator` vorgesehen. Alle drei Akteure sind Spezialisierungen allgemeiner `Netzwerkunternehmen`. Dadurch wird zum Ausdruck gebracht, dass insbesondere die Broker- und Koordinationsfunktionen *innerhalb* des Netzwerks erbracht werden.

Im Kontext von Produktionsnetzwerken ist für die vorliegende Arbeit vor allem der Koordinationsaspekt von Interesse. Dieser wird für den Sonderfall virtueller Netzwerkorganisation – bzw. virt. Produktionsnetzwerke – betrachtet. Für ein konzeptionelles Modell dieser Perspektive wird der Ansatz zur Regelung der Koordination in VU nach Specht und Kahmann [SK00] herangezogen.[12]

Das in der vorliegenden Arbeit entworfene konzeptionelle Modell für virt. Produktionsnetzwerke ist in Abb. 4.5 dargestellt. Der zentrale Aspekt ist hierin die Konkretisierung der Produktionsplanung und -steuerung auf Netzwerkebene von VPN. Hierzu werden die generellen Anwendungsfälle aus dem Produktionsnetzwerkmodell durch entsprechende Varianten für VPN spezialisiert. Im oberen Teil der Abb. finden

[12]Siehe hierzu auch Sektion 2.3.3.2, S. 63.

sich die Elemente des Produktionsnetzwerkmodells wieder. Deren Spezialisierungen sind in der unteren Hälfte zu sehen. Entsprechend dem Ansatz von Specht und Kahmann wird die Koordination auf Netzwerkebene als (komplexer) Interaktionsprozess von Akteuren des VPN verstanden, d. h. dass die Produktionssteuerung auf Netzwerkebene durch eine Interaktion zwischen teilnehmenden Produktionseinheiten (`VPN Produzent`) realisiert und durch ausgewählte Teilnehmer (`VPN Koordinator`) gesteuert wird (`Interaktionssteuerung`). Dies stellt eine *ideelle Implementierung* der Produktionsplanung auf Netzwerkebene dar. Letztere erfolgt auf Grund der virtuellen Organisation zum Zeitpunkt der Auftragserteilung und daher ad hoc. Eine Brokerrolle im VPN (`VPN Broker`) übernimmt hierbei die ad hoc-Regelung der Koordination durch Erstellung von Programmen auf Grundlage von Plänen (`Plan- und Programmerstellung`).

Modellierung von VDPN Nachdem konzeptionelle Modelle für Dienstleistungen und entsprechende DLP sowie Produktionsnetzwerke in grundlegender und virtueller Form vorliegen, kann nun eine Kombination dieser Konzepte unter dem Gesichtspunkt der Virtualisierung von Dienstleistungen untersucht werden. Dies soll an dieser Stelle zunächst in allgemeiner Form geschehen. Ziel ist es hier, die prinzipiellen Möglichkeiten zur Koordination der Dienstleistungsproduktion in virtuell organisierten Unternehmensnetzwerken (VDPN) auf Basis virtualisierter Dienstleistungen zu erfassen. Im Zuge dessen erfolgt eine problembezogene Einführung genereller VDL in Form eines konzeptionellen Modells. Ein spezielles Modell wird dann in der folgenden Sektion mit dem FRESCO-Dienstleistungsmodell eingeführt. Dieses stellt eine Verfeinerung der hier nun betrachteten generellen Basiskonzepte dar.

Das Prinzip der Dienstleistungsvirtualisierung beruht also darauf, die Organisationsform virt. Produktionsnetzwerke auf die Produktion von Dienstleistungen anzuwenden. Auf diese Weise soll ein virt. Dienstleistungsproduktionsnetzwerk (VDPN) geschaffen werden. Um dies zu erreichen, müssen die abstrakten Organisationskonzepte des konzeptionellen VPN-Modells konkret ausgestaltet werden. Hierzu soll eine Verfeinerung des konzeptionellen Dienstleistungsmodells in Bezug auf dessen Virtualisierung erfolgen. Die Virtualisierung muss hier derart erfolgen, dass sie als Mehrwert der Substitution physischer Dienstleistungseigenschaften einen Ansatzpunkt zur virt. Organisation der an ihrer Produktion beteiligten Einheiten bietet.

Im hier betrachteten konzeptionellen Modell virt. Produktionsnetzwerke wird die notwendige Flexibilität der Netzwerkstruktur dadurch erreicht, dass die Koordination der teilnehmenden Produktionseinheiten auf Basis ihrer Interaktion erfolgt. Zur Regelung der Koordination werden Pläne und Programme erstellt. Programme beschreiben die Interaktion von Akteuren im Netzwerk. Sie können bei Bedarf sofort zur Implementierung der Interaktion zwischen Produktionseinheiten verwendet werden. Pläne legen Vorgänge zur Erstellung oder Modifikation von Programmen fest. Sie führen zu einer gesteigerten Flexibilität der Netzwerkstrukturen.

Die Produktion von Dienstleistungen erfolgt inhärent durch Interaktion des Dienstleistungsunternehmens mit den Kunden. Diese Interaktion ist ein wesentlicher Teil des Dienstleistungsprozesses, durch den sich der eigentliche Dienstleistungsbegriff definiert. Obwohl diese Tatsache an sich schon dem Koordinationskonzept im virt. Produktionsnetzwerk entgegenkommt, ist sie alleine noch nicht ausreichend. Der Dienstleistungsprozess muss nämlich nicht in jedem Fall flexibel sein. So kann der Dienstleistungsprozess im Rahmen eines Dienstleistungsunternehmens durchaus auf Basis institutioneller Strukturen erbracht werden, die sich nicht ohne Weiteres ändern lassen. An dieser Stelle soll eine Virtualisierung die Flexibilität erhöhen. Das Ziel ist es dabei, den interaktiven DLP durch einen IV-Mechanismus zu ersetzen, der durch strukturierte Methoden erstellt und durchgeführt werden kann. Der virtuelle DLP kann dann als Programm zur Koordination des VDPN aufgefasst werden und seine Entwicklungsmethodik bildet die entsprechenden Pläne.

Der erste Schritt auf dem Weg zu einem entsprechenden VDPN-Modell besteht nun in einer Erweiterung des Dienstleistungsmodells in Hinblick auf die Organisation der Dienstleistungsproduktion in Form virt. Produktionsnetzwerke. Dazu sind zunächst die Rollen im Dienstleistungsmodell zu erweitern. Auf konzeptioneller Ebene kann dies entsprechend dem in Abb. 4.6 gezeigten Diagramm modelliert werden. Der `Provider` der Dienstleistung ist in diesem Fall ein Unternehmensnetzwerk, das sich im Sinne des Produktionsnetzwerkmodells aus `Netzwerkunternehmen` zusammensetzt. Des Weiteren müssen die verschiedenen Rollen von Netzwerkunternehmen in der Netzwerkorganisation beachtet werden. Die Rollen im VDPN leiten sich dabei aus denen des VPN ab. Im Einzelnen finden sich im Netzwerkunternehmen eines Providers `VDPN Broker`, `VDPN Koordinator` und `VDPN Provider`. In diesem Zusammenhang ist eine dienstleistungsspezifische Erweiterung zu beachten. Da der `Client` des Dienstleistungsmodells im Sinne der Dienstleistungscharakteristik als externer Produktionsfaktor angesehen wird, muss er ebenfalls als Produktionseinheit im Netzwerk angesehen werden. Daher wird die Rolle des `VDPN Client` im Modell sowohl vom `Client` als auch vom `VPN Produzent` abgeleitet.

Auf konzeptioneller Ebene besteht nun das Ziel darin, die organisationsspezifischen Funktionen der VPN-Akteure im `VDPN`-Modell durch die Virtualisierung von Dienstleistungen zu untermauern. Hierzu ist zunächst der VDL-Begriff einzuführen, was in Abb. 4.7 geschieht. Der Begriff der virt. Dienstleistung wird hier ganz allgemein als Erweiterung des generellen Dienstleistungsbegriffs modelliert. Genauer erweitert eine `Virtuelle Dienstleistung` den Geschäftsprozess der `Dienstleistung` in Bezug auf die Abläufe in den Dienstleistungsphasen bei Substitution physikalischer Merkmale.[13] Entsprechend beinhalten `virtuelle Dienstleistungen` erweiterte Anwendungsfälle der einzelnen Dienstleistungsphasen. Für das konzeptionelle Modell wird primär der Fall betrachtet, in dem sich die eigentliche Substitution physikalischer Merkmale

[13]Dieser Umstand wird im Anwendungsfalldiagramm durch einen Erweiterungspunkt repräsentiert, der die Erweiterung selbst (`VDL`) und deren Ort (`DL-Phasen`) spezifiziert.

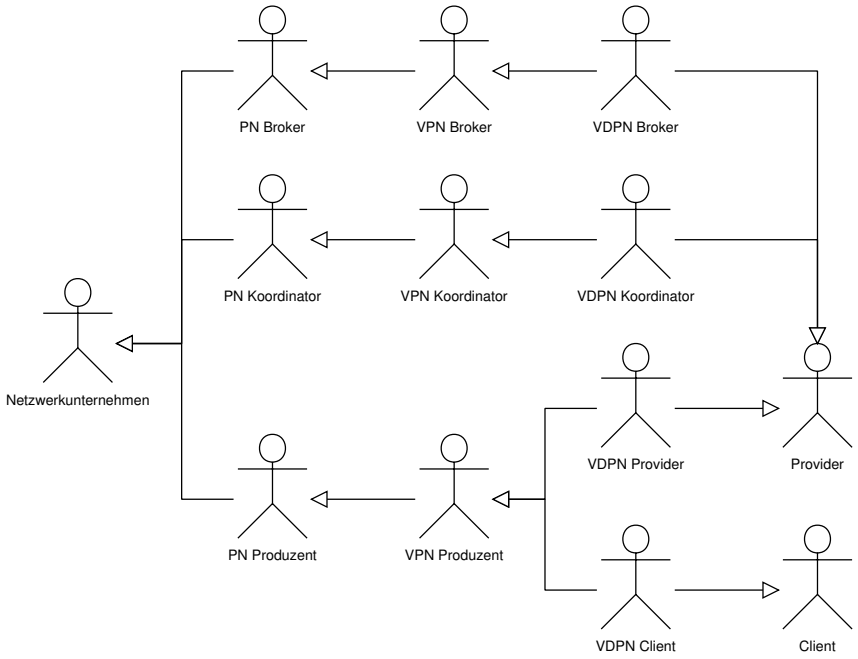

Abbildung 4.6.: Ableitung von Akteuren für ein VDPN

auf die Kontaktphase bezieht. Der hierin ablaufende Dienstleistungsprozess stellt ohne Zweifel den Kernaspekt des Dienstleistungsbegriffs dar, bei dem eine Virtualisierung die umfassendste Wirkung auf alle Akteure und deren Organisation zeigt. Dies gilt besonders für die Regelung der Koordination im VDPN durch Regelung der Interaktion zwischen Akteuren. Entsprechend beziehen sich die Erweiterungspunkte der Dienstleistungsphasen auf Zusatzfunktionen bei Planung, Durchführung und Verwertung in Bezug auf die Virtualisierung des DLP.

Als weiteren Aspekt zeigt Abb. 4.7 die primären Funktionen der `VDPN`-Akteure in Bezug auf die Phasen der virt. Dienstleistung. Der `Virt. Vorkontakt` umfasst im Sinne des Dienstleistungsmodells die Potenzialerstellung. Dies bezieht sich in diesem Fall auf den VDLP. Hierbei sind verschiedene Funktionen von `VDPN Broker`, `VDPN Provider` und `VDPN Client` beteiligt. `Broker` und `Provider` planen den VDLP simultan und koordiniert jeweils auf der Ebene des Netzwerks und der Knoten. Der `Client` sieht den geplanten VDLP ohne Einflussnahme lediglich ein und vergleicht ihn mit den Anforderungen seiner internen Prozesse. Während des `Virt. Kontakts` setzt der `VDPN Koordinator` die Vorgaben der VDLP-Planung praktisch durch. `Provider` und `Client` führen die für sie geplanten Aktivitäten in koordinierter Weise durch. Nach Beendigung des VDLP wird dessen Ablauf im `Virt. Nachkontakt` durch `Client` und `Broker` resümiert. Dazu gehört die Auswertung abgelaufener VDLP und deren Optimierung für die Zukunft. Nun kann der Begriff der virt. Dienstleistung für die vorliegende Arbeit wie folgt definiert werden:

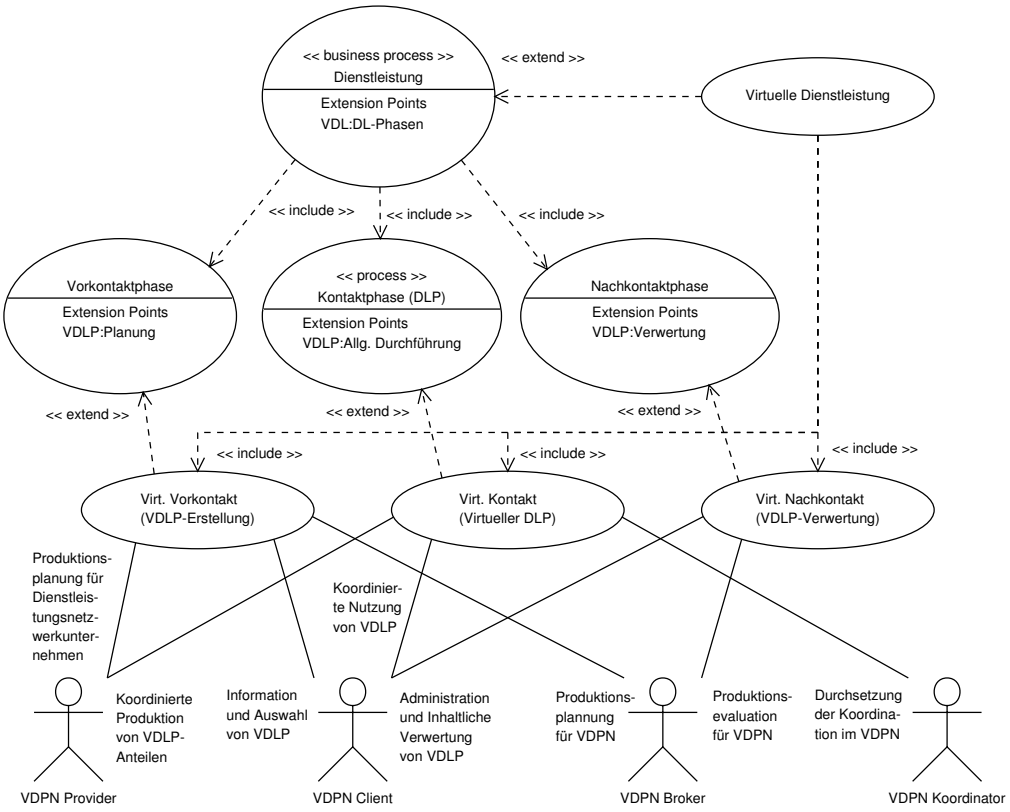

Abbildung 4.7.: Konzeptionelles Phasenmodell virt. Dienstleistungen

Definition 14 (virt. Dienstleistung) *Virtuelle Dienstleistungen sind Variationen von Dienstleistungen zur Produktion in VDPN. Hierbei werden Planung, Durchführung und Verwertung des Dienstleistungsprozesses durch IT gestützt und dienen zur ad hoc-Regelung und Implementierung der Koordination multipler Produktionseinheiten.*

Zur Virtualisierung des Dienstleistungsprozesses kommen nun verschiedene Möglichkeiten in Betracht. Diese setzen an den Bestandteilen an und heben deren „physikalische“ bzw. institutionale Eigenschaften auf. Auf Basis von IT-Mechanismen werden stattdessen zusätzliche Nutzeffekte erreicht. Dies kann für Interaktionsmuster der externen Steuer-, Hilfs- und Kernprozesse sowie für die Schnittstellen zu den darin enthaltenen Aktivitäten erfolgen, die als interne Prozesse realisiert werden. Das konzeptionelle VDLP-Modell wird in Abb. 4.8 gezeigt. Der `Virt. Kontakt` wird hier i.B. auf seine virtualisierten Anteile aufgespalten. Deren Struktur spiegelt das konzeptionelle DLP-Modell im linken Diagrammteil wider. Die Modellierung beinhaltet für jeden Anteil einen Erweiterungspunkt sowie erweiterten Anwendungsfall der virt. Variante. Hierin zeigt sich die Virtualisierungstrategie.[14]

[14]Siehe Diskussion in Sektion 4.2.1.2.

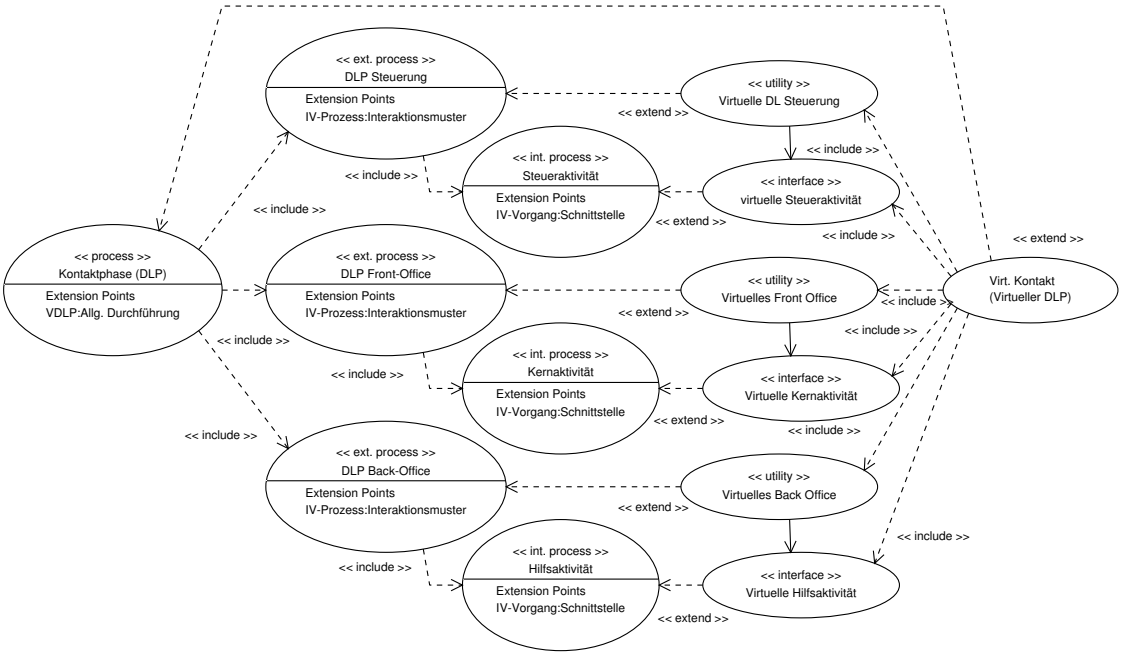

Abbildung 4.8.: Konzeptionelle Modellierung virt. Dienstleistungsprozesse

Bei externen Prozessanteilen strebt die Virtualisierung eine Substitution der darin auftretenden Interaktionsmuster durch IT-Repräsentationen an. Erweiterte Anwendungsfälle sind durch Werkzeuge gekennzeichnet, die als Nutzenwirkung eine systematische Entwicklung und Automation von Interaktionsprozessen erlauben.[15] Für die verschiedenen Vorgänge, die als interne Prozesse im Rahmen des DLP ablaufen, wird bei der Virtualisierung eine IT-Repräsentation der Prozessschnittstellen angestrebt. Die erweiterten Anwendungsfälle beziehen sich hier primär auf Nutzenwirkungen bei der exakten Spezifikation von und automatisierten Kommunikation mit Schnittstellen im Rahmen externer Prozessanteile.[16] Auf Basis dieser Modellierung kann der Begriff des virt. Dienstleistungsprozesses definiert werden:

Definition 15 (virt. Dienstleistungsprozess) *Der Begriff des virt. Dienstleistungsprozesses bezeichnet DLP, bei denen die interaktionsspezifischen Anteile auf integrativen IV-Techniken basieren. IV-gestützte Interaktionsprozesse repräsentieren Interaktionsmuster externer Prozesse als virt. Abläufe. IV-gestützte Kommunikationsendpunkte repräsentieren Schnittstellen interner Prozesse als virt. Vorgänge. Virt. Abläufe und Vorgänge können durch strukturierte Methoden spezifiziert, ausgeführt und ausgewertet werden.*

[15] Virt. Erweiterungen externer Prozessanteile sind mit dem Stereotyp ≪`utility`≫ versehen.
[16] Virt. Erweiterungen von Vorgängen sind mit dem Stereotyp ≪`interface`≫ versehen.

Der oben beschriebene Ansatz hat das Ziel, durch die Nutzenwirkungen der Virtualisierung eine Realisierung der virtuellen Organisationsstrukturen im VDPN zu ermöglichen. Die entsprechenden Zusammenhänge zwischen VDL und VDPN sind in Abb. 4.9 illustriert. Auf der Seite des VPN im rechten Teil der Abbildung werden hier noch einmal alle Aspekte des Produktionsmanagements explizit aufgezeigt. Hierbei ist die im VPN-Modell bisher nicht gezeigte Knotenebene ergänzt (`PP Knotenebene` und `PS Knotenebene`). Auf der linken Seite wird hingegen eine entsprechend vervollständigte Darstellung virt. Dienstleistungen gegenübergestellt. Insbesondere sind dabei die VDL-Phasen in verfeinerte Anwendungsfälle unterteilt, die Aktivitäten auf Netzwerk- und Knotenebene unterscheiden. Während sich dies für den VDLP auf natürliche Weise ergibt, werden für die Vor- und Nachkontaktphase entsprechende Anwendungsfälle ergänzend (`VDLP Erstellung` und `VDLP Verwertung` jeweils auf `Netzwerkebene` und `Knotenebene`).

Das Modell beschreibt die Verfeinerung der abstrakten Anwendungsfälle des VDPN-Produktionsmanagements durch die Anwendungsfälle virt. Dienstleistungen. Hierbei deckt sich die virtuelle Kontaktphase mit dem Bereich der Produktionssteuerung. Die Virtualisierung der externen Prozessanteile des DLP ermöglicht eine Steuerung der Produktion auf Netzwerkebene. Insbesondere wird durch IT-Repräsentation von Interaktionsmustern und IV-basierte Prozessunterstützung eine teil- oder vollautomatisierte Steuerung der Interaktionsprozesse zwischen `VDPN Providern` und `Clients` durch den `VDPN Koordinator` möglich. Die Virtualisierung interner Vorgänge im DLP ermöglicht hingegen die Produktionssteuerung auf Knotenebene. Diese erfolgt eigenverantwortlich durch `VDPN Provider`, `Client` und `Koordinator`. Durch die IT-Repräsentation der Vorgänge werden diese auf Netzwerkebene zur Kommunikation befähigt und lassen sich durch den `Koordinator` in Bezug auf ihre Interaktionen steuern.

Die verschiedenen Fälle des virt. `Vorkontakts` und `Nachkontakts` lassen sich dann als Verfeinerungen der VDPN-Produktionsplanung einsetzen. Dabei leisten die Anwendungsfälle der Vorkontaktphase mit Planung und Entwicklung des virt. Dienstleistungsprozesses inkl. seiner externen und internen Teilprozesse den wesentlichen Anteil der Produktionsplanung. Die Planung auf Netzwerkebene wird vom `VDPN Broker` durchgeführt. Die Virtualisierung der externen Prozessanteile erlaubt hier eine Spezifikation IV-basierter Beschreibungen von Interaktionsmustern. Diese können als Programme zur Regelung und Implementierung der Koordination dienen. Je nach den konkreten IV-Mechanismen, die zur Substitution von Interaktionsvorgängen zum Einsatz kommen, ist die Planung selbst ein strukturierter Prozess, der durch Pläne geregelt werden kann. In diesem Sinne können Pläne vorgegeben werden, die eine ad hoc-Anpassung von Programmen erlauben. Auf Knotenebene ermöglicht die Virtualisierung interner Vorgänge von `VDPN Providern` und `Clients` eine Koordination mit VDLP-Programmen. Vor diesem Hintergrund kann dann die Planung eigenverantwortlich erfolgen. Auf Netzwerkebene fließen die Ergebnisse als IV-basierte Schnittstellenspezifikationen der einzelnen internen Vorgänge ein und erlauben hier wiederum die Planung des ganzheitlichen VDLP. Die Anwendungsfälle der Nachkon-

Abbildung 4.9.: VDPN-Produktionsmanagement mittels VDLP

taktphase steuern in Bezug auf die Produktionsplanung vor allem die Evaluation virt. Dienstleistungsprozesse durch `VDPN Provider` und `Clients` bei. `Provider` können hieraus notwendige Änderungen am virt. Dienstleistungsprozess ableiten. `Clients` können den individuellen Nutzen des virt. Dienstleistungsprozess prüfen und das Ergebnis in zukünftige Planungen einbeziehen. Auf Basis dieser Modellierung kann der Begriff des virt. Dienstleistungsproduktionsnetzwerks für die vorliegende Arbeit definiert werden:

Definition 16 (virt. Dienstleistungsproduktionsnetzwerk) *Der Begriff des virt. Dienstleistungsproduktionsnetzwerk bezeichnet ein VPN, bei dem die interaktionsbasierte Koordination von Produktionseinheiten auf den Phasen einer virt. Dienstleistung basiert. Die Planung und Steuerung von Programmen basiert auf strukturierten*

Methoden zur Erstellung, Durchführung und Auswertung des VDLP. Strukturierte Vorgangsmodelle bilden Pläne zur agilen und flexiblen Programmerstellung.

Insgesamt skizziert das konzeptionelle Modell eine Strategie zur Umsetzung des VDPN-Produktionsmanagements auf Basis der Phasen virt. Dienstleistungen zur Erbringung virt. Dienstleistungsprozesse. Auf dieser Basis soll in den folgenden Sektionen der Entwurf eines spezifischen Dienstleistungsmodells beschrieben werden.

4.2.2. Das FRESCO-Dienstleistungsmodell

In den letzten Sektionen wurde die IV-Unterstützung der Dienstleistungsproduktion in virt. Dienstleistungsunternehmen auf Basis einer Virtualisierung von Dienstleistungen betrachtet. Dabei wurden verschiedene Varianten der Dienstleistungsvirtualisierung diskutiert und es wurde ein allgemeines konzeptionelles Basismodell für virt. Dienstleistungen in virt. Dienstleistungsproduktionsnetzwerken entworfen. In den folgenden Abschnitten soll beispielhaft ein konkreteres Modell virt. Dienstleistungen vorgestellt werden. Dieses verfeinert das generelle Modell virt. Dienstleistungen der vorangegangenen Sektion.

Das nun betrachtete VDL-Modell wurde im Rahmen des FRESCO-Projekts[17] entwickelt, dessen Ergebnisse in der vorliegenden Arbeit als exemplarische Lösungsmöglichkeiten der allgemein diskutierten Thematik herangezogen werden. Das in FRESCO entwickelte Dienstleistungsmodell demonstriert eine Möglichkeit zur konkreten Definition von Begriff und Mechanik virt. Dienstleistungen. Hierzu führt Sektion 4.2.2.1 den Dienstleistungsbegriff in FRESCO auf informelle Weise ein. In Sektion 4.2.2.2 folgt die Formulierung eines konzeptionellen Modells für FRESCO-VDL.

4.2.2.1. Einführung des FRESCO-Dienstleistungsbegriffs

In FRESCO wird in Bezug auf auf den Dienstleistungsbegriff grundsätzlich eine neutrale Perspektive angestrebt [PZL03]. Hiermit drückt sich aus, dass keine der im Kontext von Dienstleistungen beteiligten Rollen bevorzugt wird. Stattdessen werden im FRESCO-Ansatz die ganzheitlichen Vorgänge im Zuge der verschiedenen Phasen einer Dienstleistung in den Vordergrund gestellt.

Dementsprechend ist das FRESCO-Modell virt. Dienstleistungen (oder kurz FRESCO-Dienstleistungsmodell) auch als *dienstleistungszentriert* zu bezeichnen. Des Weiteren

[17] Das Forschungsprojekt *Foundational Research on Service Composition (FRESCO)* wurde in Zusammenarbeit zwischen der Universität Hamburg und den HP Laboratories Bristol zwischen 2002 und 2005 durchgeführt. FRESCO beschäftigte sich mit der Konzeption und Realisierung eines grundlegenden dienstleistungsorientierten Kooperationsansatzes [PZL03]. Ziel war die Entwicklung eines konzeptionellen und technischen Rahmenwerks, das Organisationen dabei unterstützt, Dienstleistungen als multilaterale Kooperationsbeziehungen zu realisieren. Im Zuge dessen wurde ein fundamentales Dienstleistungsmodell entwickelt, das Dienstleistungen als kooperative Interaktionsmuster beschreibt. Dieses Modell wurde im Rahmen eines generischen Dienstmanagementsystems auf Basis von Web- und Grid Service-Techniken implementiert.

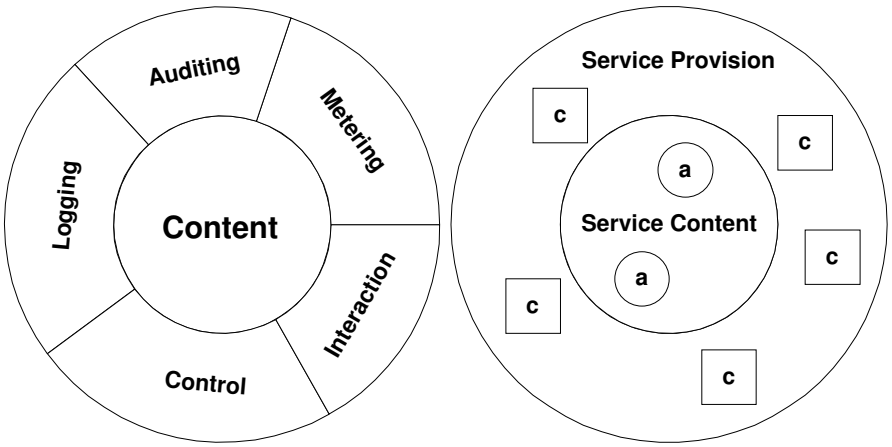

Abbildung 4.10.: Prinzip und Struktur des Dienstleistungsbegriffs in FRESCO

ist das Modell *erbringungsorientiert*. Der Fokus liegt hier nämlich im Wesentlichen auf der spezifischen Sicht des Dienstleistungsprozesses in der Kontaktphase. Die Vor- und Nachkontaktphasen des generellen Dienstleistungsbegriffs werden als sekundär betrachtet.

Der Grundgedanke des FRESCO-Dienstleistungsmodells besteht in der Einführung und Trennung der Aspekte von *Dienstleistungsinhalt* (*Service Content*) und *Dienstleistungserbringung* (*Service Provision*). Das Prinzip und die entsprechende Struktur von FRESCO-Dienstleistungen werden in Abb 4.10 dargestellt. Content erfasst die Kernleistungsprozesse[18] innerhalb des DLP, auf die sich die nutzenstiftende Wirkung der Dienstleistung gründet. Dies kann z. B. der Transport von Gütern durch Frachtführer sein. Einzelne solche Anteile werden als *Wert* oder *Asset* (`a`) bezeichnet. Unter Provision werden hingegen Administrationsprozesse zusammengefasst, die die Nutzenwirkung dem Kunden der Dienstleistung zuführen. Hierzu zählen z. B. die Steuerung und Protokollierung von Interaktionsprozessen. Ein solcher Anteil wird als *Fähigkeit* oder *Capability* (`c`) der Dienstleistung bezeichnet. Aus der Perspektive der grundlegenden Akteure einer Dienstleistung markiert Provision den durch Kunden wahrgenommenen Dienstleistungsprozess. Content stellt hingegen die getrennte Realisierung von dessen Nutzenwirkung durch den Provider dar. Für die Virtualisierung des Dienstleistungsprozesses und die dadurch ermöglichte virtuelle Organisation der Dienstleistungsproduktion ergeben sich durch eine derartige Trennung u. a. folgende Vorteile:

- Im VDPN bezieht sich der Dienstleistungsinhalt im Wesentlichen auf interne Kernkompetenzen einzelner Provider. Dienstleistungserbringung ergibt sich

[18] Der Begriff des *Kernleistungsprozesses* bezeichnet den wertschöpfenden Anteil eines Geschäftsprozesses (vgl. Sektion 2.3.1.2) und ist nicht mit dem *Kernprozess* des DLP zu verwechseln.

hier durch die Steuerung von Interaktionen zwischen Providern und Clients. Die Abgrenzung von Content und Provision spiegelt daher die Organisationsstruktur des VDPN auf Knoten- und Netzwerkebene wider.

- Inhalt und Provision markieren DLP-Anteile mit unterschiedlichem Potenzial zur Virtualisierung. Die Realisierung von Inhalten stützt sich vielfach auf materielle Prozesse; die Provision besteht hingegen zu großen Teilen aus informationellen Prozessen. Provision ist dementsprechend ein weitgehend und durchgehend virtualisierbarer Aspekt.

- Die Trennung von Provision und Content erlaubt eine Kapselung von Dienstleistungsinhalten. Hierdurch wird in VDPN der autonome Status von Providern als Netzwerkunternehmen hervorgehoben. Diese können die Realisierung von Inhalten eigenverantwortlich regeln. Die Virtualisierung des Content-Aspekts erlaubt dessen flexible Eingliederung in die Netzwerkebene.

- Die Abgrenzung von Provision fasst die interaktiven Anteile des DLP für alle beteiligten Rollen einheitlich zusammen. Im VDPN wird dadurch der koordinative Aspekt der Netzwerkebene erfasst und kann einheitlich geplant und durchgeführt werden. Die Virtualisierung der Dienstleistungserbringung erlaubt dabei eine flexible und agile Regelung der Interaktionen zwischen VDPN Akteuren.

Die Umsetzung des generellen Prinzips zeigt sich anhand der detaillierteren Strukturen der Content- und Provision-Ebenen, die im Folgenden näher beschrieben werden.[19] Der Dienstleistungsinhalt basiert in FRESCO auf einer Menge von Assets. Diese Assets bilden den *Kern* (*Service Core*) einer FRESCO-Dienstleistung (Abb 4.11, links). Assets bilden grundsätzlich einen in sich geschlossenen Dienstleistungsvorgang ab. Dieser Vorgang kann eine komplexe Prozessstruktur aufweisen. Er wird jedoch per Definition intern und eigenverantwortlich durch einen Provider geregelt. Im Kontext des Dienstleistungsmodells wird nur das resultierende Verhalten des Providers im Rahmen des Dienstleistungsprozesses betrachtet. Assets repräsentieren somit inhaltliche Aspekte, die in Bezug auf ihre Erbringung wohldefiniert sind und in diesem Rahmen als abgeschlossene Einheiten ohne Wechselwirkungen in den Dienstleistungsprozess eingebunden werden können. Aus diesem Grund stehen Assets im FRESCO-Modell immer in Bezug zur Provision-Ebene. Hierbei sind sie in eine Capability der Dienstleistung zur Erbringung einer entsprechenden Nutzenwirkung für den Kunden eingebunden.

Die Verwendung von Dienstleistungsinhalten zum Nutzen des Kunden geschieht im FRESCO-Modell mithilfe von administrativen Vorgängen zur Dienstleistungserbringung. Wie auf der Content-Ebene werden auch bei der Dienstleistungserbringung inhaltlich geschlossene Teilvorgänge strukturell getrennt. Dies geschieht hier in Form

[19] Siehe hierzu auch [GPZ03].

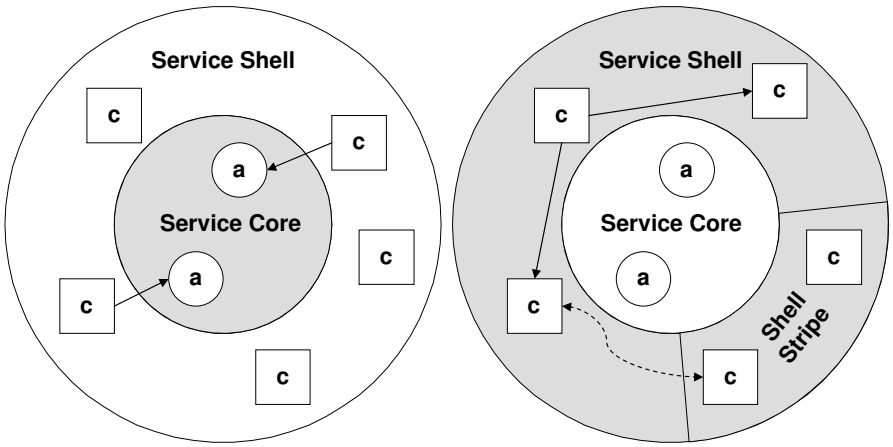

Abbildung 4.11.: Core- und Shell-Strukturen von FRESCO-VDL

von Capabilities. Capabilities repräsentieren Vorgänge, die in Kooperation mit Providern und Kunden stattfinden und diese in Verbindung setzen. In diesem Zuge greifen Capabilities administrativ in den Dienstleistungsprozess ein. Prinzipiell steht jedes Asset mit mindestens einer Capability in Beziehung, die die Interaktion mit dem Kunden beim Zugriff auf die Inhalte sowie weitere administrative Vorgänge regelt. Darüber hinaus existieren Capabilities, die auf einer übergeordneten Ebene den Ablauf des gesamten Dienstleistungsprozesses in Bezug auf die verschiedenen Assets regeln. Die Gesamtheit der Capabilities legt sich wie eine Hülle um den Content einer Dienstleistung und repräsentiert diesen nach außen. Diese Menge wird im FRESCO-Modell entsprechend als *Dienstleistungshülle* (*Service Shell*) bezeichnet (Abb. 4.11, rechts). Zur weiteren Differenzierung kann eine Einteilung von Capabilities in Teilmengen erfolgen, die *Sektionen der Hülle* (*Shell Stripes*) bilden.

Ungleich den Assets dienen die Capabilities der Shell nicht zur Kapselung von Vorgängen. Sie legen diese im Gegenteil offen. Dies geschieht mit der Intention, die Koordination von Kunden und Providern zu regeln. Um dies zu erreichen, wird das zu einer Capability gehörige Interaktionsmuster festgelegt. Dieses gibt allen Beteiligten das von ihnen verlangte Verhalten in Bezug auf die wechselseitige Interaktion vor. Darüber hinaus ermöglichen Interaktionsmuster die aktive Steuerung des Interaktionsvorgangs durch einen Koordinator. Ebenfalls ungleich den Assets stehen Capabilities in direkter wechselseitiger Verbindung zueinander. Auf diese Weise setzen sie die einzelnen Assets zueinander in Beziehung und bilden eine geschlossene Dienstleistungshülle, die den ganzheitlichen Dienstleistungsprozess repräsentiert. Als einfache Form zur Verbindung von Capabilities ist im FRESCO-Modell eine hierarchische Komposition vorgesehen.[20] Diese Art der Komposition ermöglicht die funktionale Dekomposition von Capability-Interaktionsmustern und hält durch

[20]Die hierarchische Komposition wird in der Abb. rechts durch gerade Pfeile angedeutet.

synchrone und sequenzielle Kopplung die Komplexität der Abläufe niedrig. Um komplexere Situationen des DLP-Ablaufs realisieren zu können, ist daneben eine P2P-Kopplung von Prozessen vorgesehen.[21] Diese Art der Kopplung erlaubt es, Capabilities an beliebigen Punkten ihrer Interaktionsmuster zu verknüpfen. Dadurch können asynchrone und parallele Interaktionsabläufe realisiert werden, die eine hohe Komplexität aufweisen können.

4.2.2.2. Konzeptionelle Modellierung von FRESCO-VDL

Der FRESCO-Dienstleistungsbegriff bildet auf Basis seines Grundprinzips der Trennung von Content und Provision eine vorteilhafte Basis, um durch seine Virtualisierung das Produktionsmanagement eines entsprechenden VDPN zu unterstützen. Die Strategie der Virtualisierung und deren Zusammenhang mit dem Produktionsmanagement wird im Folgenden durch ein konzeptionelles Modell von FRESCO-VDL beschrieben. Dieses Modell stellt eine Spezialisierung des in Sektion 4.2.1.3 eingeführten Basismodells dar.

Für das konzeptionelle Modell von FRESCO-VDL soll zunächst auf einige schon modellierte Basiskonzepte zurückgegriffen werden. Dazu gehören ganz fundamental die Modelle von Dienstleistungen, Dienstleistungsprozessen, virt. Produktionsnetzwerken und virt. Dienstleistungsproduktionsnetzwerken. Des Weiteren werden auch einige der Grundstrukturen des VDL-Modells als Grundlage verwendet. Dies resultiert daraus, dass der FRESCO-Dienstleistungsbegriff sich vor allem auf den Dienstleistungsprozess bezieht. Dieser ist jedoch nur ein (wesentlicher) Teilaspekt ganzheitlicher virt. Dienstleistungen im Sinne des Basismodells. Aus diesem Grund wird die fundamentale Struktur virt. Dienstleistungen mitsamt der Grundfunktionen relevanter Akteure direkt übernommen.[22] Ab hier muss die spezifische Charakteristik des FRESCO-Dienstleistungsbegriffs auf die Modellebene übertragen werden.

Struktur des FRESCO-VDLP Der Dienstleistungsbegriff in FRESCO ist in erster Linie durch seine alternative Sichtweise auf die Struktur des Dienstleistungsprozesses geprägt. Statt der Differenzierung in Kern-, Hilfs- und Steuerprozesse erfolgt hier eine Unterscheidung von Inhalt und Erbringung einer Dienstleistung. Die entsprechende Verfeinerung des DLP-Modells wird in Abb. 4.12 als Anwendungsfalldiagramm gezeigt. Im linken Teil des Diagramms sind die Basiskonzepte des allg. DLP-Modells dargestellt. Im zentralen Teil des Diagramms sind daneben die FRESCO-Konzepte als Anwendungsfälle zu sehen.[23] Diese stellen Spezialisierungen der Anteile des DLP-Basismodells dar.

[21] Die P2P-Kopplung wird in der Abb. rechts durch geschwungene Pfeile angedeutet.

[22] Siehe Diagramm in Abb. 4.7, S. 226.

[23] Verschiedene Elemente, die das spezifische FRESCO-Modell bilden, sind in den folgenden Diagrammen mit dem Präfix *FSM* gekennzeichnet. Dieses Akronym kann in diesem Kontext einfach als *FRESCO Service-Modell* gelesen werden.

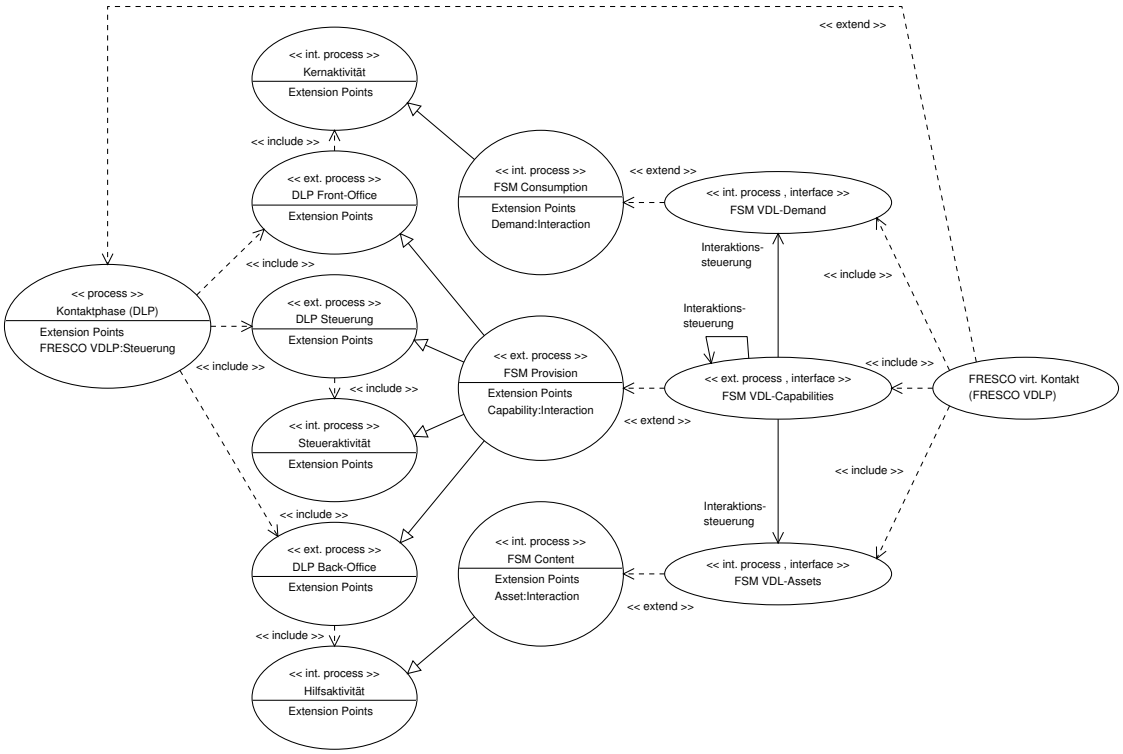

Abbildung 4.12.: Struktur des FRESCO-VDLP

`FSM Content` stellt eine Spezialisierung von `Hilfsaktivitäten` dar, die in Umfang und Bedeutung erweitert werden. Das Konzept repräsentiert nämlich Provider-interne Produktionsprozesse, die jeweils einen in sich geschlossenen Dienstleistungsinhalt darstellen. `FSM Provision` spezialisiert die vereinigten Eigenschaften von `Steueraktivitäten` sowie `Front Office`, `Back Office` und `Steuerungsprozess`. Aus abstrakter Sicht spezialisiert der kombinierte Anwendungsfall Steueraktivitäten als externe Prozesse, die jeweils Teile der Ablaufstruktur des DLP koordinieren. Dadurch werden funktional abgeschlossene administrative Vorgänge erfasst, die zur Vermittlung von Inhalten dienen. Dann wird noch ein verfeinertes Konzept ergänzt, das bisher noch nicht zur Sprache gekommen ist. Dabei handelt es sich um die verbleibenden Kernaktivitäten des Basismodells. Diese werden als `FSM Consumption` erfasst. Entsprechend dem `Content` repräsentiert `Consumption` interne Vorgänge des Konsums durch `Clients`. Diese Vorgänge stellen abgeschlossene kooperative Vorgänge von Kunden zur Nutzung von Dienstleistungsinhalten dar. `FSM Consumption` setzt so das ökonomische Konzept des externen Produktionsfaktors im konzeptionellen Modell um.

Die Virtualisierung des DLP basiert auf einer Virtualisierung der FRESCO-spezifischen Strukturelemente. Dies wird im konzeptionellen Modell durch Erweiterung der entsprechenden Anwendungsfälle dargestellt. Die Orte der diesbezüglichen Er-

weiterungspunkte deuten dabei die Art der Virtualisierung an.[24] Dies ist in allen Fällen der Interaktionsaspekt des jeweiligen DLP-Anteils. Im Falle des `Content` wird durch `Assets` der Interaktionsaspekt bei der Übergabe inhaltlicher Ergebnisse informationstechnisch substituiert. Dies geschieht im Wesentlichen durch IV-Modelle und Kommunikationsschnittstellen. Bei der `Provision` wird durch `Capabilities` die Interaktion im Rahmen von Verwaltungsvorgängen ersetzt. Das Mittel dazu bilden IV-Modelle, Steuermechanismen und Kommunikationsschnittstellen. Die Steuerfunktion von `Capabilities` in Bezug auf die anderen Anteile des DLP wird durch entsprechende Assoziationen gekennzeichnet. Diese führen von den `Capabilities` hin zu den von ihnen gesteuerten Anwendungsfällen. Eine rekursive Assoziation modelliert dabei den wichtigen Aspekt der Komposition verschiedener `Capabilities` zu einer integrierten Hülle. Schließlich erfolgt noch eine Auswechslung der Mittel zur Interaktion mit dem Kunden bei der Kooperation im Dienstleistungsprozess. Hierbei werden `Consumptions` durch den virtualisierten Anwendungsfall der `Demands` erweitert. Die Substitution basiert im Wesentlichen auf IV-Modellen und Kommunikationsschnittstellen.

`Assets`, `Capabilities` und `Demands` sind die Teile, aus denen sich in FRESCO der `Virt. Kontakt` bzw. der virt. Dienstleistungsprozess zusammensetzt. Der `FRESCO Virt. Kontakt` bildet dabei eine spezifische Erweiterung der allg. `Kontaktphase`. Der Ort der Erweiterung ist dabei speziell die Steuerfunktion.

Produktionssteuerung für FRESCO-VDLP Nach Aufstellung des verfeinerten VDLP-Modells stellt sich die Frage, wie dieses sich auf das Produktionsmanagement im virt. Dienstleistungsproduktionsnetzwerk anwenden lässt. Abb. 4.13 zeigt die Modellierung dieses Zusammenhangs zunächst für den speziellen Aspekt der Produktionssteuerung.

Ein erster Aspekt ist hier die Anpassung von Aufgaben der Akteure. Das Diagramm zeigt im unteren Bereich die Modellierung von FRESCO-Akteuren. Diese ergeben sich direkt aus den entsprechenden Akteuren des VDPN. Im Unterschied zu Letzteren ändert sich jedoch die Zuständigkeit der Akteure in Bezug auf den Aspekt der Koordination. Der `VDPN Koordinator` repräsentiert im VDPN-Modell eine übergeordnete Zuständigkeit für die aktive Steuerung von Abläufen im Zuge der Dienstleistungsproduktion durch verschiedene `VDPN Provider` und den `VDPN Client`. Diese Zuständigkeit wird im FRESCO-Modell differenziert. Grundlage hierfür ist die Struktur der Dienstleistungshülle. Diese setzt sich aus verschiedenen zusammenhängenden Verwaltungsvorgängen zusammen. Dabei kann zwischen solchen unterschieden werden, die sich auf einen spezifischen Dienstleistungsinhalt beziehen und solchen, die die verschiedenen Dienstleistungsinhalte zu einer ganzheitlichen Dienstleistung verknüpfen. Im FRESCO-Modell wird die Steuerung dieser Aspekte auf die einzelnen `Netzwerkunternehmen` verteilt. Generell verfeinern `FRESCO Provider` und `FRESCO Broker` beide die Steuerfunktion des abstrakten `FRESCO Koordinators`. Diese allg. Steuerfunk-

[24] Der Ort, an dem Erweiterungspunkte ansetzen, ist in UML als rechter Teil ihrer Signatur gegeben.

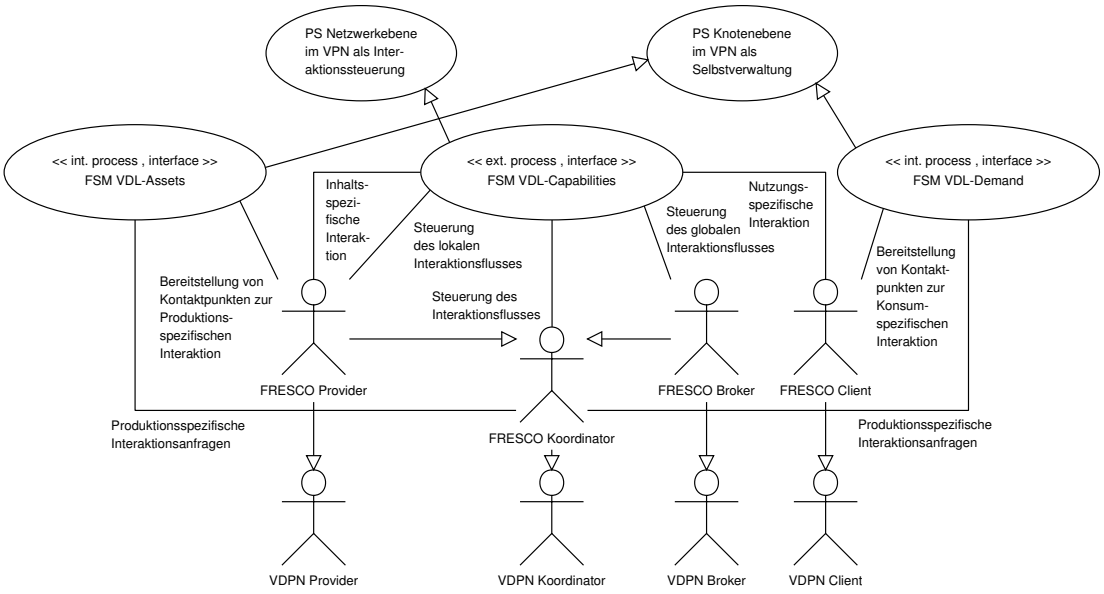

Abbildung 4.13.: VDPN-Produktionssteuerung in FRESCO

tion bezieht sich in FRESCO explizit auf den Ablauf von Interaktionen zwischen Akteuren. `FRESCO Provider` sind für die Steuerung des lokalen Interaktionsflusses zuständig, d. h. sie steuern die Interaktionsvorgänge, die sich bei der Administration der von ihnen eingebrachten Dienstleistungsinhalte ergeben. `FRESCO Brokern` obliegt hingegen die Steuerung des globalen Interaktionsflusses.

Daneben sind sowohl `FRESCO Provider` als auch `FRESCO Client` für die Steuerung ihrer internen Geschäftsprozesse verantwortlich. Im Rahmen der Produktionssteuerung für VDLP müssen sie Schnittstellen zu diesen internen Vorgängen einbringen. Diese Schnittstellen bilden lokale Kontaktpunkte. Mit ihnen beteiligen sich die Akteure an der übergeordneten Interaktion im Dienstleistungsprozess.

Die Aufteilung der Aufgaben von FRESCO-Akteuren innerhalb der einzelnen Anwendungsfälle des virt. Dienstleistungsprozesses lässt auf deren Bezug zu den Managementaufgaben allg. virt. Produktionsnetzwerke schließen. Hierbei ermöglichen `Capabilities` die Produktionssteuerung auf Netzwerkebene. Diese repräsentieren die virtualisierten Interaktionsabläufe der an der Dienstleistungsproduktion beteiligten Akteure. Der Einsatz von IT-Mechanismen ermöglicht deren effektive Steuerung. Die Produktionssteuerung auf Knotenebene wird hingegen in erster Linie durch `Assets` ermöglicht. Dazu kommen `Demands`, die den Aspekt des externen Produktionsfaktors einbeziehen. `Assets` und `Demands` repräsentieren jeweils interne Geschäftsprozesse der an der Dienstleistungsproduktion beteiligten Akteure, die im Rahmen einer Selbstverwaltung eigenverantwortlich durchgeführt werden. Der Einsatz von IT-Mechanismen ermöglicht gleichzeitig deren Kapselung und den effektiven Zugriff durch übergeordnete Steuerfunktionen der Netzwerkebene.

Produktionsplanung für FRESCO-VDLP Nach Behandlung der Produktionssteuerung verbleibt für die Gestaltung eines VDPN-Produktionsmanagements auf Basis von FRESCO-VDLP noch die Modellierung der Produktionsplanung. Wie schon im Basismodell dargestellt, deckt sich der Management-Aspekt der Produktionsplanung im virt. Produktionsnetzwerk mit den Phasen des Vor- und Nachkontakts von VDL.[25] Der Vorkontakt beinhaltet die Erstellung virt. Dienstleistungsprozesse, der Nachkontakt deren Ver- bzw. Auswertung. Hier ist es sinnvoll, Netzwerk- und Knotenebene zu differenzieren. Eine weiterführende Differenzierung dieser Anwendungsfälle wird in Abb. 4.14 modelliert.

Aus allg. Perspektive eines VDPN werden vier Anwendungsfälle unterschieden, die die wesentlichen Bestandteile der Erstellung und Verwertung von virt. Dienstleistungen bilden. Die `VDL-Erstellung` beinhaltet auf der Knotenebene zunächst die `Planung und Entwicklung von Hilfsaktivitäten`, die in Eigenverantwortung der `VDPN Provider` durchgeführt wird. Im FRESCO-Modell überträgt sich diese Aufgabe auf `FRESCO Provider`. Sie wird hier jedoch noch weiter unterteilt. Zum einen obliegt dem `FRESCO Provider` die `Entwicklung von Assets`. Dies ist aus der Perspektive der ganzheitlichen virt. Dienstleistung weitgehend transparent. Zum anderen muss der `FRESCO Provider` auch die `Entwicklung von Capabilities für Assets` leisten, da hier `Assets` und `Capabilities` wechselseitig aufeinander abzustimmen sind. Die FRESCO-spezifischen Anwendungsfälle stellen Erweiterungen des allg. VDPN-Entwicklungsvorgangs dar. Diese Erweiterungen betreffen die allg. Planungsaspekte der internen Produktion und externen Interaktion durch FRESCO-spezifische Repräsentationen in Form von `Assets` bzw. `Capabilities`.

Ein weiterer Anwendungsfall, der sowohl bei der `VDL-Erstellung auf Knotenebene` als auch bei der `VDL-Erstellung auf Netzwerkebene` benötigt wird, ist der `Abgleich von VDL Programmen`. Der Hintergrund ist hierbei die Abstimmung von verschiedenen Anteilen des ganzheitlichen virt. Dienstleistungsprozesses. Hieran sind verschiedene Akteure beteiligt. `VDPN Provider` und `VDPN Broker` stimmen jeweils die Interaktionen im Rahmen ihrer Hilfsaktivitäten und VDLP-Steuerprozesse miteinander ab. `VDPN Clients` stimmen die Schnittstellen ihrer internen Geschäftsprozesse mit den Interaktionen im Rahmen von VDLP-Kernprozessen ab. Insgesamt wird dadurch eine Koordination aller Akteure im DLP erreicht. All diese Aktivitäten der VDPN-Akteure übertragen sich prinzipiell auch auf die entsprechenden FRESCO-Akteure. Hierbei bezieht sich die Abstimmung jedoch in erster Linie auf `Capabilities`. `Sichtung und Abgleich von Capabilities` durch `FRESCO Provider`, `Broker` und `Clients` fokussieren die Abstimmung von Interaktionsmustern in sich geschlossener Verwaltungsvorgänge. Dies erweitert den Interaktionsaspekt des allg. Programmabgleichs im VDPN.

Bei der `VDL-Erstellung auf Netzwerkebene` besteht der primäre Anwendungsfall in der `Planung und Entwicklung der VDLP-Steuerung`. Hierbei entwirft der `VDPN Broker` den übergeordneten Ablauf des virt. Dienstleistungsprozesses. Diese Aufgabe findet

[25] Vgl. hierzu Abb. 4.9, S. 229.

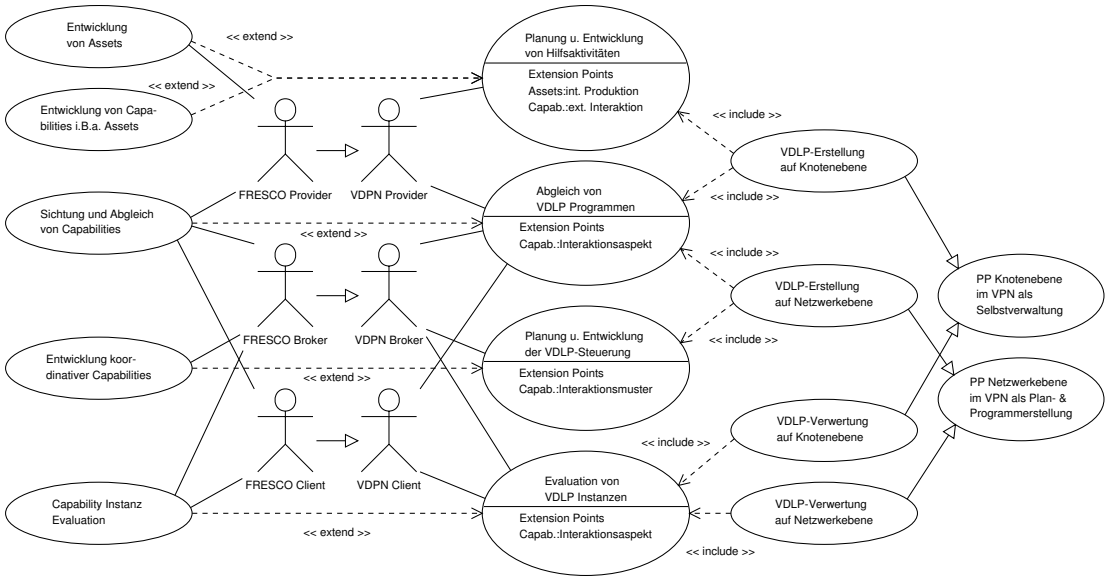

Abbildung 4.14.: VDPN-Produktionsplanung in FRESCO

sich entsprechend beim `FRESCO Broker` wieder. Diesem obliegt hier die `Entwicklung koordinativer Capabilities`. Der Anwendungsfall des FRESCO-Modells erweitert den allg. Fall im VDPN in Bezug auf den Interaktionsaspekt. Für Capabilities sind hier die Interaktionsmuster überschaubarer Verwaltungsvorgänge einzeln zu entwickeln. Dabei besteht eine wesentliche Aufgabe des `FRESCO Brokers` in der Zusammenführung schon vorhandener `Capabilities`, die von `FRESCO-Providern` in Bezug auf deren Assets vorgegeben werden. Hierzu können die vorhandenen `Capabilities` im Sinne einer Kopplung modifiziert werden. Eine andere Möglichkeit besteht in der Entwicklung spezifischer `Capabilities`, die speziell der Komposition der vorgegebenen `Capabilities` dienen.

Der letzte hier betrachtete Anwendungsfall beschreibt einen Vorgang im Rahmen der VDLP-Verwertung, der sowohl auf Netzwerk als auch auf Knotenebene einfließt. Hierbei handelt es sich um die Auswertung von Dienstleistungsprozessen nach deren Durchführung bzw. um `Evaluation von VDLP Instanzen`. Dieser Vorgang ist sowohl für `VDPN Broker` als auch `VDPN Clients` von Bedeutung. Broker prüfen bei der Evaluation die Korrektheit und Effizienz der im Netzwerk erbrachten Dienstleistungen. Sie identifizieren dabei ggf. Schwachstellen der Planung und leiten eine Änderung für die zukünftige Erbringung von Dienstleistungen ein. Kunden evaluieren ebenfalls die Korrektheit der von ihnen erhaltenen Dienstleistungen. Des Weiteren stellt das Nachvollziehen des Dienstleistungsprozesses ggf. einen Teil des Nutzeneffektes dar. Diese Funktionen übertragen sich prinzipiell auch auf den `FRESCO Broker` und `Client`. Die Auswertung betrifft hier speziell den Ablauf von Interaktionen im Rahmen einzelner Verwaltungsvorgänge. Hierbei findet durch die `Capability Instanz Evalua-`

tion insbesondere eine Erweiterung des allg. Anwendungsfalls in Bezug auf den Interaktionsaspekt statt.

Insgesamt lässt sich festhalten, dass sich durch FRESCO-VDLP ein kompletter Ansatz zur VDPN-Produktionsplanung und -steuerung ergibt. Dabei beziehen FRESCO-VDLP alle abstrakten Aspekte des VDPN-Produktionsmanagements in eine konkrete Virtualisierungsstrategie ein. Die Umsetzung dieser Virtualisierungsstrategie soll auf Basis von Techniken des Service Oriented Computing erfolgen und wird in den folgenden Abschnitten detailliert dargestellt.

4.3. Service-orient. Entwurf für virt. Dienstleistungen

In den vorangegangenen Abschnitten wurde die konzeptionelle Modellierung virt. Dienstleistungen diskutiert und am Beispiel des FRESCO-Ansatzes ausgeführt. Dies bildet den ersten Teil eines konzeptionellen Rahmenwerks zur Unterstützung der Dienstleistungsproduktion in virt. Dienstleistungsunternehmen. Das konzeptionelle VDL-Modell beschreibt dabei einen Ansatz zur IV-basierten Koordination kooperativer Tätigkeiten bei der Dienstleistungsproduktion in VDPN. Er basiert auf dem organisatorischen Prinzip, die Koordination kooperativer Tätigkeiten im VU auf die Regelung der Interaktion beteiligter Akteure zu übertragen. In diesem Sinne wird die Koordination kooperativer Tätigkeiten im VDPN auf die Regelung der Interaktion im Dienstleistungsprozess übertragen. Dabei werden die interaktiven Anteile des DLP in ihrer physikalischen bzw. institutionalisierten Form durch IV-Mechanismen ersetzt. Virtualisierung des DLP ermöglicht wiederum IV-basierte Methoden zu deren agiler und flexibler Erstellung, Durchführung und Verwertung in den verschiedenen VDL-Phasen. Diese Methoden bilden die Grundlage für Funktionen des Produktionsmanagements, die einer Regelung der Interaktionen und damit der Koordination kooperativer Tätigkeiten im VDPN dienen.

Bisher wurden die IV-Mechanismen zur Ersetzung von DLP-Anteilen und die darauf aufbauenden VDL-Methoden nur als grobe Kategorien betrachtet. Dies war für die Diskussion auf organisatorischer Ebene ausreichend. Hier lag der Fokus auf Nutzenwirkungen der Virtualisierung in Hinblick auf deren Verwertung im Rahmen des Produktionsmanagements. Diese Betrachtung liefert die Anforderungen für eine Umsetzung von VDLs durch konkrete IT. Das Ziel besteht dabei in der Konzeption übergreifender Integrationstechniken als Bestandteile der IV-Infrastruktur von VDPN. Für die informationstechnische Abbildung von VDPN-Interaktionen existieren hierfür generische Interoperabilitäts- und Koordinationsmechanismen der allg. horizontalen VU-Basisinfrastruktur. Diese können einbezogen und in Hinblick auf die Strukturen von Interaktionsvorgängen und –abläufen im Dienstleistungsprozess erweitert werden. Zusammen mit aufbauenden Methoden zur Erstellung, Durchführung und Verwertung von VDPN-Interaktionen bilden sie einen eigenständigen Koordinationsmechanismus als Teil einer spezifischen VDPN-Infrastruktur.

In den folgenden Sektionen wird zunächst die generelle Konzeptionierung einer kompletten VDL-basierten Integrationstechnik beschrieben. Dies beinhaltet (a) Konzepte zur technischen Repräsentation von VDPN-Interaktionen und (b) Konzepte für Methoden zu deren Management im VDPN. Auf dieser Basis wird dann die Realisierung der Repräsentation von VDPN-Interaktionen durch SOC-Techniken untersucht. Dies bildet den zweiten Teil des konzeptionellen Rahmenwerks zur Unterstützung der Dienstleistungsproduktion in virt. Dienstleistungsunternehmen. Die anschließende Sektion 4.4 knüpft dann in Bezug auf die Realisierung von Methoden und Vorgehensmodellen zum Management SOC-basierter VDPN-Interaktionen an.

Am Anfang der Konzeptionierung einer VDL-basierten Integrationstechnik steht eine generelle Analyse der Technikfamilie in Bezug auf Anforderungen und Formen der Realisierung. Dies erfolgt in Sektion 4.3.1. Zunächst erfolgt hier eine generische Anforderungsanalyse auf Basis der konzeptionellen Grundlagen der vorangegangenen Abschnitte. Dazu werden noch einmal die Anforderungen des Koordinationskonzepts herausgestellt und mit den Anforderungen der Virtualisierungsstrategie zusammengeführt. Auf Basis der Anforderungen wird dann die Realisierung virt. Dienstleistungen als Integrationstechniken diskutiert. Dabei wird zunächst die Klasse von Integrationstechniken zur VDL-Realisierung abgegrenzt und als E-Services definiert. E-Services werden dann in den Rahmen der IV-Infrastruktur von VDU eingeordnet und mit anderen Integrationstechniken in Beziehung gesetzt. Im Sinne des spezifischen Fokus der vorliegenden Arbeit erfolgt dann eine Untersuchung von SOC-Techniken auf ihre Potenziale zur Realisierung von E-Services. Dies wird durch die Charakteristik der Anforderungen motiviert. Hierbei ist der Interaktionsprozess autonomer Akteure maßgeblich. SOC-Techniken bieten durch ihre Potenziale zur Prozessintegration einen offensichtlichen Ansatzpunkt.

Im weiteren Verlauf wird dann in Sektion 4.3.2 ein konkreter Ansatz zur Realisierung von E-Service auf Basis von SOC-Techniken vorgestellt. Der Ansatz basiert auf dem FRESCO-Dienstleistungsbegriff und dem dafür in den vorangegangenen Abschnitten entwickelten konzeptionellen VDL-Modell. Zunächst erfolgt hier eine Gegenüberstellung des VDL-Modells mit dem Service-orient. Modell auf Basis der vorangehenden Untersuchungen von SOC-Potenzialen bei der allg. DLP-Virtualisierung. Dabei wird ein erweitertes FRESCO-SOM für *E-Services* als *Service-orient. virt. Dienstleistungen* entwickelt. Das FRESCO-SOM bildet ein abstraktes Schema für den Entwurf von E-Services als integrativen Interaktionsprozessen im Rahmen kooperativer Service-orient. Anwendungssysteme zur Produktionssteuerung in VDPN. Zur Spezifikation der konkreten Service-orient. Architektur von E-Services wird dann ein Metamodell erarbeitet. Dieses *FRESCO E-Service-Metamodell (FSM)* gibt vor, wie E-Services auf Basis Service-orient. Prinzipien zu entwerfen sind. Es ist dabei grundsätzlich generisch ausgelegt. Der Entwurf erfolgt daher auf Basis eines allgemeinen Prozessmodells. E-Service-Entwürfe lassen sich so auf verschiedene konkrete Techniken aktueller und zukünftiger WS-Standards abbilden.

4.3.1. Konzeption VDL-basierter Integrationstechniken

Im konzeptionellen VDL-Modell wurden die Möglichkeiten der Dienstleistungsvirtualisierung herausgestellt und zur Koordination von Akteuren eines virt. Dienstleistungsproduktionsnetzwerks angewendet. Die Möglichkeiten der Dienstleistungsvirtualisierung beziehen sich auf den Einsatz (integrativer) IT zur Ersetzung verschiedener Aspekte der Interaktion zwischen Netzwerkunternehmen und Kunden im Dienstleistungsprozess. Die Anwendung dieser Techniken im Rahmen des Produktionsmanagements im VDPN erfordert weitere IV-Mechanismen zur Steuerung des Lebenszyklus virtualisierter Interaktionsprozesse. Diese Mechanismen entsprechen den Phasen konventioneller Dienstleistungen und setzen diese in Hinblick auf virt. Dienstleistungsprozesse um. Insgesamt bildet das VDL-Modell den konzeptionellen Hintergrund für eine umfassende IV-Technik zur Integration der IV im VDPN.

Um diese Integrationstechnik zu realisieren, muss aus dem organisatorischen Konzept ein technisches Konzept abgeleitet werden. Hierbei sind die im VDL-Modell beschriebenen technischen Komponenten zu konkretisieren und in Hinblick auf Realisierungsmöglichkeiten zu untersuchen. Um dies zu erreichen, wird das VDL-Modell in Sektion 4.3.1.1 zunächst in Bezug auf seine Anforderungen an die Funktionen und Eigenschaften der technischen Komponenten analysiert. Hierbei soll zum einen die interaktionsbasierte Regelung und Implementierung der Koordination autonomer Akteure im virt. Produktionsnetzwerk unterstützt werden. Zum anderen fließt die Interaktivität und Prozesscharakteristik von Dienstleistungen ein. Als Resultat beziehen sich wesentliche Anteile der Anforderungen auf das Prozessmanagement und die zwischenbetriebliche Interaktion.

In Sektion 4.3.1.2 werden auf Basis der Anforderungen dann VDL-basierte Integrationstechniken abgeleitet. Dabei wird für Techniken zur Repräsentation von VDLP-Interaktionen der Begriff des E-Services eingeführt. Techniken zur Realisierung aufbauender Methoden zur Unterstützung des VDPN-Produktionsmanagements werden entsprechend als E-Service-Management vorgestellt. Die VDL-basierten Integrationstechniken werden dann in die allg. IV-Infrastruktur virt. Dienstleistungsunternehmenen eingeordnet und dort mit verschiedenen grundlegenden Integrationstechniken in Beziehung gesetzt. Des Weiteren wird das Vorgehen bei der Realisierung von E-Service auf Grundlage verschiedener Basistechniken diskutiert. Dabei wird insbesondere die Nutzung von Basistechniken des Service Oriented Computing fokussiert. Hierbei wird zunächst das grundlegende Paradigma des Service Oriented Computing im Kontext von virt. Dienstleistungen evaluiert. Dann werden verschiedene Instrumente des einfachen und erweiterten Service-orient. Modells als Ansätze zur Realisierung von E-Services untersucht. Im Wesentlichen werden dabei zum einen die Instrumente der WS-Basisarchitektur zur Kapselung und Eingliederung autonomer Provider und Kunden in ein VDPN erwogen. Zum anderen werden die verschiedenen Ansätze zur SOC-basierten Prozessintegration als Möglichkeiten zur Regelung und Implementierung des globalen Interaktionsablaufs im VDPN betrachtet.

4.3.1.1. Anforderungsanalyse virt. Dienstleistungen

Die Analyse der Anforderungen an IV-Techniken zur Dienstleistungsvirtualisierung setzt am konzeptionellen Modell allg. virt. Dienstleistungen an. Das VDL-Modell beinhaltet eine Virtualisierungsstrategie, die den Einsatz verschiedener IV-Techniken zur Substitution bestimmter Eigenschaften von Dienstleistungen vorsieht. Hintergrund der Substitution ist die Ausstattung virt. Dienstleistungen mit zusätzlichen Eigenschaften und Fähigkeiten, die auf den verwendeten IV-Techniken basieren. Diese Nutzenwirkung der Dienstleistungsvirtualisierung zielt wiederum auf die Virtualisierung von Dienstleistungsnetzwerken ab. Genauer soll dadurch das Produktionsmanagement in virt. Dienstleistungsproduktionsnetzwerke ermöglicht werden. Dazu wird ein generischer Ansatz zur Koordination kooperativer Tätigkeiten in virt. Unternehmen verfolgt. Dieser beruht darauf, in geplanter Art und Weise Programme zu erstellen, die die Interaktion von VU-Akteuren festlegen. Diese Programme dienen zum einen zur Regelung der Koordination. Zum anderen erlauben sie die aktive Durchsetzung kooperativer Regelungen. Die Eigenschaften und Fähigkeiten virt. Dienstleistungen sollen in diesem Zusammenhang die strukturierte Planung und automatisierte Durchführung von Programmen ermöglichen. Dies soll auf formalen Modellen der Interaktionsmuster und Schnittstellen des Dienstleistungsprozesses basieren und durch IV-Mechanismen zu deren Entwicklung und Ausführung realisiert werden. Die Anforderungen an diese IV-Techniken setzen sich also aus vorgegebenen *organisatorischen Anforderungen* des Koordinationsansatzes und abgeleiteten *technischen Anforderungen* zur Dienstleistungsvirtualisierung zusammen.

Organisatorische Anforderungen Auf oberster Ebene der Anforderungen steht die Realisierung der virtuellen Organisationsform. Hierbei besteht ein wesentliches Teilproblem in der Koordination der autonomen Netzwerkunternehmen. Dazu gehört zum einen die *Regelung der Koordination*. Dies betrifft die Festlegung von Beziehungen und Abhängigkeiten der Netzwerkteilnehmer zur Kooperation im Netzwerk. Zum anderen muss eine Realisierung koordinativer Regelungen erfolgen. Dies teilt sich in Institutionalisierung und Implementierung der Koordination. Auf der Ebene des strategischen Netzwerks, aus dem einzelne VU hervorgehen, wird eine langfristige Institutionalisierung in ideeller und materieller Form vorgenommen; d. h. z. B. Abschluss von Rahmenverträgen und Aufbau einer Infrastruktur zwischen den Netzwerkunternehmen. Die vorliegende Arbeit kann in diesem Sinne als ideelle Institutionalisierung der Koordination in strategischen Dienstleistungsnetzen gesehen werden, die als Rahmen für temporäre VDPN dienen. Hierbei soll die Infrastruktur vorgegeben werden, die eine *Implementierung koordinativer Regelungen* für einzelne VDPN ermöglicht. Die Implementierung bezieht sich dabei auf die ad hoc-Realisierung einer Weisungsstruktur. Diese dient zur Durchsetzung koordinativer Regelungen für die Erbringung bzw. Produktion einzelner virt. Dienstleistungen durch ein spezielles virt. Dienstleistungsproduktionsnetzwerk.

Als allg. Mittel zur VU-Koordination wird der Ansatz von Specht und Kahmann fokussiert.[26] Er setzt bei der Interaktion der VU-Akteure an. In den Interaktionen der Akteure spiegeln sich deren kooperative Beziehungen wider. Die Regelung der Interaktionen kann daher zur Regelung der Koordination kooperativer Tätigkeiten herangezogen werden. Dies soll durch *Erstellung von Programmen*, die die Interaktionen der Akteure und deren Beziehungen (z. B. Präzedenzen) spezifizieren, geschehen. Um die Koordination im VU dabei ad hoc regeln zu können, ist im Vorfeld eine *Planung der Programmerstellung* durchzuführen.[27] Während nun die Erstellung von Programmen auf die Regelung der Koordination zielt, sollen die resultierenden Programme im Folgenden zur Implementierung dieser Regelungen eingesetzt werden. Die *Durchführung der Programme* soll dabei die Interaktion der Akteure steuern.

Der generelle Ansatz zur VU-Koordination dient als Anforderung für das Produktionsmanagement im spezifischen Fall des VDPN. Hier ergeben sich verschiedene koordinationsspezifische Funktionen, die für die Produktionsplanung und -steuerung auf Netzwerk- und Knotenebene anfallen. Auf Netzwerkebene muss eine *Planung von Interaktionsmustern* zwischen Produktionseinheiten erfolgen. Hierfür ist durch *Planung der Planung von Interaktionsmustern* eine geeignete Methodik festzulegen. Zur Produktionssteuerung ist eine *Steuerung von Interaktionen* zwischen den Produktionseinheiten im Sinne der geplanten Interaktionsmuster durchzuführen. Auf Knotenebene erfolgt die Planung der internen Produktionsprozesse autonom und wird daher aus der Perspektive der VDPN-Koordination nicht betrachtet. Es muss jedoch eine *Planung von Schnittstellen* erfolgen, über die eine Produktionseinheit mit anderen zwecks Kooperation Interaktionen durchführt. Dabei ist eine Abstimmung mit den Interaktionsmustern der Netzwerkebene durchzuführen. Für den Vorgang muss daher eine mit der Netzwerkebene abgestimmte Methodik wiederum durch *Planung der Planung von Schnittstellen* bestimmt werden. Bei der Produktionssteuerung auf Knotenebene bezieht sich die geforderte Funktionalität wiederum auf die Prozessschnittstelle. In diesem Sinne ist eine *Steuerung der Schnittstellenkommunikation* im Sinne der geplanten Interaktionen durchzuführen.

Damit sind die organisatorischen Anforderungen zur Regelung der Koordination kooperativer Tätigkeiten auf abstrakter Ebene und für den spezifischen Fall des VDPN festgelegt. Die Ergebnisse werden im linken Teil von Tabelle 4.1 zusammengefasst.

Technische Anforderungen Mit den Funktionen des VDPN-Produktionsmanagements sind die Anforderungen an virt. Dienstleistungen gegeben. Deren Funktionen und Eigenschaften basieren grundsätzlich auf der im konzeptionellen Modell beschriebenen Virtualisierungsstrategie. Diese Strategie gibt die Ersetzung von physikalischen bzw institutionellen Anteilen des Dienstleistungsprozesses durch IV-Techniken vor. Hierzu werden die verschiedenen Anteile des Dienstleistungsprozesses in zwei Klassen

[26] Siehe Sektion 2.3.3.2, S. 63.

[27] Die Erstellung von Plänen auf Basis von Meta-Plänen wird ausgeklammert.

Organisatorische Anforderungen				Technische Anforderungen			
Organisationskonzept		Produktionsmanagement		virt. Dienstleistungen		virt. Dienstleistungsprozesse	
Ziel	Aspekt	Funktion	Aufgabe	Phase	Methoden	Anteil	Substitute
Regelung der Koordination	Planung der Programmerstellung	Planung Netzebene	Planung der Planung von Interaktionsmustern zwischen den Produktionseinheiten	Meta	Integrierte Vorgehensmodelle zur Entwicklung IV-gestützter Interaktionsprozesse und Kommunikationsendpunkte im Rahmen verschiedener Funktionen und Ebenen des Produktionsmanagements	Virt. Ablauf	Metamodelle für Interaktionsprozesse als Basis strukturierter Entwicklungsmethoden
		Planung Knotenebene	Planung der Planung von Schnittstellen von Produktionseinheiten			Virt. Vorgang	Metamodelle für Kommunikationsendpunkte als Basis strukturierter Entwicklungsmethoden
	Erstellung von Programmen zur Interaktion zwischen Akteuren	Planung Netzebene	Planung von Interaktionsmustern zwischen Produktionseinheiten	Vor- und Nachkontakt	Formale Spezifikation und wechselseitiger Abgleich sowie Analyse und systematische Änderung IV-gestützter Interaktionsprozesse und Kommunikationsendpunkte	Virt. Ablauf	Modelle von Interaktionsprozessen zur Protokollspezifikation
		Planung Knotenebene	Planung von Schnittstellen der Produktionseinheiten			Virt. Vorgang	Modelle von Kommunikationsendpunkten zur Schnittstellenspezifikation
Implementierung der koordinativen Regelung	Durchsetzung von Programmen zur Interaktion zwischen Akteuren	Steuerung Netzebene	Steuerung von Interaktionen in Bezug auf Interaktionsmuster	Kontakt	Automatisierte Ausführung IV-gestützter Interaktionsprozesse und der dabei anfallenden Schnittstellenkommunikation	Virt. Ablauf	IV-Plattform (Programmiermodell und Laufzeitfunktionen) zur Ausführung von Interaktionsprozessen
		Steuerung Knotenebene	Steuerung interner Prozesse und Schnittstellenkommunikation			Virt. Vorgang	Middleware (Programmiermodell und Laufzeitfunktionen) zur Durchführung von Schnittstellenkommunikation

Tabelle 4.1.: Von organisatorischen zu technischen VDL-Anforderungen

eingeteilt. Back Office, Steuerung und Front Office bilden als externe Prozesse jeweils einen Teil der DLP-Ablaufstruktur. Kern-, Hilfs- und Steueraktivitäten bilden als interne Prozesse atomare Teilvorgänge. Bei den Abläufen besteht die Virtualisierung in einer Substitution derjenigen informationellen Administrationsprozesse, die der Interaktion zwischen beteiligten Akteuren dienen. Ein Beispiel hierfür sind die Abläufe, die von der Telefonistin einer Taxizentrale ausgeführt werden. Diese Abläufe werden durch *IV-basierte Repräsentationen von Interaktionsprozessen* ersetzt und als *virt. Abläufe* bezeichnet. Bei den Vorgängen besteht die Virtualisierung in der Substitution der Schnittstellen zu internen Administrationsprozessen. Als Beispiel kann der Vorgang dienen, bei dem ein Taxi per Funk gerufen wird. Die Schnittstelle ist hier der Taxifahrer am Funkgerät. Solche Schnittstellen werden durch *IV-basierte Repräsentationen von Kommunikationsendpunkten* ersetzt und als *virt. Vorgänge* bezeichnet.

Unter „Ersetzung durch IV-Repräsentation“ ist zu verstehen, dass Abläufe und Vorgänge nicht mehr in ihrer bisherigen Form erfolgen, sondern unter Einsatz von IV-Mechanismen. Dadurch werden verschiedene Nutzenwirkungen angestrebt. Durch

virt. Abläufe sollen die Interaktionsprozesse im DLP explizit entworfen und als Spezifikation erfasst werden. Entsprechend sollen durch virt. Vorgänge die Kommunikationsendpunkte autonom geplanter Vorgänge explizit spezifiziert und mit den Interaktionsprozessen abgestimmt werden. Diese Nutzenwirkung fließt als IV-basierte Methode zur Erstellung von virt. Dienstleistungsprozessen in die Vorkontaktphase ein. Des Weiteren sollen die Interaktionsprozesse des DLP durch virt. Abläufe automatisch ausgeführt werden. Durch virt. Vorgänge soll dabei eine automatisierte Kommunikation mit den Vorgängen von Kunden und Dienstleistungsunternehmen erfolgen. Diese Nutzenwirkung fließt als IV-basierte Methode zur Durchführung von virt. Dienstleistungsprozessen in die Kontaktphase ein. Schließlich sollen Interaktionsprozesse durch virt. Abläufe u. a. in Bezug auf Änderungen evaluiert werden. Parallel sollen Kommunikationsendpunkte durch virt. Vorgänge u. a. in Bezug auf Änderungen evaluiert werden. Diese Nutzenwirkung fließt als IV-basierte Methode zur Auswertung von virt. Dienstleistungsprozessen in die Nachkontaktphase ein.

Die grundsätzlichen Nutzenwirkungen virtueller Abläufe und Vorgänge bilden die Basis zur Erfüllung der organisatorischen Anforderungen. Dies erfolgt durch Anwendung auf das VDPN-Produktionsmanagement. Hierbei kommt die VDLP-Erstellung als Methode der Produktionsplanung zum Einsatz. Interaktionsprozesse des VDLP fungieren als Interaktionsmuster im VDPN und dienen dort als Programme. Im Zusammenhang mit den Kommunikationsendpunkten autonomer Vorgänge erlauben sie die Regelung der Koordination zwischen Produktionseinheiten und Kunden. Auch die VDLP-Auswertung kann als Methode der Produktionsplanung verwendet werden. Evaluation und Änderung von VDLP bilden dabei Methoden zur Verbesserung der Planungsqualität. Für den Gesamtprozess der Planung als solchen wird dann im VDPN-Management wiederum eine geplante Vorgehensweise gefordert. Daher muss für VDL eine Metaebene berücksichtigt werden, die die Möglichkeit zur Vorgabe formaler und problemorientierter Entwicklungsprozesse für VDLP bietet. Der Entwicklungsprozess soll dabei insbes. die agile Erstellung virt. Dienstleistungsprozesse unterstützen, um der Dynamik von VDPN gerecht zu werden. Die VDLP-Durchführung bildet dann die Grundlage der Produktionssteuerung. Die automatische Ausführung von VDLP entspricht der Steuerung von Interaktionen im VDPN nach Vorgabe von Interaktionsmustern. Damit sind die wesentlichen Aufgaben des VDPN-Produktionsmanagements durch IV-basierte Methoden der VDL-Phasen abgedeckt.

Die angestrebten Nutzenwirkungen bilden Anforderungen an die Eigenschaften IV-basierter Repräsentationen von Interaktionsprozessen und Kommunikationsendpunkten sowie an die darauf aufsetzenden Methoden und Mechanismen einer virt. Dienstleistung. Diese sollen nun im Gesamtzusammenhang verdeutlicht werden. Vor der Produktion von VDL in VDPN erfolgt als Erstes die Planung von Interaktionsmustern und Schnittstellen durch Netzwerkunternehmen in Broker- und Provider-Rolle. Dies wird durch VDLP-Erstellung realisiert und beinhaltet Entwurf, Spezifikation und Abgleich von Interaktionsprozessen zwischen Kommunikationsendpunkten. Die Spezifikation von Interaktionsprozessen erfordert grundsätzlich ein formales Prozess-

modell. Dabei müssen zumindest Rollen, Interaktionsvorgänge und Ablaufstruktur berücksichtigt werden. Die Spezifikation von Kommunikationsendpunkten erfordert daneben ein formales Kommunikationsmodell. Hierbei müssen zumindest atomare Kommunikationsvorgänge als Schnittstelle enthalten sein. Dafür sind verschiedene z. B. nachrichtenbasierte oder operationale Ansätze denkbar. Zur Regelung der Koordination zwischen Netzwerkunternehmen erfolgt eine wechselseitige Vorgabe und Abstimmung von Interaktionsprozessen und Kommunikationsendpunkten. In diesem Sinne müssen sich die Prozess- und Kommunikationsmodelle als Interaktionsprotokolle und Schnittstellenspezifikationen einsetzen lassen. Zu deren Formulierung werden im vorliegenden zwischenbetrieblichen Kontext standardisierte Prozessbeschreibungssprachen und IDLs benötigt.

In der operationalen Phase des VDPN erfolgt dann die Produktion von VDL. Die Produktionssteuerung wird hier durch Netzwerkunternehmen in Koordinator- und Provider-Rolle durchgeführt. Der Koordinator steuert hierbei die Interaktionen im VDPN im Sinne des Interaktionsmusters. Dies wird durch VDLP-Durchführung realisiert und beinhaltet Konstruktion und Ausführung von Interaktionsprozessen. Die Ausführung von Interaktionsprozessen erfordert grundsätzlich ein Programmiermodell, das durch die Laufzeitfunktionen einer Prozess-Plattform implementiert wird. Vor der Ausführung muss die Spezifikation des Interaktionsprozesses in eine ausführbare Repräsentation auf Basis des Programmiermodells überführt werden. Provider steuern ihre internen Prozesse sowie die Schnittstellenkommunikation. Die Steuerung interner Prozesse erfolgt in transparenter Weise. Die Steuerung von Schnittstellenkommunikation wird durch die Konstruktion von Kommunikationsendpunkten und die automatisierte Ausführung von Kommunikationsvorgängen erreicht. Die automatische Kommunikation über zwischenbetriebliche Schnittstellen erfordert grundsätzlich ein Programmiermodell, das durch die Laufzeitfunktionen einer Middleware implementiert wird. Vor der Ausführung von Kommunikationsvorgängen muss die Spezifikation des Kommunikationsendpunktes in eine Repräsentation auf Basis des Programmiermodells überführt werden. Die Laufzeitfunktionen müssen organisationsübergreifend ausgelegt sein.

Am Ende der Produktion steht die Evaluation des operativen Produktionsvorgangs durch ein Netzwerkunternehmen in Broker-Rolle und ggf. die Anpassung von Interaktionsmustern und Schnittstellen. Dies wird durch VDLP-Auswertung umgesetzt. Hierbei findet eine Evaluation der Ausführung von Interaktionsprozessen und Schnittstellenkommunikation statt. Bei Bedarf erfolgt eine Änderung der Spezifikationen. Hierzu sind formale Methoden entscheidend, um die Konsistenz von Spezifikationen zu bewahren. Auf Basis der Modelle von Interaktionsprozessen und Kommunikationsendpunkten und der darauf basierenden Mechanismen muss ein formaler Entwicklungsprozess unterstützt werden. Formale Entwicklungsprozesse können grundsätzlich durch Berücksichtigung einer Metaebene für Modelle profitieren. Auf Basis von Metamodellen können formale Entwicklungsprozesse und -methoden exakt definiert werden.

Damit sind die technischen Anforderungen zur Virtualisierung von Dienstleistungen auf Basis von deren Anwendung zur Regelung der Koordination kooperativer Tätigkeiten für den spezifischen Fall des VDPN festgelegt. Die Ergebnisse werden im rechten Teil von Tabelle 4.1 zusammengefasst.

4.3.1.2. Realisierung virt. Dienstleistungen als E-Services

Nachdem die Anforderungen an die technische Realisierung von VDLs vorliegen, können mögliche Realisierungskonzepte diskutiert werden. Dazu soll zunächst die Klasse entsprechender VDL-basierter Integrationstechniken untersucht werden. Diese Klasse wird auf Basis der Anforderungen charakterisiert und als *E-Services* eingeführt. Konkreter sollen als E-Services diejenigen Techniken bezeichnet werden, die der Realisierung virt. Vorgänge und Abläufe von VDLP-Interaktionen dienen. Techniken zur Realisierung von Prozessen und Methoden in Bezug auf Spezifikation, Abgleich, Ausführung, Analyse und Änderung von VDLP-Interaktionen bilden hingegen das *E-Service-Management*. Die E-Service-Techniken werden dann in das Schema allg. integrativer IV-Infrastruktur für virt. Dienstleistungsunternehmenen eingeordnet. Dabei werden die Beziehungen zu verschiedenen anderen Integrationstechniken erläutert und der Beitrag einer E-Service-Technologie herausgestellt. Bei dieser Betrachtung werden auch diejenigen technischen Aspekte bestimmt, die bei der Realisierung einer E-Service-Technologie als unmittelbare Grundlagen einfließen. Bezüglich der Realisierung wird zudem ein generelles Vorgehensmodell abgeleitet.

E-Services bezeichnen ein Realisierungskonzept für VDL. Die konkrete Realisierung kann dabei in unterschiedlicher Art und Weise erfolgen. Dies hängt im Wesentlichen von den verwendeten Basistechniken ab. Diese Basistechniken müssen in erster Linie die Repräsentation von VDLP-Interaktionen unterstützen. Darüber hinaus hat die Wahl dann auch Einfluss auf die verschiedenen Management-Methoden. In Bezug auf die Anforderungen zur Realisierung virt. Vorgänge und Abläufe bietet sich insbesondere die Web Service-Abstraktion mit den verschiedenen SOC-Mechanismen zur Repräsentation und Durchführung einfacher und komplexer zwischenbetrieblicher Interaktionen an. Daher schließt sich an die grundlegende Einordnung VDL-basierter Techniken eine spezifische Untersuchung zur Realisierung von E-Services durch SOC-Techniken an. Dabei wird zunächst das Service-orient. Paradigma in den Kontext der IV-Infrastruktur virt. Dienstleistungsunternehmenen eingeordnet. Dann werden die Potenziale einzelner WS-Techniken aus den Bereichen der Basisarchitektur und Prozessintegration in Bezug auf die Realisierung verschiedener Aspekte von E-Services untersucht. Dies bildet im weiteren Verlauf eine Grundlage für die Realisierung des E-Service-Managements durch ein Vorgehensmodell zur Service-orient. Entwicklung von E-Services, was in Sektion 4.4 als dritter Teil des konzeptionellen Rahmenwerks untersucht werden wird.

E-Services als Integrationstechnik zur VDL-Realisierung Wie schon in den vorangegangenen Abschnitten dargestellt wurde, beschreibt das konzeptionelle Modell virt. Dienstleistungen den Einsatz von IV-Techniken zur Koordination von Akteuren innerhalb des spezifischen Kontextes virt. Dienstleistungsproduktionsnetzwerke. Die Aufgabe der IV-Techniken besteht dabei (a) in der Abbildung und (b) im Management von Interaktionsprozessen zwischen VDPN-Akteuren. Interaktionsprozesse bilden die Verbindung zwischen Dienstleistungen und der Koordination ihrer virtuell organisierten Produktion. Bei Dienstleistungen ist der Interaktionsprozess inhärenter Bestandteil des zentralen DLP. Bei der VU-Koordination werden die Beziehungen der Akteure über deren Interaktionsprozess geregelt. Auf Grund dieser Überschneidung kann durch Manipulation des DLP die Koordination im VDPN beeinflusst werden. Das Grundprinzip virt. Dienstleistungen besteht darin, durch Virtualisierung des DLP dessen Gestaltung durch strukturierte Prozesse und zum Teil automatisierte Methoden zu ermöglichen. Diese werden dann als Instrument zur Koordination der Akteure im VDPN verwendet.

Die Virtualisierung von Interaktionsprozessen beinhaltet die beiden Aspekte der Interaktionsvorgänge und der Ablaufkoordination. Die Virtualisierung manueller Interaktionsvorgänge erfolgt durch formale Abbildung der dabei ablaufenden Kommunikation und durch automatische Durchführung entsprechender Kommunikationsvorgänge zwischen Anwendungssystemen von Produktionseinheiten. Die Virtualisierung manueller Ablaufkoordination erfolgt mittels formaler Abbildung koordinativer Kommunikationsregeln und deren automatischer Durchsetzung durch Anwendungssysteme von Koordinatoren. Der Virtualisierungsvorgang von Interaktionsprozessen stellt im Sinne dieser Charakteristik eine Integration von Anwendungssystemen dar.[28] Anwendungssysteme liegen im Wesentlichen als Administrations- und Dispositionssysteme der verschiedenen Produktionseinheiten vor. Es handelt sich also um eine zwischenbetriebliche horizontale Integration operationaler Systeme. Hierbei müssen die verschiedenen Integrationsaspekte berücksichtigt werden. Den Hintergrund bildet die Integration von Kernkompetenzen der Netzwerkteilnehmer. Hierzu erfolgt zum einen eine funktionale Integration und zum anderen eine Prozessintegration in Bezug auf Inhalte und Verhaltensaspekte der Schnittstellen von Produktionseinheiten. Da die Integration auf einem Interaktionsprozess basiert, bilden Interoperabilität der Kommunikation, Datenintegration und vor allem Koordination im untersuchten Kontext die wichtigsten Aspekte. Auf Basis einer solchen Integration kann ein hoher Automatisierungsgrad erreicht werden. Das Ergebnis ist ein kooperatives Anwendungssystem, das einen globalen Dienstleistungsprozess realisiert und die Dienstleistungsproduktion steuert. Der integrative Kern koordiniert die Kommunikation zwischen operativen Systemen der Produktionseinheiten. Er repräsentiert den virtualisierten Interaktionsprozess. Dieser integrative Kern wird in der vorliegenden Arbeit als *E-Service* bezeichnet und wie folgt definiert:

[28] Vgl. Sektion 3.2.1.2 zur Integration betrieblicher IT-Systeme.

Definition 17 (E-Service) *Der Begriff des E-Service bezeichnet die Abbildung eines virt. Dienstleistungsprozesses als koordinierten Ablauf von Kommunikationsvorgängen zwischen operativen Anwendungssystemen von Produktionseinheiten eines virt. Dienstleistungsproduktionsnetzwerks zum Zweck der Integration zu einem zwischenbetrieblichen Anwendungssystem, das der Steuerung der Dienstleistungsproduktion dient.*

Während die Realisierung der Integration eines kooperativen Anwendungssystems zur Steuerung der Dienstleistungsproduktion alleine schon von großem Nutzen ist, gehen die Anforderungen im Kontext von VDPN noch darüber hinaus. Hier wird angestrebt, den virt. Dienstleistungsprozess als Steuerprogramm einzelner Produktionsdurchläufe individuell anzupassen (Flexibilität) und diese Anpassung möglichst ad hoc vorzunehmen (Agilität). Diese Anforderungen richten sich an die Erstellung, Durchführung und Auswertung von VDLP in den Phasen virt. Dienstleistungen. Übertragen auf die Realisierung von VDLP durch E-Services müssen die Phasen virt. Dienstleistungen durch entsprechende Prozesse und Methoden zur flexiblen und agilen Entwicklung von E-Services realisiert werden. Konkret bezieht sich das auf die Entwicklung des integrativen Kerns eines kooperativen Anwendungssystems. Dieser koordiniert die Kommunikation zwischen operativen Systemen von VDPN-Produktionseinheiten. Die Entwicklungsprozesse und -methoden müssen den Entwurf von E-Service-Spezifikationen, die Konstruktion entsprechender integrativer Komponenten des kooperativen Anwendungssystems und die Evaluation der Integration in Bezug auf Änderungen leisten. Ein derartiges Vorgehen wird in der vorliegenden Arbeit als E-Service-Management bezeichnet und wie folgt definiert:

Definition 18 (E-Service-Management) *Der Begriff des E-Service-Managements umfasst Entwicklungsprozesse und Methoden zur Erstellung, Durchführung und Auswertung von E-Services, die eine derartige Flexibilität und Agilität aufweisen, dass sie den Anforderungen zur ad hoc-Planung in virt. Dienstleistungsproduktionsnetzwerken genügen.*

E-Services und E-Service-Management kombinieren verschiedene Aspekte der generischen IV-Infrastruktur für virt. Dienstleistungsunternehmenen im Sinne der organisatorischen Anforderungen von virt. Dienstleistungsproduktionsnetzwerken. Entsprechend können die Anteile der Technologie in verschiedene Bereiche einer spezifischen IV-Infrastruktur von VDPN eingeordnet werden. Tabelle 4.2 stellt diese Beziehungen dar.

Zunächst kann festgestellt werden, dass E-Services eine anwendungsspezifische Integrationstechnik der Basisinfrastruktur darstellen. Der Schwerpunkt dieser Technik liegt zum einen auf dem Aspekt der interoperablen Kommunikation mit AS der Produktionseinheiten. Deren Vernetzung wird hierbei auf Basis des Internets vorausgesetzt. Das Gleiche gilt für eine grundlegende Sicherheit der Datenübertragung, die einen orthogonalen Aspekt darstellt. Im Rahmen von E-Services wird stattdessen die

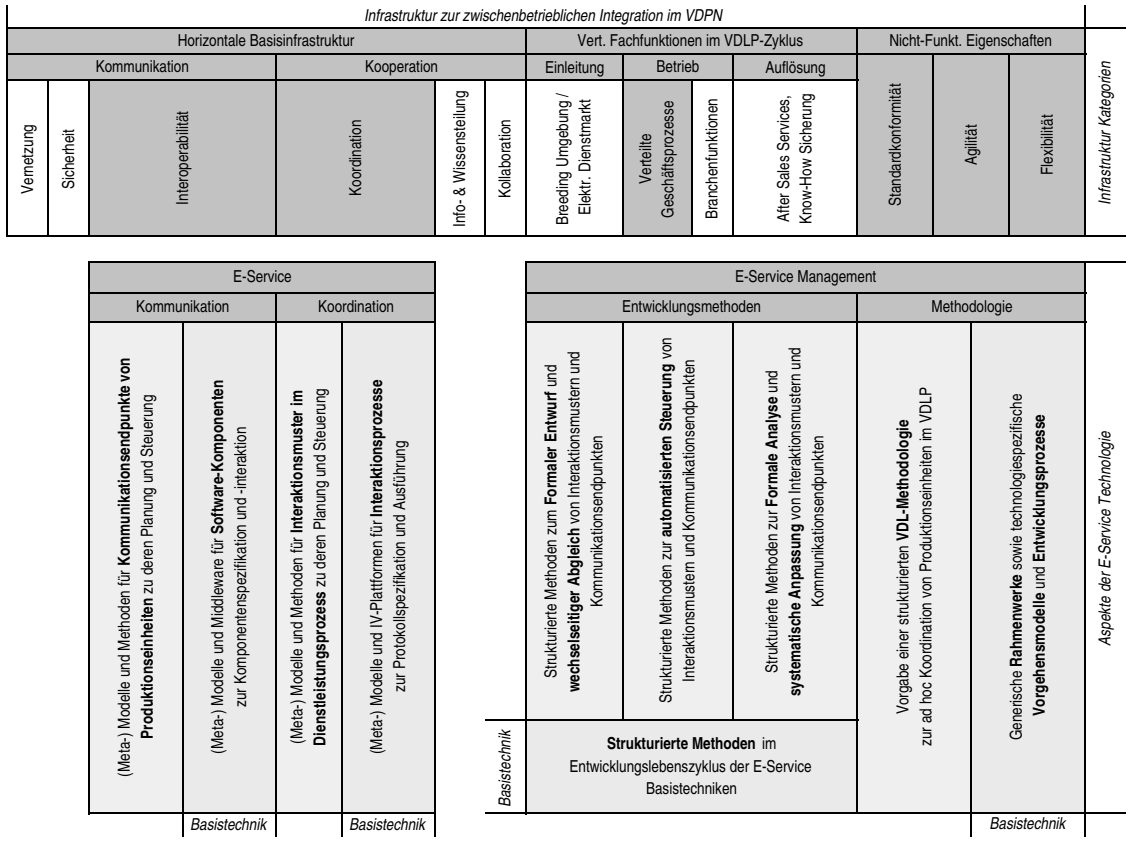

Tabelle 4.2.: Einordnung von E-Service-Techniken in die IV-Infrastruktur von VDPN

spezifische Kommunikation mit Endpunkten der Produktionseinheiten modelliert und implementiert. Dazu können vorhandene Middleware-Techniken eingesetzt werden, die hier verschiedene Formen zwischenbetrieblicher Kommunikation von Software-Komponenten ermöglichen. Ein weiterer Schwerpunkt von E-Services liegt auf dem Aspekt der Koordination von Kommunikationsvorgängen. Es handelt sich dabei um eine Form der Kooperationsunterstützung, die im Gegensatz zur Kollaboration auf strukturierte Prozesse ausgelegt ist. Die Integration im VDPN findet nämlich auf operationaler Ebene der betrieblichen IV statt. Informations- und Wissensteilung wird hier als orthogonaler Aspekt vorausgesetzt. Im Rahmen der Koordinationstechniken bilden E-Services eine spezifische Form zur Modellierung und Implementierung dienstleistungsspezifischer Interaktionsmuster. Hierbei können vorhandene IEWS-Techniken als Grundlage dienen.

Die Prozesse und Methoden des E-Service-Managements lassen sich hingegen der vertikalen IV-Infrastruktur virt. Dienstleistungsproduktionsnetzwerke zuordnen. Hierbei bilden die einzelnen Methoden Beiträge zu den (Fach-)Funktionen des Produktionsmanagements in verschiedenen Phasen des VDPN-Lebenszyklus. In Bezug

auf die Kategorien generischer Fachfunktionen kann das E-Service-Management der Unterstützung zwischenbetrieblicher Geschäftsprozesse zugeordnet werden. Letzteres wird im Grundmodell der Infrastrukturkategorien dem VDPN-Betrieb zugeordnet. E-Service-Management bezieht sich aber auf den gesamten Lebenszyklus. In der VDPN-Einleitung wird die Produktionsplanungsfunktion durch formalen Entwurf und Abgleich von Interaktionsmustern und Kommunikationsendpunkten unterstützt. Im VDPN-Betrieb trägt die automatisierte Ausführung entsprechender Integrationsmechanismen zur Produktionssteuerungsfunktion bei. Bei der VDPN-Auflösung leistet die formale Analyse und systematische Anpassung von Interaktionsmustern und Kommunikationsendpunkten einen weiteren Beitrag zur Produktionsplanungsfunktion. Die spezifischen Fachfunktionen können generell von vorhandenen Entwicklungsmethoden der für die E-Service-Realisierung verwendeten Basistechniken profitieren. Dazu gehören etwa die objektorientierte Modellierung und generative Techniken für Middleware-Komponenten oder die verschiedenen Managementfunktionen von Inter-Enterprise Workflow-Systemen.

Für die Nutzenwirkung der verschiedenen IV-Techniken der IV-Infrastruktur sind bestimmte nicht-funktionale Eigenschaften von nicht zu unterschätzender Bedeutung. Wenn die generelle Integrationsfunktionalität der verschiedenen Techniken nicht flexibel und agil zum Einsatz gebracht werden kann, stellen sich die Vorteile virtueller Organisation nämlich nicht ein. E-Service-Technologie trägt diesem Umstand Rechnung, indem sie das Konzept von Plänen umsetzt, die in Hinblick auf nicht-funktionale Eigenschaften optimiert werden müssen. Die Umsetzung erfolgt dabei durch ein Vorgehensmodell für das E-Service-Management. Hierbei können generische Referenzmodelle und Rahmenwerke sowie Vorgehensmodelle und Entwicklungsprozesse für die verwendeten Basistechniken den Hintergrund bilden.

An dieser Stelle ist die spezifische Klasse der zur Dienstleistungsvirtualisierung notwendigen IV-Technik charakterisiert und abgegrenzt. Auf dieser Grundlage kann nun die Realisierung spezifischer Lösungen untersucht werden. Diese Realisierung von E-Services sollte auf Grund der integrativen Charakteristik dem allg. Vorgehen einer Systemintegration folgen.[29] Dies beinhaltet die Nutzung von Rahmenwerken, die Erstellung von Systemmodellen, die Harmonisierung der Komponenteninteraktion, die Informationsverknüpfung und die Etablierung einer Koordinationsebene. Konkret gestalten sich diese Schritte wie folgt:

- *Nutzung von Rahmenwerken* – E-Services müssen sich als betriebliche IV-Technik in den Kontext betrieblicher Informationssysteme eingliedern. Die Integration mit anderen IV-Techniken und den betrieblichen Prozessen ist ein komplexer und dynamischer Vorgang. Dieser sollte zur Sicherung einer konsistenten Vorgehensweise im Rahmen übergeordneter Rahmenwerke und Referenzmodelle erfolgen. Verschiedene Ansätze nehmen oft orthogonale Perspektiven ein und

[29] Siehe Sektion 3.2.1.2.

können auch kombiniert werden. Ein entsprechendes Rahmenwerk wird z. B. in Form des FRESCO-Ansatzes in den Sektionen dieses Kapitels vorgestellt. Abstraktere Referenzmodelle finden sich in Bezug auf Enterprise-Architektur.

- *Erstellung von Systemmodellen* – E-Service-basierte Integration führt zu zwischenbetrieblich verteilten Anwendungssystemen erheblicher Komplexität. Das Verständnis derartiger Systeme und die Entwicklung strukturierter Entwicklungsprozesse und -methoden sollten deshalb durch ein Systemmodell gefördert werden. Dies sollte in Form von Metamodellen erfolgen, die zum einen die abstrakte Systemklasse beschreiben und zum anderen einen Rahmen zur Modellierung konkreter Systeme vorgeben. Für E-Services sind dabei vor allem Metamodelle von Interaktionsmustern und Kommunikationsendpunkten von Interesse.

- *Harmonisierung der Komponenteninteraktion* – Kommunikationsvorgänge mit operativen Anwendungssystemen der Produktionseinheiten im VDPN bilden einen grundlegenden Aspekt von E-Services. Um dies im zwischenbetrieblichen Kontext zu ermöglichen, müssen Vorgaben für einheitliche Paradigmen, Modelle und Schnittstellen der Kommunikation aller Produktionseinheiten gemacht werden. Dies kann insbesondere die Vorgabe zur Verwendung von Basistechniken zwischenbetrieblicher Middleware beinhalten.

- *Informationsverknüpfung* – Bei den Kommunikationsvorgängen im Rahmen von E-Services findet ein Informationsaustausch mit den Produktionseinheiten des VDPN statt. Um im zwischenbetrieblichen Kontext ein gemeinsames Verständnis dieser Informationen zu gewährleisten, müssen wiederum Vorgaben erfolgen. Hierbei können z. B. branchenspezifische Standards für kommunizierte Dokumente verwendet werden. Eine andere Möglichkeit besteht in der Festlegung generischer Standards zur Datenbeschreibung, die zur Festlegung von Datenmodellen verwendet werden.

- *Etablierung einer Koordinationsebene* – Die Koordination von Kommunikationsvorgängen ist der zweite Kernaspekt integrativer E-Service-Techniken. Die Regelung der Koordination basiert hierbei auf der Spezifikation von Interaktionsmustern zwischen Produktionseinheiten. Hierfür muss ein formales Modell dienstleistungsspezifischer Interaktionsprotokolle entworfen werden. Zudem muss festgelegt werden, wie Regelungen durch entsprechende Protokolle im Betrieb des VDPN durchgesetzt werden sollen. Generell können hier Ansätze zur zwischenbetrieblichen Prozessintegration als Grundlage dienen.

Auf Basis dieses Vorgehensmodells kann eine E-Service-Technologie prinzipiell auf ganz unterschiedliche Art realisiert werden. Eine wesentliche Determinante ist dabei die Wahl von Basistechniken, vor allem in Bezug auf die Komponenteninteraktion und

die Koordination entsprechender Interaktionsprozesse. Unter diesem Gesichtspunkt bieten sich ganz besonders SOC-Techniken an, da in deren Kontext beide Aspekte in homogener Form abgedeckt werden. Im Folgenden werden die entsprechenden Potenziale einzelner SOC-Techniken eingehender untersucht.

Realisierung von E-Services durch SOC-Techniken Der E-Service-Begriff umschreibt eine Art der Integrationstechnik, die auf den spezifischen Kontext virt. Dienstleistungsproduktionsnetzwerke ausgelegt ist und sich in deren IV-Infrastruktur eingliedert. Das technische Konzept beinhaltet sowohl Anteile der horizontalen Basisinfrastruktur als auch verschiedene generische Methoden zur Unterstützung des VDPN-Lebenszyklus. Als Teil der horizontalen Basisinfrastruktur liegt der Schwerpunkt von E-Service-Techniken auf den Aspekten der interoperablen Komponenteninteraktion sowie der Koordination dieser Interaktionen. Dabei leisten sie einen Beitrag zur Spezialisierung genereller Integrationstechniken im Kontext virt. Dienstleistungsproduktionsnetzwerke und unter dem Gesichtspunkt des Produktionsmanagements. In diesem Kontext decken sich die Anforderungen an generelle Funktionen der Komponenteninteraktion und Koordination weitgehend mit denen des allg. B2Bi, d. h. es wird eine global standardisierte Interoperabilitätsplattform benötigt, deren Mechanismen zur Integration von Anwendungssystemen für den zwischenbetrieblichen Kontext ausgelegt sind. In der vorliegenden Arbeit wird der diesbezügliche Einsatz von SOC-Techniken fokussiert.

Theoretisch können vielfältige Basistechniken zur Realisierung von E-Services angewendet werden. Jedoch haben SOC-Techniken durch ihre Potenziale zur zwischenbetrieblichen Integration und Interaktion erst den Anstoß zur Untersuchung von Möglichkeiten der Virtualisierung von Dienstleistungen gegeben. Web Services erlauben die zwischenbetriebliche Repräsentation geschäftlicher Interaktionen und den Zugriff auf gekapselte innerbetrieblicher Prozesse und Anwendungssysteme. Darauf setzen verschiedene Integrationsmechanismen sowohl für die Durchführung einzelner Interaktionen als auch die Regelung und Implementierung komplexer Interaktionen auf. Diese sind explizit für die Anforderungen im zwischenbetrieblichen Kontext ausgelegt und daher in Bezug auf kritische Aspekte dezentralisiert. Alle Techniken sind zudem standardisiert und weit verbreitet. Insgesamt eröffnen SOC-Techniken die Möglichkeiten einer integrativen Middleware für den organisationsübergreifenden Kontext von VDPN. Hier können sie durch das E-Service-Konzept auf die Ebene des Produktionsmanagements angehoben werden.

Die Web Service-Abstraktion und die damit einhergehenden Mechanismen, die u. a. Beschreibung, Publikation, Bindung, Zugriff, Koordination, Komposition und Management von Web Services im zwischenbetrieblichen Kontext ermöglichen, bieten sich im Zusammenhang der Dienstleistungsvirtualisierung insbesondere als Grundlage zur Realisierung virt. Dienstleistungsprozesse an. Sie bilden dabei Basistechniken, die in Bezug auf die Anteile des E-Service-Konzepts zu spezialisieren sind. Entsprechende E-

Services stellen sich dann als Service-orient. Komponenten zur Integration kooperativer Anwendungssysteme im virt. Dienstleistungsproduktionsnetzwerk dar. Methoden des E-Service-Managements bilden entsprechende Verfeinerungen allgemeiner Service-orient. Entwicklungsmethoden in Bezug auf das VDPN-Produktionsmanagement. Im Folgenden werden zunächst die Möglichkeiten zur Service-orient. Realisierung von E-Services untersucht. Ein darauf aufsetzendes Service-orient. Vorgehensmodell für das E-Service-Management folgt weiter hinten in Sektion 4.4.

Grundsätzlich kann zunächst festgestellt werden, dass sich das Service-Paradigma sehr gut in den Kontext virt. Dienstleistungsunternehmenen eingliedert. Die Konzepte des einfachen und erweiterten Service-orient. Modells lassen sich nämlich ganz natürlich mit denen des VDPN-Modells identifizieren. Dies liegt zum einen daran, dass das Service-Paradigma natürlich vom Grundkonzept her auf dem Dienstleistungsmodell beruht. Entsprechende Übereinstimmungen finden sich bei den Rollen und Rollenfunktionen dieser beiden Modelle. In beiden Grundmodellen werden Clients und Provider unterschieden. Der Potenzialerstellung von Providern des Dienstleistungsmodells steht die Implementierung und Beschreibung von Web Services durch SOM-Provider gegenüber. Die interaktive Abwicklung des Dienstleistungsprozesses zwischen Client und Provider entspricht dem Web Service-Zugriff.

Zum anderen spiegelt das erweiterte SOM die Verhältnisse im virt. Dienstleistungsproduktionsnetzwerk wider. Web Services werden hier von einer Menge verschiedener Service Provider erbracht, die mit den Produktionseinheiten eines VDPN identifiziert werden können. Die Service Provider werden von einem Service-Aggregator kombiniert, der konzeptionelle Parallelen mit dem Koordinator im VDPN besitzt. Nach außen werden Web Services schließlich von einem Service-Betreiber angeboten, der mit dem VDPN-Broker konform geht. In dieser Konstellation kommen die SOC-Mechanismen dann insbesondere der Virtualität der Organisationsform zugute. Diese erlauben es einzelnen Service Providern, das Angebot ihrer WS-Komponenten zu spezifizieren und zu publizieren. Der Service Betreiber wird durch Service Broker dabei unterstützt, alternative Service Provider für die Erbringung komplexer Web Services zu identifizieren. Der Aggregator kann identifizierte WS-Komponenten durch Service-Komposition kombinieren und gleichzeitig eine lose Bindung zu den Service Providern bewahren, um diese ggf. später wieder zu ersetzen. Dieser Ablauf weist große Ähnlichkeit mit der Bildung eines VU auf. Die Flexibilität bei der Aggregation von Service Providern und die Agilität der dabei verwendeten automatisierten Mechanismen fördern hier prinzipiell die Effektivität der virtuellen Organisationsform.

Die Konzeptionellen Übereinstimmungen der Modelle führen dazu, dass sich die Abstraktionen und Mechanismen des Service-orient. Modells auch in natürlicher Weise in das E-Service-Konzept eingliedern lassen. Tabelle 4.3 gibt einen Überblick.

SOC-Techniken besitzen Potenziale für die Realisierung von E-Service-Basisfunktionen in beiden grundsätzlichen Kategorien integrativer Techniken. Zunächst können durch SOC-Techniken Basisfunktionen zur interoperablen Kommunikation beige-

E-Service					SOC Potential	SOC Substituierung	WS-Technologien	
Kommunikation	(Meta-) Modelle und Methoden für Kommunikationsendpunkte von Produktionseinheiten zu deren Planung und Steuerung		Erweiterung					
	(Meta-) Modelle und Middleware für Software-Komponenten zur Komponentenspezifikation und -interaktion	Mechanismen zur Publikation von Schnittstellenbeschreibungen	Basistechnik	Service-orientierte Systemintegration	Auffindung & Auswahl durch Verzeichnisse	WS Vermittlung	Z.B. UDDI
		Standardsprachen zur Schnittstellenbeschreibung			Beschreibung & Publikation durch IDLs	WS Beschreibung	Z.B. WSDL
		Mechanismen zur Schnittstellenkommunikation			Bindung & Interaktion durch P2P-Middleware	WS Zugriff	Z.B. SOAP
Koordination	(Meta-) Modelle und Methoden für Interaktionsmuster im Dienstleistungsprozess zu deren Planung und Steuerung		Erweiterung					
	(Meta-) Modelle und IV-Plattformen für Interaktionsprozesse zur Protokollspezifikation und Ausführung	Prozessrepräsentation als Koordinationsprotokoll	Basistechnik	Service-orientierte Prozessintegration	Regelung von Kollaboration durch Koordinationsprotokolle	WS Koordination	Z.B. BPEL, WSCI
		Ausführung von Prozessrepräsentationen a.B.v. Schnitstellenkommunik.			Implementierung von Koordinationsregeln durch Kompositionsschemas	WS Komposition	Z.B. BPEL

Tabelle 4.3.: SOC-Techniken für E-Services

steuert werden. Das E-Service-Konzept der Kommunikationsendpunkte von Produktionseinheiten lässt sich hier direkt auf Web Services abbilden. Entsprechend können Modelle zur Planung und Spezifikation von Kommunikationsendpunkten auf den verschiedenen Modellen zur WS-Beschreibung aufsetzen. Hier kann allen voran WSDL genannt werden. Die Steuerung der Schnittstellenkommunikation von Kommunikationsendpunkten wird dementsprechend auf Mechanismen zur automatischen Durchführung von WS-Zugriffen übertragen. Dabei spielt SOAP als interoperables XML-basiertes Protokoll zum automatisierten Austausch von Nachrichten eine wesentliche Rolle. Die Planungsfunktion des VDPN-Produktionsmanagement kann schließlich noch von weiteren SOC-Mechanismen profitieren. Dazu gehört z. B. die Publikation von Service-Beschreibungen durch UDDI.

Die zweite wichtige Kategorie der spezifischen Integrationsfunktionalität von E-Services betrifft die Koordination. Hier bietet sich der Einsatz Service-orient. Techniken zur Prozessintegration an. Grundsätzlich bieten SOC-Techniken hier Möglichkeiten zur Koordination und Komposition von WS-basierten Interaktionen. Service-Koordination ermöglicht die Spezifikation von Konversationen und Choreografien als Protokollen zur Regelung der Interaktion zwischen Service Providern. Dabei werden in den meisten Fällen formale Prozessmodelle verwendet. Dies lässt sich als Grundlage für die Planung und Spezifikation von Interaktionsmustern in Dienstleistungsprozessen anwenden. Service-Komposition beinhaltet daneben Techniken, die zur Implementierung von Interaktionsprotokollen durch korrespondierende Kompositionsschemata dienen. Dies kann eine Basis zur automatisierten Steuerung von Interaktionsmustern durch E-Services bilden.

Bei Nutzung Service-orient. Basistechniken können SOC-Techniken in verschiedenartiger Weise die Prozessintegration im Rahmen von E-Services unterstützen. Dies kann grundsätzlich in Bezug auf die Regelung und Implementierung von Interaktionsabläufen erfolgen.[30] Von den verschiedenen Ansätzen zur Service-orient. Regelung von Interaktionen bietet sich im Kontext von E-Services vor allem die freie Variante als sinnvolle Unterstützung an. Dies ermöglicht VDPN-Providern die autonome Planung und Vorgabe der externen Interaktionsprozesse ihrer Produktionseinheiten. VDPN-Broker können durch Methoden der freien Regelung die Vorgaben von Providern in manueller Weise kombinieren. Sie können aber auch von Methoden der abhängigen Regelung profitieren, um Vorgaben von Providern durch automatisierte Verfahren zu verarbeiten. Bei den Service-orient. Methoden zur Implementierung von Regelungen sind dann neben freien Varianten vor allem automatische Techniken von Interesse. Diese fördern nämlich die wichtigen Aspekte der Flexibilität und Agilität bei der Planung. Dies ist der Fall, da dann Regeländerungen ohne Zeitverlust in Implementierungen überführt werden können.

Insgesamt zeigt sich bei der Gegenüberstellung von E-Services und Service Oriented Computing, dass die zugrunde liegenden Modellen klare Übereinstimmungen zeigen. Auf dieser Basis wurden verschiedene SOC-Techniken als Basistechniken zur Realisierung von E-Services identifiziert. Hierauf aufbauend lassen sich verschiedene Varianten für eine Abbildung von Modellen und Techniken finden. Diese fundieren im Wesentlichen auf dem zugrunde liegenden VDL-Modell. Im Folgenden wird ein solches technisches Konzept zur Realisierung von E-Services für das FRESCO-Dienstleistungsmodell ausgeführt.

4.3.2. SOM-Erweiterung und SOA-Entwurf in FRESCO

Zur Konkretisierung der Diskussion über die Realisierung von virt. Dienstleistungen durch E-Services soll im Folgenden das Realisierungskonzept im FRESCO-Projekt vorgestellt werden. Hierbei wird grundsätzlich der in den letzten Sektionen beschriebene Ansatz aufgegriffen, die Realisierung von E-Services auf Basistechniken des SOC aufsetzen zu lassen. In diesem Sinne beschäftigt sich Sektion 4.3.2.1 zunächst mit der Abbildung des FRESCO-Dienstleistungsmodells auf das klassische Service-orient. Modell. Hierbei werden die grundsätzlichen Kategorien einer integrativen E-Service-Technik entsprechend der bisherigen Diskussion auf Kategorien von SOC-Techniken abgebildet. Dies wird mit den Konzepten des FRESCO-Dienstleistungsmodells kombiniert. Dabei ergibt sich u. a. auf Basis des Prinzips zur Trennung von Content und Provision eine im Vergleich zum klassischen SOM verfeinerte Struktur. Das Resultat ist ein spezifisches FRESCO-SOM, das Rollen und Funktionen im Rahmen von Service-orient. Anwendungssystemen zur Produktionssteuerung in virt. Dienstleistungsproduktionsnetzwerken charakterisiert.

[30] Siehe Sektion 3.4.3.1.

Das FRESCO-SOM stützt sich auf grundlegende Aspekte des Service-orient. Paradigmas. Dazu gehören die Web Service-Abstraktion sowie die darauf aufsetzende Regelung und Implementierung von Interaktionsabläufen. Entsprechend diesem Modell können E-Services durch verschiedene SOC-Techniken realisiert werden. Z. B. könnte eine Implementierung der automatischen Steuerung von Interaktionsabläufen u. a. auf Basis der Sprachen BPEL oder WSFL erfolgen. Dabei könnten zur Implementierung eines FRESCO E-Service sogar verschiedene dieser Techniken kombiniert werden. Gerade im Kontext der WS-Koordination und -Komposition ist die Entwicklung von SOC-Standards und -Techniken noch nicht abgeschlossen und ändert sich häufig. Aus diesem Grund soll für das FRESCO-SOM keine direkte Abbildung auf eine konkrete SOC-Technik festgelegt werden. Stattdessen wird die Realisierung von E-Services durch ein generisches Metamodell festgelegt. Im FRESCO E-Service-Metamodell werden verschiedene Service-orient. Basismechanismen in technikneutraler Weise repräsentiert. Diese Repräsentationen lassen sich auf entsprechende konkrete SOC-Techniken abbilden, ohne von einer spezifischen Technik abhängig zu sein. Die generischen Basismechanismen werden im Metamodell dann zu den Elementen von FRESCO-E-Services in Beziehung gesetzt. Das FSM dient in der Folge als Grundlage für eine Reihe von Entwicklungsmethoden für FRESCO-E-Services. Es dient dabei als Schema zum Service-orient. Entwurf von FRESCO-E-Services und erlaubt die generative Implementierung von E-Service-Modellen auf Basis verschiedener konkreter SOC-Techniken. In Sektion 4.3.2.2 erfolgt zunächst die Spezifikation des FSM auf Basis von UML. Ein Vorgehensmodell zur flexiblen und agilen Entwicklung von FRESCO-E-Services im Kontext von VDPN wird dann im dritten Teil des konzeptionellen Rahmenwerks ausgearbeitet.

4.3.2.1. Das Service-orient. FRESCO-Modell

Durch das FRESCO-Dienstleistungsmodell wird ein spezifisches Konzept für virt. Dienstleistungen festgelegt.[31] Der Kernaspekt liegt in der Einführung und wechselseitigen Abgrenzung der Konzepte von Dienstleistungsinhalten (Content) und Dienstleistungserbringung (Provision). Dienstleistungsinhalte basieren auf einer Menge von Anlagen oder Werten (Assets), die zusammengenommen den Kern (Core) einer Dienstleistung bilden. Jedes Asset stellt einen in sich geschlossenen inhaltlichen Aspekt dar, der durch einen Provider auf Basis seiner Ressourcen realisiert wird. Die Nutzung eines inhaltlichen Aspekts wird dabei über einen klar spezifizierten Interaktionsvorgang geregelt. Dem Dienstleistungsinhalt steht ein Dienstleistungskonsum (Consumption) gegenüber. Dienstleistungskonsum begründet sich auf dem Bedarf (Demand) eines Client. Ein Bedarf bezieht sich dabei genau wie ein Asset auf einen in sich geschlossenen inhaltlichen Aspekt, der in Bezug auf den Interaktionsvorgang zur Nutzung klar geregelt ist. Angebot und Nachfrage werden durch einen admi-

[31] Siehe Sektion 4.2.2.

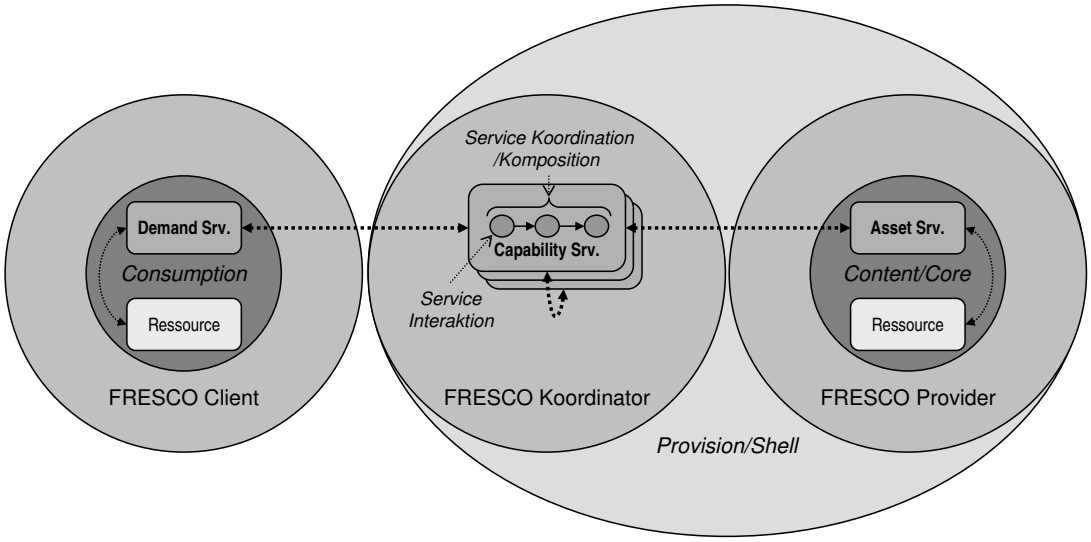

Abbildung 4.15.: Grundstruktur des Service-orient. FRESCO-Modells

nistrativen Prozess zusammengeführt, der die Nutzung von Assets zur Erfüllung von Demands regelt. Dieser Prozess der Dienstleistungserbringung basiert auf einer Menge von Fähigkeiten (Capabilities). Eine Capability regelt dabei den Ablauf einer in sich geschlossenen Interaktion zwischen Akteuren im Zuge eines administrativen Prozesses. Sie dient als Grundlage zur Interaktionssteuerung durch einen Koordinator. Dies kann Interaktionen zwischen Clients, Providern und Koordinatoren betreffen. Im Sinne der gängigen Gliederung des DLP in Kern-, Hilfs- und Steuerprozesse können Capabilities sich auf die Nutzung von Assets (Kernprozess), die Kombination von Assets (Hilfsprozess) oder die Koordination von Teilprozessen (Steuerungsprozess) beziehen. Die Menge aller Capabilities dient zur Regelung eines zusammenhängenden Interaktionsablaufs zur integrierten Nutzung der verschiedenen Assets und bildet so eine Hülle (Shell) um den Dienstleistungskern.

Entsprechend der allg. Diskussion in den vorangegangenen Sektionen erfordert die Realisierung des FRESCO-Dienstleistungsmodells eine Abbildung virtueller Modellelemente auf konkrete IV-Techniken. Dies beinhaltet im Wesentlichen integrative Techniken zur Repräsentation des FRESCO-VDLP sowie Methoden zu dessen Erstellung, Durchführung und Auswertung oder kurz: *FRESCO-E-Services* und *FRESCO E-Service-Management*. Deren Realisierung basiert in FRESCO grundsätzlich auf dem Service-orient. Paradigma. Die Art und Weise der Service-orient. Realisierung von FRESCO E-Services und deren Management wird in einem Modell beschrieben, das als Service-orient. FRESCO-Modell (FRESCO-SOM) bezeichnet wird [ZLP04, ZL04a]. Das Grundprinzip des FRESCO-SOM ist in Abb. 4.15 dargestellt.

E-Services verkörpern eine Abbildung von virt. Vorgängen und Abläufen eines VDLP auf Techniken zur Realisierung von Kommunikationsendpunkten und Interaktions-

mustern. Diese bewirken eine koordinierte Kommunikation von Anwendungssystemen der Produktionseinheiten eines VDPN zum Zweck der Produktionssteuerung. Grundsätzlich lässt sich zunächst feststellen, dass Assets und Demands eine Spezialisierung allg. virt. Vorgänge darstellen und somit in FRESCO E-Services als Kommunikationsendpunkte abgebildet werden. Capabilities bilden hingegen eine Verfeinerung allg. virt. Abläufe, die sich in FRESCO E-Services als Interaktionsmuster wiederfinden. Im FRESCO-SOM werden diese Aspekte der Kommunikation und Koordination beide als Services realisiert.

Auf der einen Seite werden die Kommunikationsendpunkte von Assets und Demands als atomare Services repräsentiert. Der Interaktionsvorgang zur Nutzung von Dienstleistungsinhalten wird dementsprechend auf Basis einer Service-Beschreibungssprache spezifiziert. Die Beschreibung soll hier zumindest die operationale Schnittstelle einbeziehen. Darüber hinaus ist eine optionale Verhaltensbeschreibung in Form der unterstützten Konversation von großem Nutzen. Für die Steuerung der automatisierten Schnittstellenkommunikation ist dann der externe Zugriff auf die Services vorgesehen. Die Implementierung der Services liegt hingegen außerhalb des Modells. An dieser Stelle knüpfen Modelle interner Service-orient. Architektur an. Diese beschreiben die Implementierung der durch Services repräsentierten geschäftlichen Interaktionen durch interne Prozesse und Anwendungssysteme von Provider bzw. Client. Entsprechende Techniken basieren oft auf A2Ai-Middleware.[32]

Auf der anderen Seite werden die Interaktionsprozesse von Capabilities durch zusammengesetzte Services repräsentiert. Der Interaktionsablauf des administrativen Prozesses wird dabei als Folge von Service-Interaktionen spezifiziert. Dies geschieht durch ein Interaktionsprotokoll. Das Protokoll dient der Regelung der Interaktion zwischen den Akteuren. Providern dient es als Vorgabe des von ihnen erwarteten Verhaltens bei der Interaktion mit Assets. Clients leiten daraus die zu erwartende Leistung und die dazu notwendigen Fähigkeiten in Bezug auf das Interaktionsverhalten ihrer Demands ab. Daneben dient es bei verteilter Koordination den Koordinatoren als Grundlage zur Abstimmung ihrer Capabilities. Grundsätzlich besteht das Basisprinzip der Steuerung von Interaktionen zwischen den Services verschiedener Akteure darin, dass alle Kommunikationsvorgänge von einem dazwischen liegenden Capability-Service angestoßen werden. Das Interaktionsprotokoll kann also in Form einer Konversation des Capability-Service formuliert werden. Zur automatisierten Steuerung der Interaktion wird das Konversationsprotokoll als Kompositionsschema implementiert. Bei der Ausführung des Schemas werden die Kommunikationsvorgänge des Interaktionsprotokolls automatisch durchgeführt.

Auf Basis der Service-orient. Abbildung des FRESCO-VDLP werden auch die Akteure und Anwendungsfälle des konzeptionellen Dienstleistungsmodells im Sinne des Service-orient. Modells gedeutet. Abb. 4.15 zeigt die Zuordnung Service-orient. E-Service-Elemente zu den FRESCO-Akteuren. Dies erfolgt hier neben Client und Provi-

[32] Siehe Sektion 3.3.2.3.

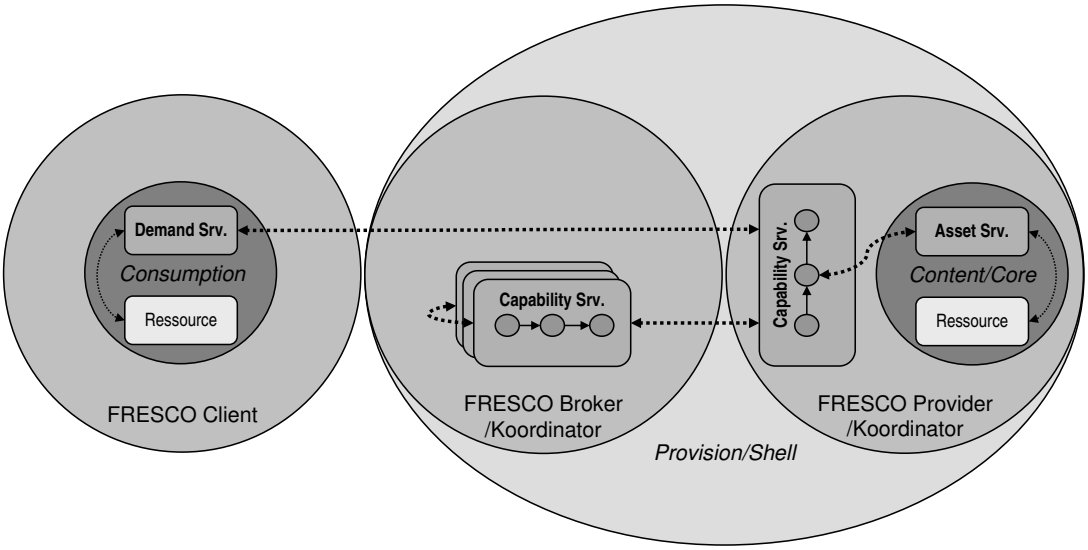

Abbildung 4.16.: VDLP-Struktur im FRESCO-SOM

der zunächst für die abstrakte Rolle des Koordinators. FRESCO-Provider entsprechen im Wesentlichen der Provider-Rolle des klassischen SOM. Die Funktionen der Rolle bestehen in Beschreibung und Publikation sowie in Implementierung und Betrieb von Asset-Services. FRESCO-Koordinatoren können hingegen mit der Aggregator-Rolle des erweiterten SOM identifiziert werden. Sie regeln und steuern die multilateralen Interaktionen von Capability, Asset und Demand Services. Dazu spezifizieren sie auf Basis von Asset- und Demand-Beschreibungen Interaktionsmuster und publizieren sie als Protokolle und Beschreibungen von Capability Services. In der Folge implementieren und betreiben sie Capability-Services als Kompositionsschemata. FRESCO-Clients sind schließlich ebenfalls im Wesentlichen als SOM-Provider zu klassifizieren. Sie beschreiben ihre Demands zur Nachfrage eines E-Service. Zum Konsum des E-Service implementieren und betreiben sie dann ihre Demand Services.

Die Akteure im FRESCO-spezifischen konzeptionellen Modell virt. Dienstleistungen sind in Bezug auf die oben beschriebenen Rollen und Funktionen leicht erweitert. Dabei ist die Rolle des Koordinators als abstrakt spezifiziert und tritt nur in verfeinerten Formen auf, die in Abb. 4.16 dargestellt werden. Die eine Form ist durch den FRESCO-Provider gegeben. Dieser übernimmt die grundlegenden Funktionen des Koordinators und spezialisiert sie für eine Teilaufgabe. Hierbei beschränkt sich die Regelung und Steuerung von Interaktionsmustern auf solche, die sich unmittelbar auf die Asset-Services des Providers beziehen. Entsprechende Capability Services besitzen ein Protokoll, das den Interaktionsablauf zwischen Asset Services und Demand Services zur Nutzung von Inhalten des Providers beschreibt. Zur Protokollspezifikation kann eine vorherige Suche nach Demands hilfreich sein. Die zweite Form des Koordinators ist der FRESCO-Broker. Dessen Teilaufgabe der Koordination bezieht sich auf Interak-

tionsmuster, die in Bezug zu anderen Capability Services stehen. Um entsprechende Capabilities zur Koordination verschiedener Provider eines ganzheitlichen E-Service zu spezifizieren, sucht er nach publizierten Capability und Demand Services. Entsprechend dem Demand kombiniert er dann die Capabilities von Providern passender Assets.

In der beschriebenen Art und Weise gibt das FRESCO-SOM ein fundamentales Konzept zur Service-orient. Realisierung des FRESCO-Dienstleistungsmodells vor. Dabei erfolgt die Realisierung von E-Services durch Abbildung von VDLP-Elementen auf verschiedenartige Service-Abstraktionen. E-Service-Management basiert auf einer Abbildung der Akteure und Methoden des VDL-Modells auf die Rollen und Funktionen des erweiterten SOM. Die Umsetzung des Realisierungskonzepts erfordert nun zunächst die Überführung der Service-orient. VDLP-Realisierung in eine Service-orient. Software-Architektur für E-Services. Auf dieser Basis erfolgt dann die Erweiterung von SOM-Mechanismen zu Service-orient. Entwicklungsmethoden für das E-Service-Management.

4.3.2.2. Das FRESCO E-Service-Metamodell

Das FRESCO-SOM gibt im Zuge der Service-orient. Abbildung des VDLP die grundlegenden Komponenten und Beziehungen einer Service-orient. Architektur für FRESCO E-Services vor. Es beschreibt jedoch lediglich grobe Kategorien und lässt viel Spielraum zur Ausgestaltung einer entsprechenden Software-Architektur. Im Folgenden wird das FRESCO-SOM durch ein Metamodell für E-Services konkretisiert. Dieses Metamodell definiert konkrete Komponenten und Beziehungen einer Service-orient. Software-Architektur. Mit ihrer Hilfe können Modelle von E-Services entworfen und implementiert werden.

Die Kategorien des FRESCO-SOM, die u. a. Beschreibung, Publikation, Koordination und Komposition von operationalen Schnittstellen, Interaktionsprotokollen und Kompositionsschemata umfassen, können zur Realisierung von E-Services durch verschiedene konkrete Techniken ausgefüllt werden. Dazu existieren im SOC-Kontext in der Regel jeweils verschiedene konkurrierende Lösungen, die sich oftmals rasch weiterentwickeln. Entsprechend konkretisierte Software-Architekturen sind an die Lebenszyklen einzelner Techniken gebunden und können ggf. schnell nutzlos werden.

In FRESCO wird daher ein generisches Architekturkonzept verfolgt, das sich im FRESCO E-Service-Metamodell (FSM) manifestiert. Das FSM modelliert die im FRESCO-SOM vorgesehenen Service-orient. Konzepte von Grund auf neu und eigenständig. Dies geschieht in einer Weise, die unabhängig von SOC-spezifischen Techniken ist. Stattdessen werden nur fundamentale Basiskonzepte verwendet. Im Wesentlichen sind das Elemente eines grundlegenden Prozessmetamodells. Mit ihrer Hilfe werden Metamodelle für die Komposition von Services durch Kompositionsschemata und die Koordination von Services durch Interaktionsprotokolle aufgestellt. Darauf setzt schließlich das Metamodell der FRESCO E-Service-Komponenten und ihrer Beziehungen auf.

Das Ziel der FSM-Konzeption ist es, den Service-orient. Entwurf konkreter E-Service-Modelle im Sinne des FRESCO-Dienstleistungsmodells zu ermöglichen. Dafür werden durch das FSM die notwendigen Kategorien von Modellelementen und -strukturen vorgegeben. Diese bauen auf den generischen Service-orient. Konzepten des FSM auf. Entsprechend lassen sich auch alle konkreten E-Service-Modelle, die auf Grundlage des FSM entworfen wurden, auf die Service-orient. Basiskonzepte zurückführen. Hierauf bauen im Folgenden verschiedene Service-orient. Entwicklungsmethoden für E-Services auf. Insbesondere können die Service-orient. Basiskonzepte des FSM dabei auch zu konkreten SOC-Techniken in Beziehung gesetzt werden. Auf Basis solcher Abbildungen lassen sich dann u. a. Methoden entwickeln, um die Implementierung von E-Service-Modellen in generativer Weise mittels Transformation automatisch durchzuführen. Solche Methoden werden im dritten Teil des Rahmenwerks konzipiert. Konkrete Beispiele zur Abbildung auf und Transformation in BPEL finden sich im nächsten Kapitel. Zuvor wird der Entwurf des FSM beschrieben.

Das FSM wird im weiteren Verlauf auf Basis von UML modelliert. Hierzu kommen im Wesentlichen Klassendiagramme zum Einsatz. Wegen der späteren Verwendung von FSM-Konzepten zur Beschreibung von E-Service-Modellen wäre an dieser Stelle auch eine formale Spezifikation der Semantik wünschenswert. Dies könnte im Hinblick auf das E-Service-Metamodell z. B. durch die Verknüpfung der FSM-Konzepte mit den Anwendungsfällen des konzeptionellen Dienstleistungsmodells mittels OCL-Constraints geschehen (denotionale Semantik). Da aber das dem FSM zugrunde liegende WfMC-Workflow-Prozess-Metamodell ebenfalls keine formale Semantik besitzt und daher keine durchgehende Spezifikation erfolgen könnte, wurde auf diesen Aspekt im Rahmen der vorliegenden Arbeit verzichtet. Stattdessen wird die Semantik des FSM informal beschrieben.

Abb. 4.17 zeigt ein UML-Diagramm, in dem eine Übersicht der Paketstruktur mit den wichtigsten Klassen dargestellt wird. Die Pakete des FSM beinhalten jeweils einen in sich geschlossenen Aspekt des Metamodells. Die Abhängigkeiten zwischen den Paketen zeigen dabei die Hierarchie der einzelnen Teilbereiche an. Die Basis des FSM wird durch ein Prozessmetamodell gebildet. Hierfür wird in FRESCO das Workflow-Metamodell der WfMC herangezogen (`workflow`). Dieses gliedert sich in verschiedene Teile auf, die Workflow-Prozesse (`process`) und deren Strukturierung in Pakete (`package`) definieren. Auf Basis von Workflow-Konzepten werden Konzepte Service-orient. Interaktionen formuliert. Hierzu führt das Metamodell der Web Service-Komposition (`composition`) u. a. die Begriffe atomarer und zusammengesetzter Web Services (im weiteren Sinn) ein. Auf diesen Konzepten setzt ein weiteres Service-orient. Metamodell der Web Service-Koordination (`coordination`) auf. Dieses nutzt Konzepte Service-orient. Interaktionsprozesse und spezialisiert diese im Sinne eines Interaktionsprotokolls. Am Ende werden die Service-orient. Konzepte angewendet, um im Rahmen des E-Service-Metamodells (`eServices`) die Realisierung des FRESCO-Dienstleistungsmodells zu spezifizieren. Ein wesentlicher Anteil dieses Metamodells betrifft die Realisierung von Capabilities und deren Kopplung. Schließlich wird

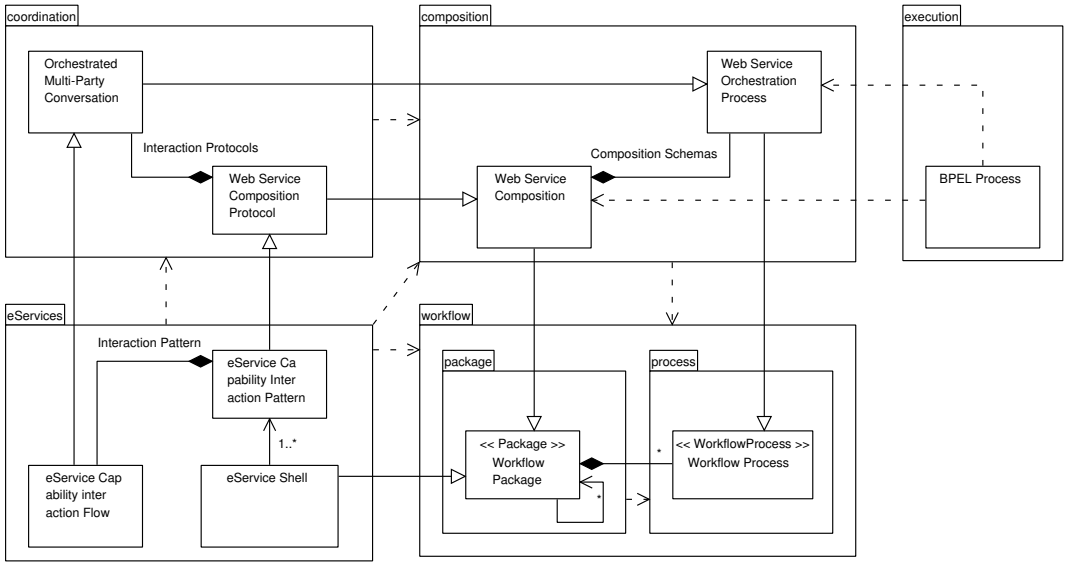

Abbildung 4.17.: Übersicht des FRESCO E-Service-Metamodell

noch grob die Beziehung zu konkreten Standards angedeutet. In FRESCO wird hier insbesondere BPEL zur Implementierung von E-Service-Modellen verwendet (`execution`).[33]

In den folgenden Sektionen werden die einzelnen Pakete des FSM im Detail beschrieben. Die Darstellung startet mit dem Workflow-Metamodell. Im Anschluss werden die Konzepte SO Interaktion und Interaktionsprozesse ausgeführt. Am Ende steht die Spezifikation des E-Service-Metamodells.

Workflow-Metamodell Die konzeptionelle Basis des FSM wird durch ein Metamodell gebildet, das grundlegende Workflow-Konzepte umfasst. Die unmittelbare Funktion dieses Modells besteht darin, dass es den fundamentalen organisationstheoretischen Begriff des Arbeitsprozesses in formaler Weise repräsentiert, bei dem Aktionsträger in inhaltlich verknüpften Aktivitäten einer geregelten Methode die zielgerichtete Transformation von Eingaben in Ausgaben vollziehen.[34] Deren formale Repräsentation als Workflow dient der Koordination von Aktionsträgern. Workflows regeln deren Abhängigkeiten bei der Ausführung von Aktivitäten in Bezug auf Kausalität und Daten. Derartige Regelungen spielen eine entscheidende Rolle auf allen Abstraktionsebenen des FSM. Zum einen sollen auf der Ebene der E-Services informationelle Administrationsprozesse im Kontext der Prozessorganisation von Dienstleistungen abgebildet werden. Dazu bilden die Workflow-Konzepte ein

[33] Die Beziehung zu BPEL soll an dieser Stelle nur am Rande Erwähnung finden. Später werden die Zusammenhänge noch ausführlich diskutiert.

[34] Siehe Sektion 2.3.1.1.

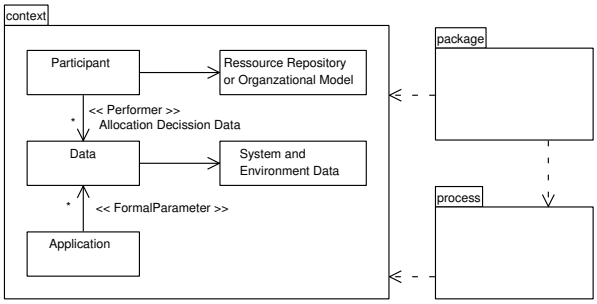

Abbildung 4.18.: Übersicht des Workflow-Metamodells

unmittelbares Instrument. Zum anderen liegt der Workflow-Begriff auch auf der Ebene Service-orient. Techniken sehr vielen Ansätzen der Service-Komposition und -Koordination zugrunde.

In Bezug auf konkrete Workflow-Konzepte besteht in Forschung und Technik keine Einigkeit. Obwohl die grundsätzlichen Bestandteile und Aspekte von Workflows intuitiv gut verstanden sind, werden sie in einer großen Menge individueller Ansätze auf eigenständige Art und Weise ausgelegt. Dies stellt im Kontext von E-Service ein gewisses Problem dar, denn als Technik der IV-Infrastruktur von virt. Dienstleistungsproduktionsnetzwerken ist Standardkonformität ein entscheidender Aspekt. Hier müssen die Konzepte von allen autonomen Produktionseinheiten verstanden und umgesetzt werden können. Ein weiteres Problem betrifft die Spezialisierung von Workflow-Konzepten zu Konzepten der Service-Komposition und -Koordination. Die Abbildung der Service-orient. Konzepte auf eine möglichst breite Palette konkreter SOC-Techniken wird durch exotische Workflow-Konzepte erschwert. Aus diesem Grund wird als Basis der im FSM definierten Workflow-Konzepte das *Workflow Process Metamodel* [WfM02] aus dem *Workflow Reference Model* der *Workflow Management Coalition* herangezogen. Die hierin beschriebenen Konzepte zeichnen sich durch ihre Standardisierung und Generik aus. Die WfMC fasst als Konsortium die wichtigsten Hersteller und Nutzer von Workflow-Techniken zusammen. Das Workflow Reference Model repräsentiert das Ergebnis dieser Zusammenarbeit. Das Modell beinhaltet den kleinsten gemeinsamen Nenner akzeptierter und verstandener Konzepte. Aus diesem Grund besitzt das Modell beste Voraussetzungen für den Einsatz in virt. Unternehmen.

Die WfMC selbst spezifiziert das Workflow-Prozess-Metamodell in verbaler Form und benutzt keine formale Notation. Zur Spezifikation des FSM wird das Workflow-Metamodell in UML überführt. Abb. 4.18 zeigt das Resultat im Überblick. Es besteht aus drei Paketen, die einen allgemeinen Kontext (`context`), Workflow-Prozesse (`process`) und deren Zusammenfassung in Paketen (`package`) beschreiben. Der Workflow-Kontext beschreibt grundlegende Konzepte, auf denen Workflow-Prozesse aufsetzen. Dazu gehören Teilnehmer (`Participant`), Applikation (`Application`) und Daten (`Data`) als Re-

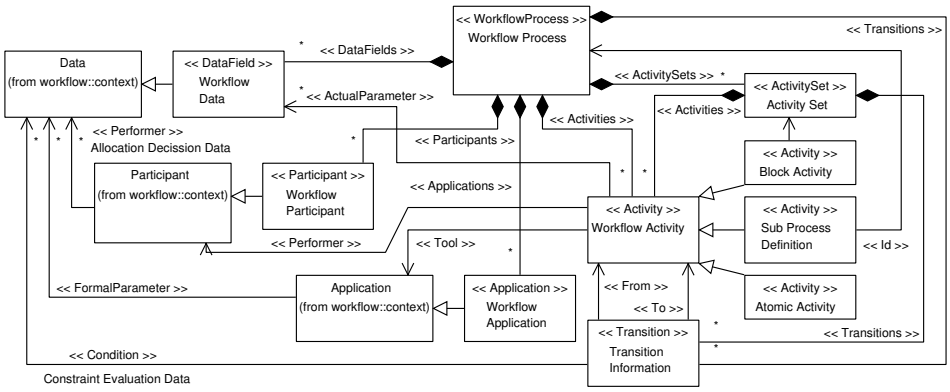

Abbildung 4.19.: Workflow Process Metamodel

präsentationen der organisatorischen Konzepte von Aktionsträgern, Arbeitsmethoden und Informationen. Applikationen können dabei als Anwendungssysteme ausgelegt werden. Um die Einbettung in den organisatorischen Kontext anzudeuten, werden für Daten und Teilnehmer entsprechende abstrakte Bezugsmodelle definiert (`System and Environment Data` bzw. `Ressource Repository or Organizational Model`). Daten dienen als Eingabe für Applikationen und zur Allokation von Teilnehmern.

Der Kern des Workflow-Metamodells ist durch das Workflow-Prozess-Metamodell gegeben, das in Abb. 4.19 gezeigt wird. Das zentrale Konzept ist hier der Workflow-Prozess (`Workflow Process`). Dieser beinhaltet im Wesentlichen Aktivitäten und Transitionen sowie Deklarationen von einbezogenen Applikationen, Teilnehmern und Datenelementen. Aktivitäten (`Workflow Activity`) repräsentieren eine Arbeitseinheit. Diese kann wiederum andere Aktivitäten oder ganze Workflow-Prozesse beinhalten oder in elementarer Form von einem Teilnehmer mittels eines Anwendungssystems erbracht werden. Transitionen (`Transition Information`) spezifizieren die Abfolge von Aktivitäten, d. h. dass das Prozessmodell auf einer Graphstruktur aus Knoten (Aktivitäten) und Kanten (Transitionen) basiert. Deklarationen von Applikationen (`Workflow Application`), Teilnehmern (`Workflow Participant`) und Daten (`Workflow Data`) definieren die Bestandteile elementarer Aktivitäten. Neben der Funktion von Datenelementen als Eingabe für Applikationen dienen diese zur Steuerung des Ablaufs durch Bedingungen von Transitionen und zur Auswahl von Teilnehmern zur Durchführung von Aktivitäten.

Das zugehörige Metamodell für Workflow-Pakete (Abb. 4.20) definiert strukturelle Elemente zur Gliederung komplexer Workflow-Definitionen. Zentrales Konzept ist hier das Workflow-Paket (`Workflow Package`). Workflow-Pakete fassen verschiedene Workflow-Prozesse sowie global verwendete Applikationen (`Workflow Application`), Teilnehmer (`Workflow Participant`), Datendefinitionen (`Workflow Data Types`) und organisatorische Bezugsmodelle zusammen. Zudem können Workflow-Pakete wechselseitig aufeinander Bezug nehmen. So wird die einheitliche Definition und Strukturierung

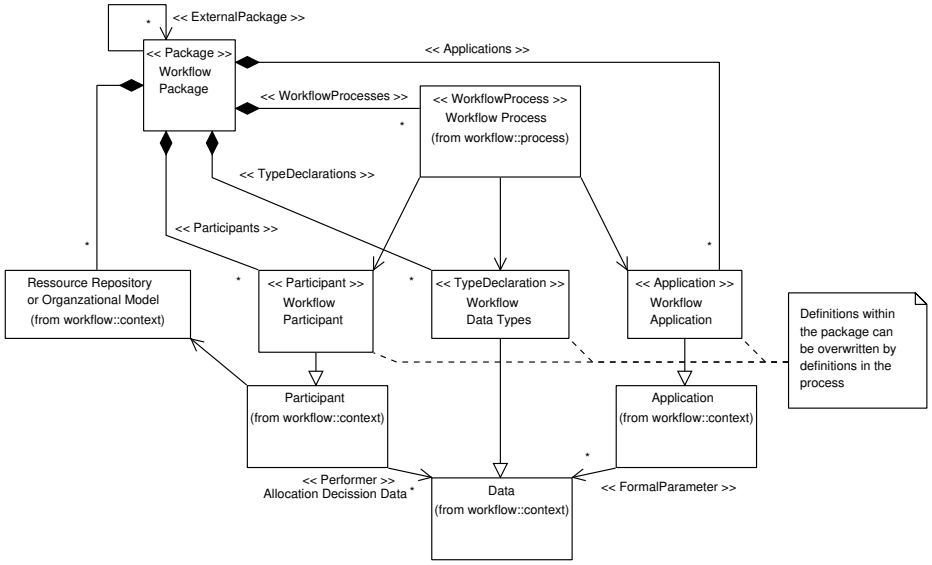

Abbildung 4.20.: Workflow Package Metamodel

von komplexen Anwendungen mit einer Vielzahl verschiedener Workflow-Prozesse erleichtert.

Insgesamt bildet das generische Workflow-Prozess-Metamodell die Grundlage, um im Folgenden verschiedene spezielle Formen von Arbeitsprozessen zu regeln. Zum einen sind das Kompositionsprozesse zur Realisierung zusammengesetzter Services, bei denen der Zugriff auf Service-Komponenten geregelt werden muss. Zum anderen sind das die Fähigkeiten (Capabilities) einer virt. Dienstleistung zur informationellen Administration bei der Nutzung von Dienstleistungsinhalten, bei denen die Interaktion mit Akteuren eines VDPN geregelt werden muss. Die Strukturelemente des Workflow-Paket-Metamodells bilden vor allem auf der Ebene der E-Services eine Grundlage zur Modellierung komplexer Strukturen, bei denen verschiedene zusammenhängende Arbeitsprozesse berücksichtigt werden müssen.

Web Service-Interaktion Metamodell Die nächste Stufe des FSM definiert grundlegende Konzepte Service-orient. Interaktion. Hier werden diejenigen Konzepte im FSM eingeführt, die im FRESCO-SOM als Grundlage zur Realisierung von E-Services bestimmt wurden. Im Einzelnen sind das Web Services-Abstraktionen, WS-Interaktionsprotokolle und WS-Kompositionsschemata. Web Services-Abstraktionen realisieren im FRESCO-SOM die Assets und Demands von Clients und Providern als Kommunikationsendpunkte. Sie dienen dabei zur Spezifikation und Durchführung einer zusammenhängenden Menge bilateraler Interaktionen. Zur Realisierung virt. Dienstleistungsprozesse ist darüber hinaus die Planung und Steuerung von Interaktionsabläufen von Interesse. Dieser Aspekt wird im FRESCO-Dienstleistungsmodell

durch Capabilities repräsentiert und im FRESCO-SOM durch zusammengesetzte Web Services als Interaktionsmuster umgesetzt. Bei dieser Strategie werden zwei komplementäre Sichtweisen auf Service-orient. Interaktionsprozesse kombiniert. Zum einen kann die Spezifikation des Ablaufs von Interaktionen mit Web Services als Protokoll dienen, das zur Koordination beteiligter WS-Clients und -Provider dient. Zum anderen kann eine solche Spezifikation als Anweisung zur automatischen Durchführung von Interaktionen verwendet werden. Wenn in letzterem Fall eine Festlegung von Interaktionspartnern als Client und Provider vorliegt, wird dadurch ein zusammengesetzter Web Service definiert. Zusammengesetzte Capability-WS dienen im FRESCO-SOM gleichermaßen als Koordinationsprotokolle zur Regelung und als Kompositionsschema zur Steuerung von Interaktionsmustern zwischen den Kommunikationsendpunkten von Asset- und Demand-WS. In diesem Sinne wird im FSM das Konzept des Service-orient. Interaktionsprozesses definiert und in subsequenten Verfeinerungen zunächst als Kompositionsschema (Abb. 4.21) und dann als Koordinationsprotokoll (Abb. 4.22) ausgelegt. Dies geschieht in Form der Spezialisierung von Arbeitsprozessen bzw. Workflows. Nachfolgend wird die Modellierung beider Konzepte einzeln dargestellt.

Das fundamentale Konzept Service-orient. Interaktion ist im FSM die WS-Komposition. Als Grundlage hierzu werden Service-orient. Interaktionsprozesse als Verfeinerung von Workflow-Prozessen spezifiziert. Abb. 4.21 zeigt dies als UML-Diagramm. Auf der rechten Seite des Diagramms sind die verwendeten Superklassen des Workflow-Metamodells zu sehen. Auf der linken Seite sind deren Verfeinerungen als Teilkonzepte der WS-Komposition dargestellt. Generell sind die wesentlichen Bestandteile eines Service-orient. Kompositionsmodells durch Komponenten-, Koordinations- und Aggregationsmodell gegeben, wobei sich das Koordinationsmodell weiter in die Aspekte von Orchestrierungs- und Datenmodell teilt.[35] All diese Bereiche werden im FSM auf Workflow-Konzepte zurückgeführt.

In Bezug auf das Komponentenmodell sieht das FSM grundsätzlich Web Services als Komponenten vor. Diese werden auf Basis von Workflow-Applikationen eingeführt. Eine `Workflow Application` wird dabei als einzelne `Web Service Operation` spezialisiert. Hierzu wird sie durch eine Referenz (`Web Service Description Reference`) auf die Spezifikation einer Operation (`WSDL Port/Opertion Reference`) als Teil einer Service-Beschreibung (`WSDL Document Reference`) ergänzt. In diesem Sinne wird für WS-Komponenten eine Service-Beschreibung in Form von operationalen Schnittstellen vorausgesetzt. In der Regel ist das WSDL. Bei der Komposition führen Teilnehmer dann auf Basis von WS-Operationen Interaktionen durch. Diese Teilnehmer werden im Kompositionsschema als Rollen abstrahiert. Die verschiedenen WS-Operationen einer Rolle definieren deren kompletten Web Service. Hierbei muss grundsätzlich zwischen WS-Komponenten und dem resultierenden zusammengesetzten Web Service unterschieden werden. WS-Komponenten werden von spezifischen Kompositionsschemata getrennt spezifiziert. Dazu wird ein verfeinertes Workflow-Paket als wieder

[35] Siehe Sektion 3.4.2.3.

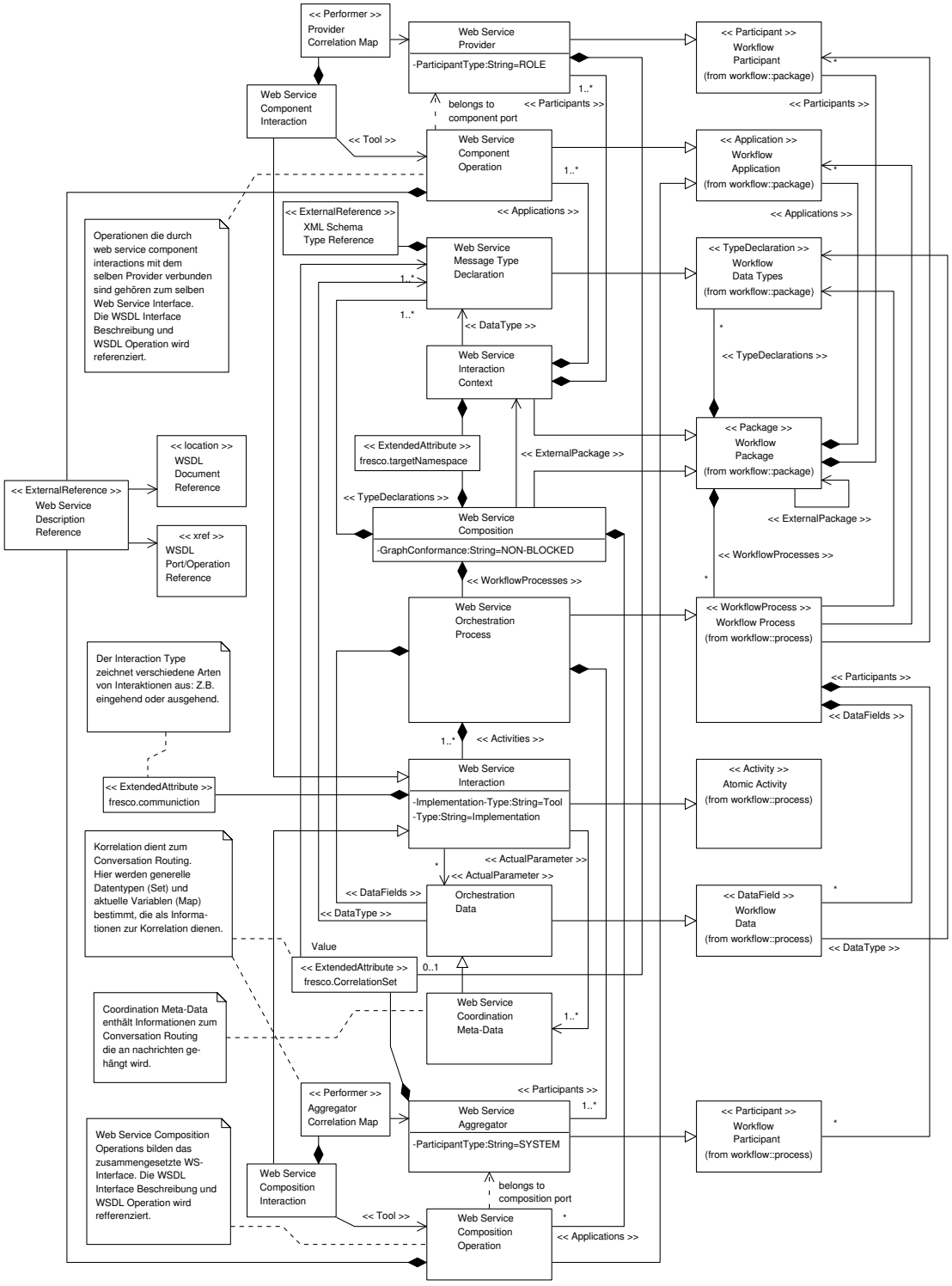

Abbildung 4.21.: FRESCO Service Composition Metamodel

verwendbarer Kontext allgemeiner WS-basierter Interaktionen definiert (`Web Service Interaction Context`). Hierin werden dann Operationen (`Web Service Component Operation`) und Rollen (`Web Service Provider`) von Komponenten spezifiziert, die paarweise jeweils Interaktionen (`Web Service Component Interaction`) zum Aufruf im Rahmen verschiedener Kompositionen bilden. Des Weiteren wird im Interaktionskontext das Datenmodell definiert. Hierzu werden Workflow-Datendefinitionen zu `Web Service Message Type Declarations` verfeinert. Diese spezifizieren Dokumente, die im Zuge von WS-Interaktionen ausgetauscht werden. Die Beschreibung von Nachrichtentypen erfolgt durch Referenz auf externe XML-Schema-Spezifikationen (`XML Schema Type Reference`). Der aus einer Komposition hervorgehende Web Service wird als Teil eines entsprechenden Kompositionsschemas spezifiziert. Ein Kompositionsschema (`Web Service Composition`) ist ein spezielles Workflow-Paket, das als wichtigsten Bestandteil den `Web Service Orchestration Process` als spezielle Form der Workflow-Prozess-Definition beinhaltet. Im Kompositionsschema werden die Operationen des zusammengesetzten Web Service (`Web Service Composition Operation`) spezifiziert. Sie bilden Interaktionen (`Web Service Composition Interaction`), die einem Teilnehmer in der Rolle des Aggregators (`Web Service Aggregator`) obliegen.

In Bezug auf das Koordinationsmodell erbt das FSM den graphbasierten Ansatz des Workflow-Prozess-Konzepts. Dieser wird jedoch so eingeschränkt, dass keine Zyklen erlaubt sind (`NON-BLOCKED`). Der Orchestrierungsprozess des Kompositionsschemas spezifiziert dann Interaktionen mit Komponenten-Providern (`Web Service Component Interaction`) und Clients des zusammengesetzten Web Service (`Web Service Composition Interaction`). Die beiden Interaktionsvarianten sind spezielle Formen einer Interaktion im Rahmen des Orchestrierungsprozesses (`Web Service Interaction`). Diese stellt eine spezielle Form der Workflow-Aktivität dar, bei der die Application ein Web Service ist. In WfMC-Terminologie bilden WS hierbei ein `Tool`. Daneben wird hier noch die Richtung einer Interaktion definiert. Dies geschieht durch ein spezielles Attribut (`fresco.communication`). In Bezug auf Interaktionen ist außerdem eine spezielle Variante der Workflow-Datenelemente (`Orchestration Data`) vorgesehen. Dies sind Prozessvariablen, die die kommunizierten Nachrichten beinhalten. Alle anderen Aspekte der Workflow-Prozess-Definition werden für Kompositionsschemata übernommen. Das gilt insbesondere für Transitionen zur Spezifikation des Interaktionsverlaufs und die damit zusammenhängenden Konzepte.

In Bezug auf das Aggregationsmodell besteht das Grundprinzip darin, den Aggregationsaspekt so weit wie möglich vom Kompositionsmodell zu trennen. Weder im Kompositionsschema noch im -kontext sind Konzepte zur Spezifikation konkreter Teilnehmer oder Ressourcen vorhanden, mit denen via Web Services Interaktionen im Sinne der Komposition durchgeführt werden sollen. Genauso wenig gibt es Konzepte zur Spezifikation einer Allokationsstrategie. Dies ist dadurch motiviert, dass die Allokation ein gesondertes und wenig strukturiertes Problem erheblicher Komplexität darstellt. Dieses sollte daher sin separater Weise behandelt werden. In diesem Zusammenhang sind jedoch im Service-orient. Kontext noch weitere Konzepte zu

berücksichtigen. Hier muss bei Konversationen, die mehrere Interaktionen beinhalten, sichergestellt werden, dass alle Interaktionen an den gleichen Teilnehmer gehen und dieser auch mehrere parallel geführte Konversationen unterscheiden kann. Dabei soll jedoch keine zu feste und im organisationsübergreifenden Kontext potenziell instabile Bindung hergestellt werden. Die Lösung im Service-orient. Kontext besteht hier in der Regel in der Korrelation von Konversationen mit Teilen der ausgetauschten Nachrichten. Im FSM wird daher das Konzept des `Correlation Set` eingeführt. Dadurch werden diejenigen Informationen bestimmt, anhand derer die Teilnehmer einer Komposition vom Aggregator identifiziert werden sollen. Anders herum werden spezifische Datenelemente (`Web Service Correlation Metadata`) zur Kennzeichnung der zusammenhängenden Interaktionen einer Komposition als administrative Metadaten von Nachrichten eingeführt und mithilfe eines spezifischen Ausdrucks (`Provider Correlation Map`) unter den Parametern einer Interaktion gekennzeichnet. Auf dieser Basis können Client und Komponenten-Provider zusammenhängende Interaktionen einer Konversation mit dem Aggregator erkennen und ein entsprechendes Conversation Routing durchführen.[36]

Das Metamodell der WS-Komposition fasst Workflow-Prozesse als Interaktionsprozesse mittels Web Services auf. Ein Kompositionsschema dient als Anweisung für einen Aggregator, um Web Services von Teilnehmern in verschiedenen Rollen aufzurufen und im Gegenzug Aufrufe von diesen über einen eigenen Web Service entgegenzunehmen. Der Aggregator führt also eine multilaterale Konversation mit verschiedenen Teilnehmern durch. Eine Voraussetzung dafür ist die Einigkeit aller Konversationspartner über den Ablauf der Interaktionen. Der Ablauf muss dementsprechend zwischen ihnen geregelt werden. Diese Aufgabe wird von Interaktionsprotokollen übernommen. Im FSM werden diese Protokolle durch ein Metamodell für WS-Koordination (`coordination`) festgelegt. Die Koordination ist dabei nicht allgemeingültig für beliebige Interaktionsabläufe ausgelegt, sondern bezieht sich direkt auf solche, die im Verlauf eines Kompositionsprozesses nach dem FSM Metamodell für WS-Kompositionen ablaufen. Dementsprechend werden die Elemente eines Interaktionsprotokolls auf die entsprechenden Konzepte von Kompositionsprozessen zurückgeführt.

Abb. 4.22 zeigt ein Diagramm, das die Zusammenhänge zwischen Kompositionsprozessen und Koordinationsprotokollen wiedergibt. Im oberen Teil sind die hier relevanten Konzepte der WS-Komposition dargestellt. Im unteren Teil befinden sich die davon abgeleiteten Konzepte der WS-Koordination. Das Grundprinzip zeigt sich in der Verfeinerung von Orchestrierungsprozessen als multilaterale Konversation (`Orchestrated Multi-Party Conversation`). Dabei werden komplexe Interaktionsprozesse einer Menge verschiedener Teilnehmer auf Basis von deren einzelnen Konversationen mit einer zentralen Orchestrierungsinstanz definiert. In der Folge werden die kompositionsspezifischen Konzepte Service-orient. Interaktionsprozesse im Sinne eines dazu in Beziehung stehenden Koordinationsprotokolls spezialisiert. Auf oberster Ebene

[36] Siehe Sektion 3.4.2.2.

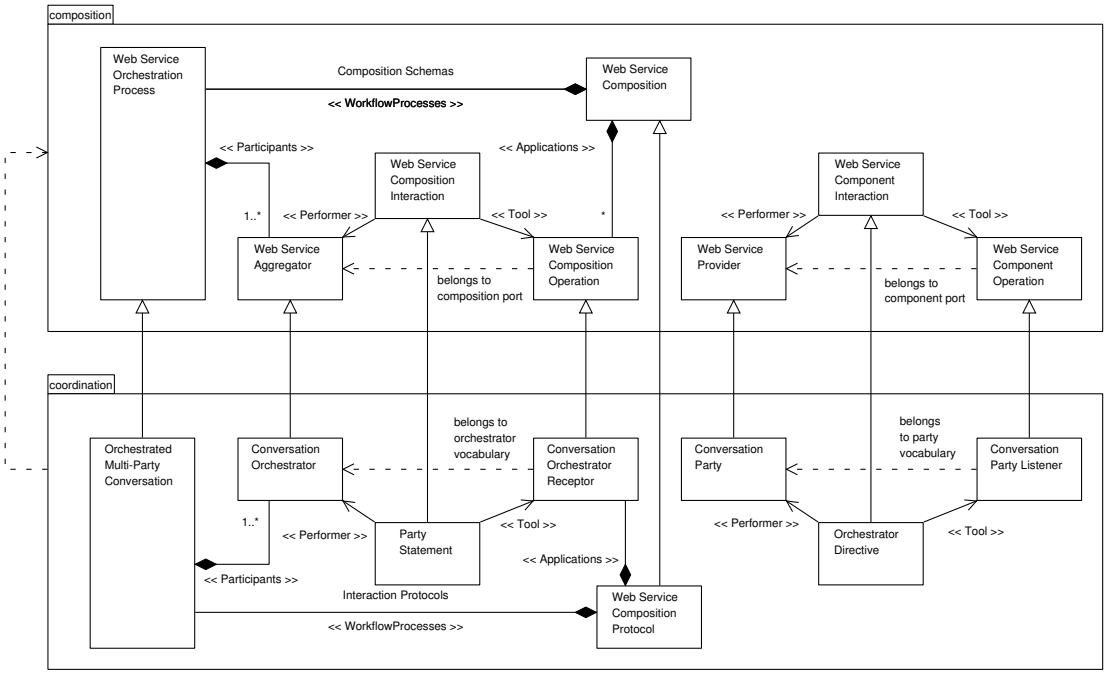

Abbildung 4.22.: FRESCO Service Coordination Metamodel

wird aus dem Konzept der Komposition das des kompositionsspezifischen Protokolls abgeleitet (`Web Service Composition Protocol`). Dessen wichtigster Teil ist die schon angesprochene multilaterale Konversation. Die Teilnehmer der Konversation sind zum einen die interagierenden Parteien (`Conversation Party`) als spezielle Form von WS-Providern. Diese können im Rahmen der Konversation über spezifische Kommunikationsendpunkte (`Conversation Party Listener`) angesprochen werden, die spezielle WS-Operationen von WS-Komponenten darstellen. Zum anderen ist an der Konversation ein zentraler Orchestrator (`Conversation Orchestrator`) als spezielle Form des Aggregators beteiligt. Dieser verfolgt die Konversation über Kommunikationsendpunkte (Conversation Orchestrator Receptor), die von den Operationen zusammengesetzter WS abgeleitet sind. Interaktionen im Rahmen einer Konversation gehen nun entweder von den Konversationsteilnehmern an den Kommunikationsendpunkt des Orchestrators (`Party Statement`) oder vom Orchestrator an den Kommunikationsendpunkt eines Teilnehmers (`Orchestrator Directive`).

Die Konzepte Service-orient. Interaktion werden nun zu E-Service-Konzepten verfeinert. Sie dienen zur Service-orient. Interpretation der anwendungsnahen Konzepte, die sich primär auf die Basiskonzepte der Arbeitsprozesse beziehen, d. h. dass E-Services sich auf die fundamentalen Konzepte von Arbeitsprozessen bzw. Workflows stützen, ihre Realisierung jedoch auf Basis der Beziehungen im FSM durch Service-orient. Techniken erfolgt.

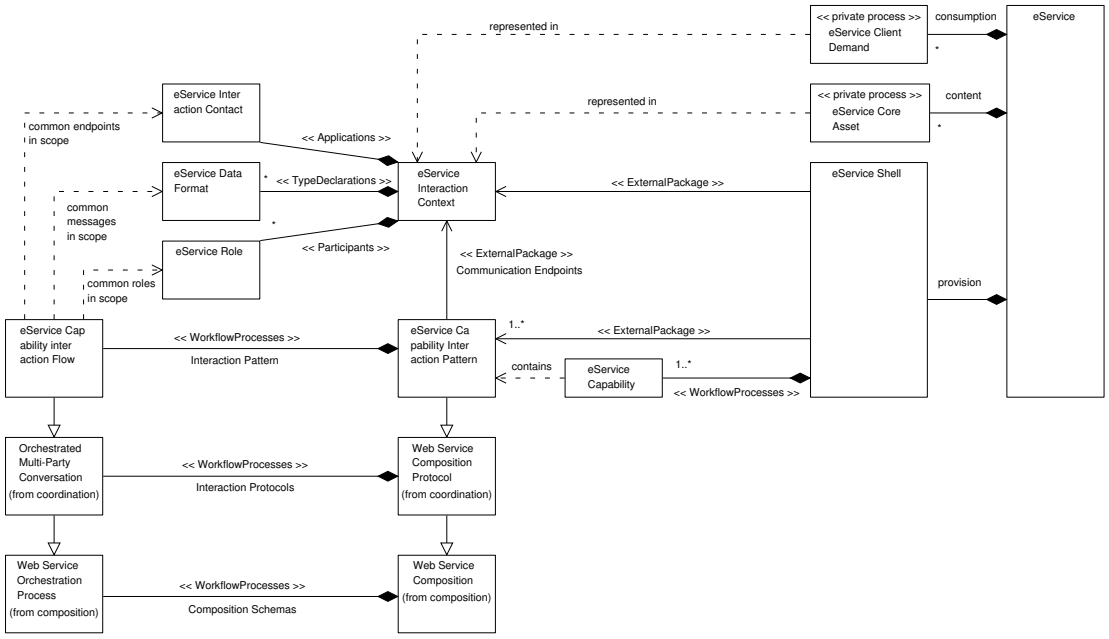

Abbildung 4.23.: FRESCO E-Service-Metamodel

E-Service-Metamodell Die Konzepte des FRESCO-Dienstleistungsmodells werden im E-Service-Metamodell des FSM mit den grundlegenden Konzepten von Arbeitsprozessen und deren Service-orient. Spezialisierungen in Beziehung gesetzt.

Das Diagramm in Abb. 4.23 zeigt zunächst eine Übersicht des E-Service-Konzepts mit den grundlegenden Strukturelementen des Metamodells. Dies ist zunächst der `eService` selbst, der den virt. Dienstleistungsprozess aus dem FRESCO-Dienstleistungsmodell repräsentiert. Der `eService` ist im Sinne des Dienstleistungsmodells in die drei Bereiche Content, Provision und Consumption aufgegliedert. Da das FRESCO-Modell grundsätzlich Erbringer-orientiert ausgelegt ist, liegt der Schwerpunkt auf den entsprechenden Aspekten, die sich im Wesentlichen in der Shell (`eService Shell`) und den verschiedenen Interaktionsmustern (`eService Capability Interaction Pattern`) im Rahmen jeweils einer Capability (`eService Capability`) widerspiegeln. Die Aspekte von Produktionseinheiten, die einzelne Inhalte als Assets (`eService Asset`) einbringen und von Kunden, die deren Konsum in Form von Demands (`eService Demand`) dagegenhalten, werden hier aus der Perspektive der Erbringung dargestellt. Aus dieser Perspektive steht die Interaktion mit Produktionseinheiten und Kunden im Rahmen der Nutzung von Assets durch Demands im Vordergrund. Aus diesem Grunde bilden Rollen (`eService Role`), Kommunikationsendpunkte (`eService Interaction Contact`) und Nachrichtenformate (`eService Data Format`) aller am E-Service beteiligten Assets und Demands einen gemeinsamen Kontext (`eService Interaction Context`), auf den alle Interaktionsmuster aller Capabilities der Shell in einheitlicher

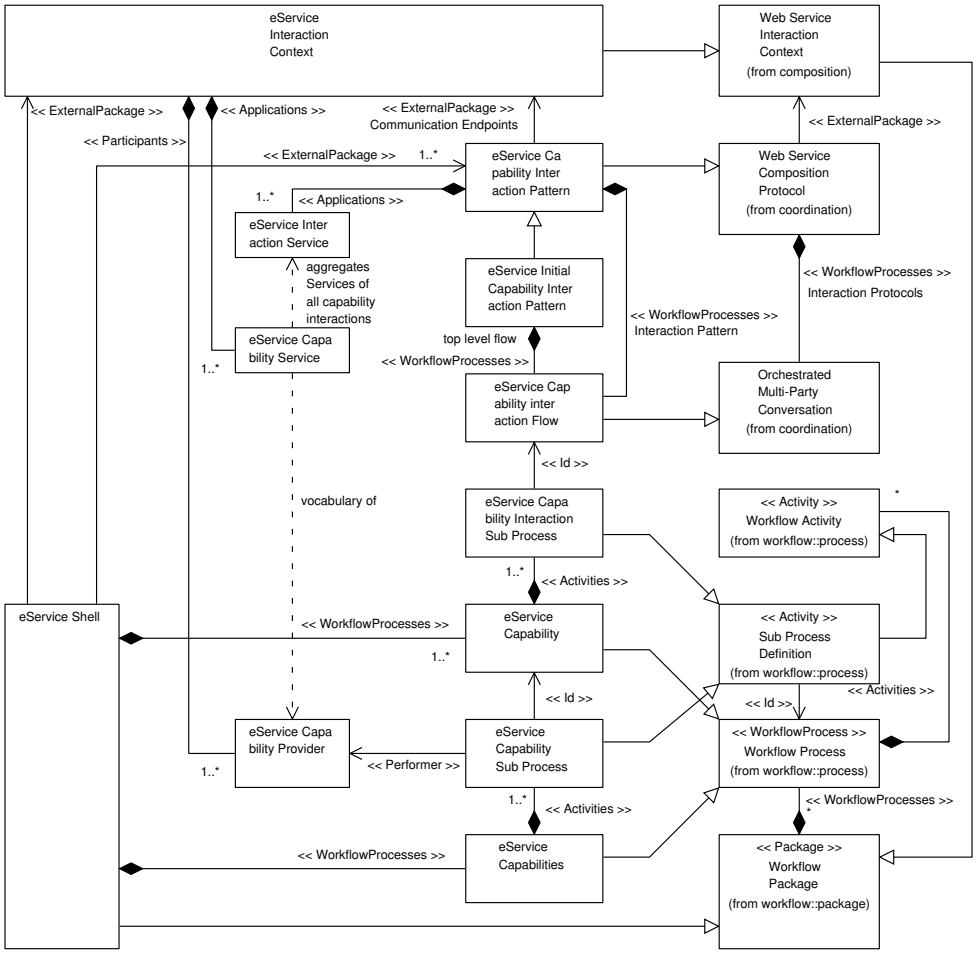

Abbildung 4.24.: FRESCO E-Service-Shell-Metamodel

Weise zurückgreifen. Interaktionsmuster werden dabei als spezielle Form eines Interaktionsprotokolls modelliert und beinhalten als wesentlichen Bestandteil einen Interaktionsprozess (`eService Capability Interaction Flow`) zwischen den Kommunikationsendpunkten von E-Service-Teilnehmern. Damit erben Interaktionsmuster gleichzeitig die Eigenschaften von Kompositionsschemata und bilden eine spezielle Form allg. Arbeitsprozesse.

Bevor auf die Funktionsweise von Interaktionsprozessen näher eingegangen wird, soll zunächst die Modellierung der Shell betrachtet werden. Diese wird in Abb. 4.24 dargestellt. Im rechten Teil der Darstellung sind Konzepte verschiedener Abstraktionsebenen zu sehen, auf denen die Modellierung der Shell aufsetzt. Die `eService Shell` selbst ist als Verfeinerung eines `Workflow Package` ausgelegt. Sie referenziert zum einen den `eService Interaction Context`, der die Kommunikationsendpunkte von Akteuren des VDLP beinhaltet. Zum anderen verweist sie auf `eService Capability Interaction`

Patterns als Interaktionsmuster des VDLP. Die wesentliche Funktion der Shell besteht in der Strukturierung von Interaktionsmustern des E-Service. Hierzu beinhaltet sie zwei spezialisierte Arten von Workflow-Prozessen. Zum einen beinhaltet die Shell Repräsentationen aller Capabilities eines E-Service. Diese Repräsentationen sind als einfache Workflow-Prozesse modelliert (eService Capability). Solche Prozesse bestehen lediglich aus speziellen Sub-Prozess-Aktivitäten, die die Interaktionsprozesse von Interaktionsmustern der Capability referenzieren (eService Capability Interaction Sub Process). Zum anderen beinhaltet die Shell den speziellen Workflow-Prozess eService Capabilities. Dieser fasst alle Capability-Repräsentationen zusammen, indem er sie durch Sub-Prozess-Aktivitäten (eService Capability Sub Process) referenziert. Eine dieser Referenzen wird als initialer Steuerprozess ausgezeichnet. Die so bestimmte Capability beinhaltet genau ein Interaktionsmuster (eService Initial Capability Interaction Pattern), das am Anfang der Kontaktphase einer virt. Dienstleistung aktiv wird und mit der Steuerung des VDLP beginnt. Ferner ordnen die Referenzen den Capabilities jeweils eine Rolle (eService Capability Service) zu. Die Rolle geht mit der Verpflichtung einher, die Kontaktpunkte aller Interaktionsmuster (eService Interaction Services) als (eService Capability Services) in einer einheitlichen Schnittstelle zusammenzufassen. So wird bewirkt, dass die Interaktionsmuster einer Capability als Einheit von genau einem VDPN-Teilnehmer erbracht werden.

Während die Shell im Wesentlichen der Strukturierung von VDLP-Elementen dient, verkörpern die Interaktionsmuster einer Capability die Logik der virtuellen Dienstleistungserbringung. Das entsprechende Metamodell wird in Abb. 4.25 gezeigt. In den vorangegangenen Diagrammen wurde schon beschrieben, dass eService Capability Interaction Patterns als Spezialisierung Service-orient. Interaktionsprotokolle modelliert werden. Sie sind in Form von Repräsentationen (eService Capabilities) in der eService Shell registriert. Ihr wesentlicher Bestandteil ist ein Interaktionsprozess (eService Capability Interaction Flow), der vom Konversationsprozess (Orchestrated Multi-Party Conversation) der Protokollebene abgeleitet wird. Zur Modellierung von E-Service-Interaktionsmustern werden Elemente der Service-orient. Ebene im Sinne virt. Dienstleistungsprozesse spezialisiert.

Auf der Ebene der E-Services beschreiben Interaktionsmuster den Interaktionsablauf zwischen Providern, Kunde und einem Koordinator. Der Koordinator tritt hier als regulierende und steuernde Instanz zwischen den Akteuren auf. Aus diesem Grund wird er durch das Konzept des eService Interaction Coordinator als Verfeinerung des Conversation Orchestrator modelliert. Die Kommunikationsendpunkte des Koordinators werden als eService Interaction Services herausgestellt. Über diese Services können die verschiedenen Akteure über jeweils spezifische Arten der Interaktion mit dem Koordinator in Kontakt treten: Interaktionen zwischen Kunden und Koordinator (eService Client Statement) zwischen Providern und Koordinator (eService Provider Statement) sowie zwischen Koordinatoren und Providern anderer Capabilities (eService Inter-Capability Takeover). Der letzte Fall tritt bei Komposition von Capabilities auf und wird am Ende dieses Abschnitts genauer dargestellt.

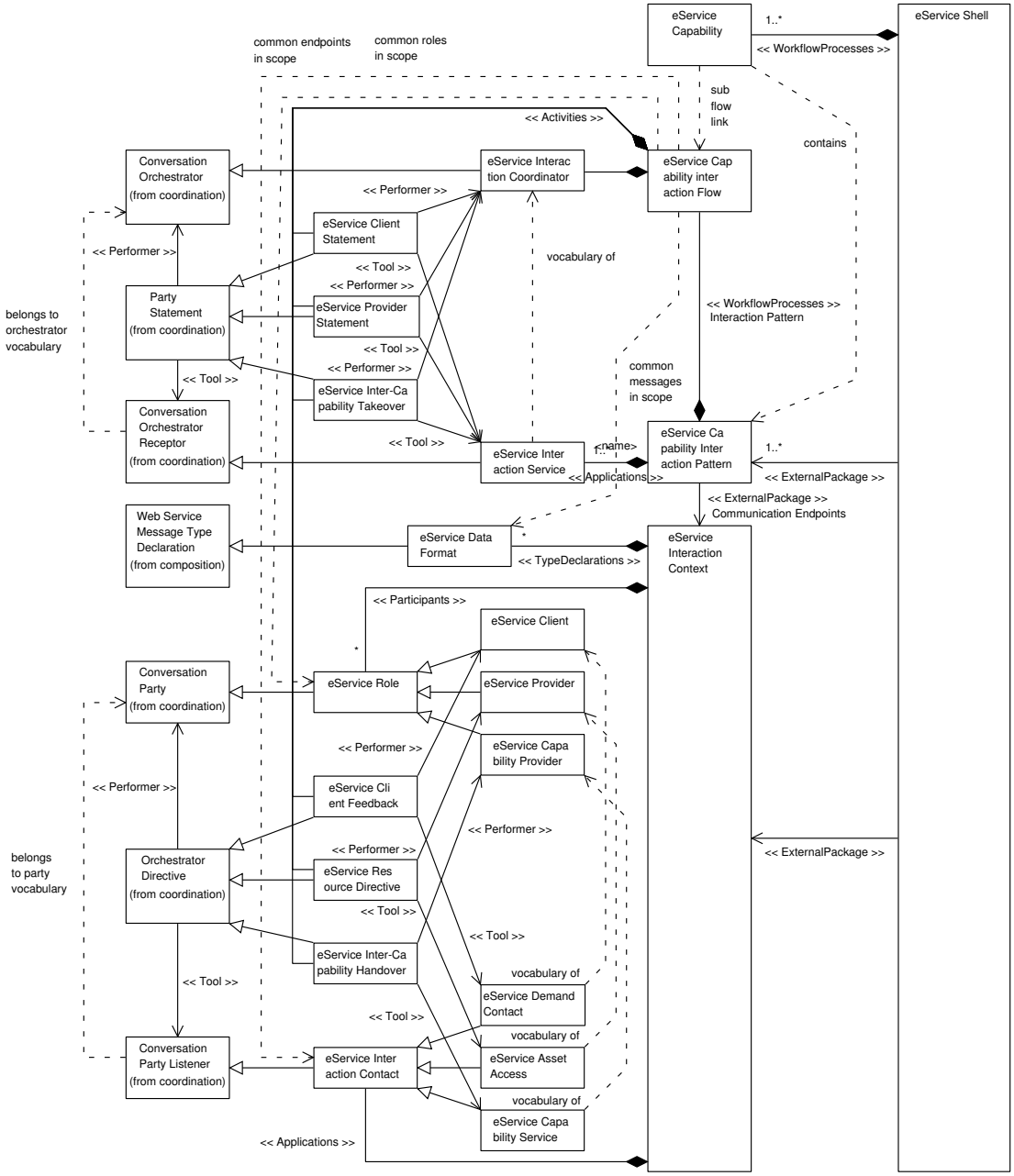

Abbildung 4.25.: FRESCO E-Service Capability-Metamodel

Die Akteure eines Interaktionsmusters – seien dies Kunden mit Demands, Provider mit Assets oder Provider einer assoziierten Capability – werden über ihre Kommunikationsendpunkte in den Prozess eingebunden. Diese Endpunkte werden im `eService Interaction Context` festgelegt, der den allg. `Web Service Interaction Context` erweitert. Hierin modelliert das Konzept des `eService Data Format` kommunizierte Inhalte der Anwendungsebene als spezielle Form der `Web Service Message Type Declaration`. Des Weiteren werden die Parteien und Kommunikationsendpunkte einer Konversation zu den speziellen Rollen und Kontaktpunkten von E-Services erweitert. Allgemein wird die Partei einer Konversation zunächst zur abstrakten Rolle eines E-Services (`eService Role`) spezialisiert. Der Kommunikationsendpunkt einer Konversation wird dementsprechend als abstrakter Kontaktpunkt für E-Service-Interaktionen (`eService Interaction Contact`) verfeinert. Entsprechend den drei Arten von Akteuren werden entsprechende Rollen (`eService Client`, `eService Provider` und `eService Capability Provider`) sowie Kontaktpunkte (`eService Demand Contact`, `eService Asset Access` und `eService Capability Service`) abgeleitet. Auf Basis dieser Konzepte werden nun drei Arten von Interaktionen mit Akteuren unterschieden, die vom Koordinator im Rahmen des Interaktionsmusters durchgeführt werden können: Interaktionen zwischen Koordinator und den Demands von Kunden (`eService Client Feedback`), Interaktionen zwischen Koordinator und den Assets von Providern (`eService Ressource Directive`) sowie Interaktionen zwischen dem Koordinator und dem Provider einer assoziierten Capability (`eService Inter-Capability Handover`).

Am Ende stellen sich E-Service-Interaktionsmuster durch das Capability-Metamodell als spezielle Form des Arbeitsprozesses dar. Über die verschiedenen Stufen der Modellierung ist dessen Charakteristik als zentral regulierter und gesteuerter Interaktionsprozess zwischen den Akteuren eines virt. Dienstleistungsproduktionsnetzwerks herausgearbeitet worden. Zugleich wurde seine Abbildung auf Service-orient. Konzepte der Koordination und Komposition festgelegt. Auf der Ebene des E-Service-Metamodells sind diese Interaktionsmuster schon in die fundamentalen Strukturen des E-Service-Konzepts eingegliedert worden. Zur kompletten technischen Abbildung des FRESCO-Dienstleistungsmodells im Sinne des FRESCO-SOM fehlt nun nur noch der Aspekt der Kopplung bzw. Komposition von Capabilities der Shell zu einer geschlossenen Dienstleistungshülle. Diese Betrachtungen schließen im Folgenden die Darstellung des FSM ab.

Das Metamodell der Prozesskopplung bzw. Komposition assoziierter Capabilities ist als letztes Diagramm des FSM in Abb. 4.26 abgebildet. Es zeigt einen Ausschnitt des Capability-Metamodells aus Abb. 4.25 und erweitert dieses um spezifische Interaktionsformen zwischen Koordinator und Provider assoziierter Capabilities. Solche Interaktionen sind dort zunächst in abstrakter Form eingeführt worden. Aus der Sicht einer Capability kann zunächst einer ihrer Interaktionsprozesse über die Kontaktpunkte (`eService Interaction Service`) von dessen Koordinator (`eService Interaction Coordinator`) von einem Koordinator einer assoziierten Capability kontaktiert werden.

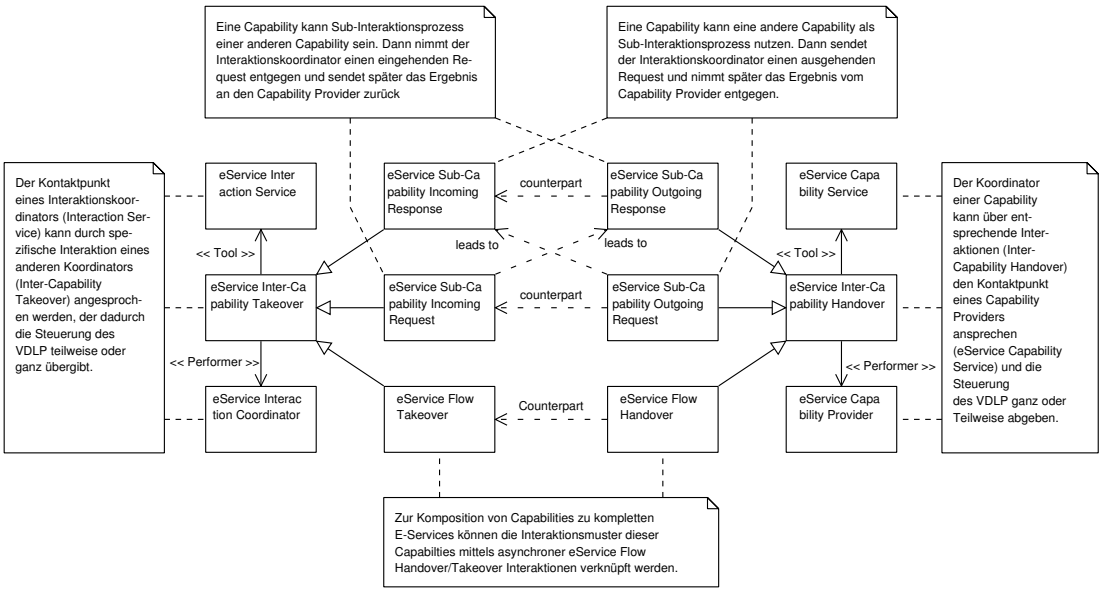

Abbildung 4.26.: FRESCO E-Service Capability-Assoziationen

Dies geschieht generell in Form einer Interaktion des Typs `eService Inter-Capability Takeover`. Dies deutet an, dass der Koordinator die Steuerung des VDLP ganz oder teilweise von dem assoziierten Koordinator übernimmt. Anders herum kann der Koordinator den Kontaktpunkt (`eService Capability Service`) des Providers einer assoziierten Capability (`eService Capability Provider`) ansprechen. Dies geschieht dann in der generellen Form einer Interaktion des Typs `eService Inter-Capability Handover`. Dies deutet an, dass der Koordinator die Steuerung des VDLP ganz oder teilweise an den Provider der assoziierten Capability abgibt.

Bei der hier angesprochenen Übergabe der Steuerung werden nun zwei konkrete Formen unterschieden. Zum einen kann das Interaktionsmuster einer Capability einen Sub-Interaktionsprozess des Interaktionsmusters einer anderen Capability bilden. Die entsprechende Kopplung der Interaktionsprozesse ist synchronisiert und besitzt ein einfaches und gut überschaubares Verhalten. Hierbei wird der Koordinator vom assoziierten Koordinator mittels einer Interaktion des Typs `eService Sub Capability Incoming Request` kontaktiert. Daraufhin steuert der Koordinator den angesprochenen Interaktionsprozess der Capability, während der assoziierte Koordinator die Steuerung ruhen lässt. An einem definierten Punkt reicht der Koordinator dann ein Ergebnis der Capability mittels Interaktion vom Typ `eService Sub Capability Outgoing Response` zurück. Der Koordinator beendet damit die Steuerung des Interaktionsprozesses und der assoziierte Koordinator setzt die Steuerung fort. Das Modell beinhaltet daneben auch den umgekehrten Fall, bei dem der Koordinator das Interaktionsmuster einer anderen Capability als Sub-Interaktionsmuster nutzt. Der Verlauf dieser Kopplung ist ganz entsprechend in umgekehrter Weise.

Die zweite Form der Steuerungsübergabe besteht in einer asynchronen Kopplung. In diesem Fall kontaktiert der Koordinator des Interaktionsprozesses einer Capability den Provider einer assoziierten Capability mittels einer Interaktion vom Typ `eService Flow Handover`. Danach setzt er die Steuerung seines Interaktionsprozeses unmittelbar fort. Der Koordinator des angesprochenen Interaktionsprozesses der assoziierten Capability verarbeitet die Nachricht völlig unabhängig davon und vollzieht die Prozesssteuerung in paralleler Weise. In umgekehrter Richtung kann der Koordinator einer Capability durch einen Koordinator einer assoziierter Capability mittels der Interaktion `eService Flow Takeover` kontaktiert werden. Der assoziierte Koordinator setzt dann seine Steueraktivitäten fort und die Verarbeitung der Interaktion durch den Koordinator läuft davon unabhängig und parallel ab. Generell erlaubt die Konstruktion der asynchronen Kopplung mittels `Flow Handover/Takeover` die Modellierung sehr vielfältiger Verhaltensmuster. Z. B. können dadurch Sequenzen, Parallelisierungen und Schleifen von Steuervorgängen verschiedener Akteure erreicht werden.

4.4. Service-orient. Entwicklung zur Regelung der VDL-Produktion

Im letzten Abschnitt wurde die Realisierung virt. Dienstleistungen auf Basis integrativer IV-Technik eingehend untersucht. Dabei wurden zwei wesentliche Aspekte einer solchen Technologie abgeleitet und durch die Begriffe des E-Services und E-Service-Managements zusammengefasst. E-Services bilden die Kommunikationsendpunkte und Interaktionsmuster virt. Dienstleistungsprozesse auf integrative Techniken ab. Sie realisieren dadurch den integrativen Kern eines zwischenbetrieblich verteilten Anwendungssystems zur Steuerung der Dienstleistungsproduktion im virt. Dienstleistungsproduktionsnetzwerk. E-Service-Management betrifft strukturierte Vorgangsmodelle, um die Erstellung, Durchführung und Auswertung von E-Services und E-Service-basierten Anwendungssystemen für das Produktionsmanagement von VDPN anzuwenden. Für E-Services wurden im letzten Abschnitt grundsätzliche Realisierungsmöglichkeiten auf Basis des SOC-Paradigmas diskutiert. Zudem wurde ein exemplarisches Service-orient. Modell für das konzeptionelle FRESCO-Dienstleistungsmodell entworfen und durch ein formales E-Service-Metamodell konkretisiert. Auf Basis dieser Ergebnisse fokussieren die folgenden Sektionen nun den Aspekt des E-Service-Managements. Dies bildet den dritten Teil des konzeptionellen Rahmenwerks.

Den Hintergrund des E-Service-Managements bildet die Konzeption von E-Services als Mechanismus zur Unterstützung der virtuellen Organisation von Dienstleistungsunternehmen. Entsprechende virt. Dienstleistungsunternehmen basieren auf einem strategischen Dienstleistungsnetzwerk, in dem autonome Dienstleistungsunternehmen als Netzwerkunternehmen miteinander assoziiert sind. Aus der Perspektive der Dienstleistungsproduktion bilden diese DLU Produktionseinheiten eines virt. Produk-

tionsnetzwerks, in dem Dienstleistungen produziert werden. Produktionseinheiten planen die Produktion von Dienstleistungen in weitgehend unabhängiger Weise auf Basis ihrer Kernkompetenzen und bringen die resultierenden Produktionsprogramme in den gemeinsamen Kompetenzpool des strategischen Netzwerks ein. Ein DLU in Broker-Rolle nimmt Anfragen potenzieller Kunden an das strategische Dienstleistungsnetzwerk entgegen. Auf Basis der Anforderungen plant der Broker dann die Dienstleistungsproduktion auf Netzwerkebene, wobei er die Produktionsprogramme einzelner Produktionseinheiten einbezieht. Die Planung des Produktionsprogramms auf Netzwerkebene führt zu einem spezifischen VDPN, das für die Produktion der angefragten Dienstleistung optimiert ist und nur zu diesem einen Zweck besteht. Dies bildet einen Teil der Vereinbarung zwischen Netzwerkunternehmen und Kunde zur Bildung eines VDU. In der Folge setzen DLU in Koordinator-Rolle das Produktionsprogramm auf Netzwerkebene durch Steuerung der Produktion zwischen den Produktionseinheiten um. Produktionseinheiten führen die Steuerung ihrer Produktion im Sinne individueller Produktionsprogramme eigenständig durch. Am Ende wird das VDU mitsamt dem VDPN aufgelöst, wobei die Produktionsprogramme zur Planung zukünftiger VDPN ausgewertet und bewahrt werden.

Ein wesentlicher Aspekt im Lebenszyklus des VDPN ist die einfache und schnelle Regelung der Koordination kooperativer Tätigkeiten bei der Produktion von Dienstleistungen durch verschiedene Produktionseinheiten. Genau zu diesem Zweck soll die Virtualisierung von Dienstleistungen dienen. Der Dienstleistungsprozess bestimmt gleichermaßen das Produkt und den interaktiven Produktionsvorgang der Dienstleistungsproduktion. Die darin enthaltenen Interaktionen zwischen Produktionseinheiten und Kunden spiegeln deren Kooperation bei der Dienstleistungsproduktion wider. In diesem Sinne lässt sich mittels Gestaltung der Interaktion im Dienstleistungsprozess die Koordination der kooperativen Produktion im VDPN regeln. In einem virt. Dienstleistungsprozess werden die interaktiven Anteile des DLP durch die integrativen IV-Techniken eines E-Services realisiert. Ein derartiger E-Service repräsentiert und realisiert die Koordination kooperativer Tätigkeiten eines spezifischen VDPN. Der E-Service-Lebenszyklus soll nun zur Realisierung von Teilaspekten des Produktionsmanagements in Dienstleistungsnetzwerken bzw. von VDPN eingesetzt werden. Die Entwicklung von E-Services soll der Regelung der Koordination im Zuge der Produktionsplanung für VDPN dienen. Die Ausführung von E-Services soll als Mittel zur Produktionssteuerung im VDPN zum Einsatz kommen. Die Evaluation und Änderung von E-Services soll bei der Auflösung von VDPN das Erfahrungswissen in Bezug auf die Koordination bewahren. Es ist wichtig zu beachten, dass bei einer derartigen Vorgehensweise der Lebenszyklus von E-Services dem von VDPN angeglichen werden muss. Dazu wird ein spezifisches Vorgehensmodell zur Entwicklung von E-Services mitsamt geeigneten Phasen eines Entwicklungsprozesses und entsprechenden Entwicklungsmethoden benötigt.

Spezifische Prozesse und Methoden zur Entwicklung von E-Services im Kontext des Produktionsmanagements von VDPN werden in der vorliegenden Arbeit als E-Service-

Management zusammengefasst. Aus übergeordneter Perspektive muss sich ein solches Vorgehensmodell dabei in ein ganzheitliches Konzept zur Gestaltung und Führung von Dienstleistungsunternehmen einfügen. Aus diesem Grund ist bei der Konzeption des E-Service-Managements generell die Integration in den Kontext betrieblicher Prozesse zu beachten. Diesem Aspekt widmet sich die nachfolgende Sektion 4.4.1. Hier wird das E-Service-Management zunächst in die ganzheitliche Sicht des Enterprise Engineering nach dem Schema des Enterprise-Architektur-Rahmenwerks von GERAM eingeordnet. In diesem Rahmen wird dann die konzeptionelle Grundlage des E-Service-Managements als spezifische Enterprise Engineering Methodology gelegt. EEMs bilden die durch eine Enterprise-Architektur vorgegebenen zusammenhängenden Lebenszyklen betrieblicher Komponenten auf entsprechende Vorgehensmodelle ab. Diese beinhalten konkrete Entwicklungsprozesse und -methoden zur praktischen Durchführung des Enterprise Engineering. In diesem Sinne werden zunächst die relevanten Lebenszyklen der Enterprise-Architektur von VDPN analysiert. Auf dieser Basis wird dann ein abstraktes Vorgehensmodell zur Service-orient. Entwicklung von E-Services entwickelt.

Das abstrakte Vorgehensmodell bildet eine Grundlage, um in Bezug auf konkrete E-Service-Modelle spezifische Prozesse und Methoden entwerfen zu können. Für ein solches konkretes Vorgangsmodell können wiederum Werkzeuge entwickelt werden, die den praktischen Vorgang des E-Service-Managements unterstützen. Um den Entwurf eines konkreten Vorgangsmodells exemplarisch zu demonstrieren, schwenkt die Betrachtung in Sektion 4.4.2 wieder auf die konkreten Ansätze im FRESCO-Projekt. In diesem Kontext wird zunächst die Anpassung des im abstrakten Vorgehensmodell beschriebenen Entwicklungsprozesses an die spezifischen Eigenschaften des konzeptionellen FRESCO-Dienstleistungsmodells und FRESCO E-Service-Metamodell diskutiert. Im Anschluss werden die Entwicklungsmethoden der einzelnen Phasen erörtert.

4.4.1. Integration des E-Service-Managements in den DLU-Kontext

Die informationstechnische Repräsentation des Interaktionsprozesses einer Dienstleistung durch E-Services bietet eine Basis für verschiedene Möglichkeiten, durch deren Erstellung, Durchführung und Auswertung das Produktionsmanagement im VDPN zu unterstützen. Hierzu müssen nicht nur die organisatorischen und technischen Modelle von E-Services die Interaktionsprozesse im Rahmen des Dienstleistungsprozesses in geeigneter Weise abbilden. Auch die dispositiven Unternehmensprozesse müssen durch ein Vorgehensmodell zum E-Service-Management in geeigneter Weise wiedergegeben werden.

Zur ganzheitlichen Integration des VDL-Konzepts in den betrieblichen Kontext von Dienstleistungsunternehmen bietet sich der Rückgriff auf Referenzmodelle an. Dies ist ein grundlegender Schritt genereller Vorgehensmodelle der Systemintegration.[37]

[37] Siehe Sektion 3.2.1.2

Ein solches Modell wurde im letzten Kapitel mit der *Generalised Enterprise Reference Architecture And Methodology (GERAM)* eingeführt.[38] GERAM beschreibt ein abstraktes Rahmenwerk für die Bestandteile des Enterprise Engineering. Hier gibt ein EA-Referenzmodell die Strukturierung und Integration einer Vielzahl verschiedener Theorien, Formalismen, Modelle, Sprachen, Mechanismen und Methoden auf allen Ebenen und in allen Bereichen eines Unternehmens vor, inkl. solcher im Zusammenhang mit betrieblichen Informationssystemen. In Sektion 4.4.1.1 werden die Bestandteile einer E-Service-Technologie aus den Kategorien von GERAM abgeleitet und entsprechend strukturiert. Eine derartige Klassifikation soll als Grundlage zur Integration von E-Service-Technologie in den Kontext verschiedener Organisationen dienen. Des Weiteren bildet sie eine Übersicht der bei der Realisierung des E-Service-Managements zu berücksichtigenden Bestandteile.

Einer der grundlegenden Aspekte von Enterprise-Architekturen ist der Lebenszyklus verschiedener Unternehmensbestandteile. Das schließt im Falle virt. Dienstleistungsproduktion das Dienstleistungsnetzwerk, die darin temporär gebildeten VDPN sowie das Dienstleistungsprodukt mitsamt seiner virtualisierten Anteile ein. Damit einher geht deren Lebenshistorie, d. h. die Zusammenhänge verschiedener Bestandteile in Bezug auf deren Lebenszyklen. Dieser Aspekt zeigt sich bei der virt. Dienstleistungsproduktion u. a. durch die Zusammenhänge der Lebenszyklen von VDPN und E-Services. Eine E-Service-Technologie zur Unterstützung der Produktionssteuerung im VDPN muss grundsätzlich dessen Lebenszyklus unterstützen. Dies führt in erster Linie zu Anforderungen an das E-Service-Management. Zum einen liegen hier die Eigenschaften des zu unterstützenden Lebenszyklus von VDPN als Anforderung zugrunde. Zum anderen beeinflusst die Wahl konkreter integrativer IV-Techniken für E-Services den Gestaltungsspielraum für Entwicklungsprozesse. In der vorliegenden Arbeit werden vor allem SOC-Techniken als Basis für E-Services untersucht. In Sektion 4.4.1.2 konzentriert sich die Diskussion daher auf diesen Bereich. Hierbei werden die Lebenszyklen virt. Dienstleistungsproduktionsnetzwerke und Service-orient. Anwendungssysteme gegenübergestellt. Auf Basis dieses Vergleichs erfolgt dann der Entwurf eines optimierten Entwicklungslebenszyklus für Service-orient. E-Services als abstraktes Vorgehensmodell für das E-Service-Management.

4.4.1.1. E-Service-Management als Enterprise Engineering

Enterprise Engineering ist ein Paradigma für Konstruktion, Betrieb und Auflösung von Unternehmen im Sinne einer Ingenieursdisziplin. Hierbei werden Modelle und Methoden verschiedener funktionaler Ebenen, Bereiche und Aspekte der Unternehmensführung als *Enterprise-Architektur-Rahmenwerk* zusammengefasst und miteinander in Beziehung gesetzt. Eine abstrakte Oberklasse solcher Rahmenwerke wird durch GERAM definiert. Dessen *Generalised Enterprise Reference Architecture (GERA)* defi-

[38] Siehe Sektion 3.3.1.2

niert hier insbesondere *Lebenszyklen* verschiedener Unternehmensbestandteile und strukturiert deren Modelle und Methoden in Bezug auf einzelne Lebenszyklusphasen. Dies umfasst Konzept-, Architektur- und Detailentwurf sowie Implementierung, Betrieb und Auflösung des Unternehmens selbst sowie seiner verschiedenen Bereiche und Produkte. Ferner führt GERA mit der *Lebenshistorie* ein Konzept ein, das zur Erfassung der Dynamik von Lebenszyklen im Einzelnen und im Zusammenhang dient. Die Vorgaben von Lebenszyklen und Lebenshistorie in GERA beziehen sich auf *Enterprise Engineering Methodology (EEM)* zur Beschreibung von Methoden und auf *Enterprise Modelling Languages (EMLs)* zur Beschreibung von Modellen in und zwischen einzelnen Phasen.

Insgesamt sollen nach Vorgabe von GERA bzw. konformer *Enterprise-Architekturen (EAs)* die Unternehmensprozesse als Ströme und Transformationen von Material und Informationen mitsamt den dazu notwendigen Ressourcen und Personen möglichst vollständig erfasst werden. Dies bezieht alle Arten von Prozessen inkl. materieller, informationeller, primärer, sekundärer, innovativer, operativer und Managementprozesse ein. Darüber hinaus ist die Verknüpfung einzelner Prozesse nach Vorgabe übergeordneter Geschäftsprozesse und die daraus resultierende Informationsarchitektur zu erfassen. Diese ganzheitliche Sicht erlaubt dann den Entwurf sowie die wechselseitige Abstimmung und Integration einer Vielzahl von Unternehmensaspekten. Durch die besondere Beachtung dynamischer Aspekte in der Lebenshistorie von Unternehmensbestandteilen erfolgt dies nicht nur für einzelne Zeitpunkte, sondern kontinuierlich. All diese Vorgänge werden durch *Enterprise Engineering Tools (EET)* mittels Implementierung von EEM und EMLs der EA unterstützt. Als zusätzliche Unterstützung kann hier die Semantik von EMLs im Sinne der EA explizit durch *Generalised Enterprise Modelling Concepts (GEMC)* beschrieben werden. Die Berücksichtigung wiederverwendbarer Teillösungen in Form von *Partial Enterprise Models (PEM)* führt zu weiteren Erleichterungen beim Enterprise Engineering.

Mithilfe von EET werden beim Enterprise Engineering nach den Mustern von EEM und mit den Konstrukten von EMLs *Enterprise Models (EM)* erstellt. EM bilden das Unternehmen im Modell ab. Modelle können daraufhin analysiert, optimiert und schließlich implementiert werden. Die Implementierung von EM-Aspekten führt zu *Enterprise Operative Systems (EOS)*, die wiederum einen operationalen Aspekt der Unternehmung realisieren. Zur Implementierung des EOS werden so weit wie möglich vorhandene *Enterprise Modules (EMO)* als Komponenten wiederverwendet. Solche EOS können dann nicht zuletzt auch integrierte Informationssysteme sein. Diese fügen sich in die ganzheitliche Unternehmensbetrachtung ein und unterstützen hier die informationellen Prozesse. In diesem Kontext werden geschäftliche Anwendungssysteme direkt aus den Anforderungen übergeordneter Modellebenen abgeleitet, auf Basis formaler Modelle und Methoden produktmäßig entwickelt und über die Lebenshistorie kontinuierlich angepasst.

Daneben besteht auch das Enterprise Engineering selbst aus informationellen Prozessen, die durch Anwendungssysteme zu implementieren sind. Hierfür sieht

GERAM eine Infrastruktur aus *Enterprise Model Execution and Integration Services (EMEIS)* vor. Diese EMEIS implementieren zum einen Prozesse zur Verwendung von EET im Zuge des Enterprise Engineering und zur Implementierung von EM als EOS. EET-Prozesse werden als *Model Development Tools* implementiert. Prozesse zur Implementierung von EM werden hingegen durch *Model Execution Services* umgesetzt. Zum anderen werden Prozesse zur Kollaboration im Zuge des Enterprise Engineering implementiert. Die entsprechenden *Shared Services* bilden eine Infrastruktur, die insbesondere in organisationsübergreifenden Szenarien den interoperablen Austausch von Modellen erlaubt.

Auch das in der vorliegenden Arbeit untersuchte Konzept zur Dienstleistungsvirtualisierung basiert auf Unternehmensprozessen eines VDPN, erfasst deren informationellen Aspekt der Interaktion in einem Modell und überführt diesen letztendlich in ein Anwendungssystem zur Unterstützung operativer Steuervorgänge. Im konzeptionellen Rahmen von VDPN und VDL werden hier durch E-Services die Interaktionen zwischen Netzwerkunternehmen erfasst. Für diese Aspekte werden im Wesentlichen Modelle von Interaktionsmustern und Endpunkten definiert. E-Service-Modelle werden letztendlich auf verschiedene integrative Basistechniken abgebildet. Sie bilden dann den integrativen Kern eines zwischenbetrieblichen operativen Anwendungssystems zur Produktionssteuerung im VDPN. E-Service-Management beinhaltet ein Vorgehensmodell, das Gestaltungsmethoden für E-Services als Gestaltungsprozess strukturiert, der dem VDPN-Lebenszyklus konform geht. Dies beinhaltet Erstellung, Abgleich, Auswertung und Änderung von E-Service-Modellen sowie Konstruktion und Ausführung verteilter Produktionssteuerungssysteme. Damit stellt sich E-Service-Management als spezielle Facette des Enterprise Engineering dar. Diese konzentriert sich auf spezifische Bestandteile und Aspekte der Modelle, Methoden, Werkzeuge und operativen Systeme im VDPN.

Als Aspekt des ganzheitlichen Enterprise Engineering kann E-Service-Management in Hinblick auf das GERAM-Referenzmodell strukturiert werden. Hierbei gibt GERAM die Kategorien und Beziehungen notwendiger Bestandteile vor. Eine solche Ausrichtung der E-Service-Technologie an den generischen Vorgaben eines Referenzmodells hat mehrere potenzielle Vorteile. Das Referenzmodell gibt einen standardisierten Ansatz vor, an dem sich die Konzeption von E-Service-Techniken im Allgemeinen und des E-Service-Managements im Besonderen ausrichten kann. Hierdurch können konzeptionelle Fehler vermieden werden. Außerdem fördert der Bezug auf den Standard die Akzeptanz und das Verständnis der E-Service-Technologie im organisationsübergreifenden Kontext des VDPN. Darüber hinaus liefert die Eingliederung in ein umfassenderes Rahmenwerk den Ansatzpunkt zur Integration des E-Service-Managements mit anderen unternehmerischen Gestaltungsprozessen.

Abb. 4.27 zeigt die Strukturierung und das Zusammenspiel verschiedener Bestandteile des E-Service-Managements nach Vorgabe der Kategorien des GERAM-Rahmenwerks. Dessen zentraler Aspekt ist die Enterprise-Architektur, für die durch

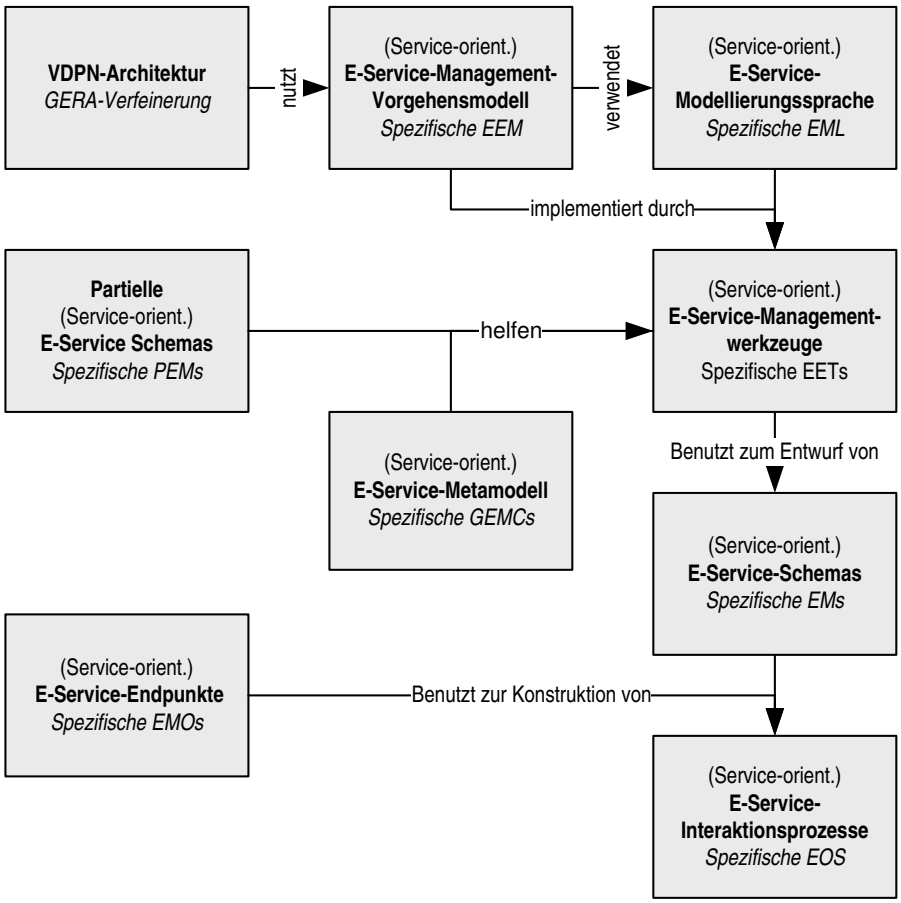

Abbildung 4.27.: E-Service-Management im Rahmen des Enterprise Engineering

GERA ein Referenzmodell vorgegeben wird. Dessen generische Konzepte, die u. a. Lebenszyklen, Lebenshistorie, Prozessmodellierung und Modell-Infrastruktur beinhalten, bilden eine konzeptionelle Grundlage für das E-Service-Management. Sie müssen jedoch durch spezifische Eigenschaften der Lebenszyklen, Prozesse und Ressourcen bei virtueller Dienstleistungsproduktion verfeinert werden. Diese Aspekte einer *VDPN-Architektur* werden durch konzeptionelle VDPN- und VDL-Modelle definiert.

Die VDPN-Architektur wird durch Modelle und Methoden für E-Services und E-Service-Management umgesetzt. Die Erfassung organisatorischer und technischer Konzepte der virt. Dienstleistungsproduktion basiert auf spezifischen EMLs für E-Services. Diese *E-Service-Modellierungssprachen* basieren wiederum auf spezifischen GEMC für E-Services. GEMC können dabei z. B. durch ein *E-Service-Metamodell* formal spezifiziert werden. Ein derartiges Metamodell legt dann die Konzepte von E-Service-Modellierungssprachen fest. Hierfür eignen sich besonders Service-orient. Ansätze, die vielen spezifischen Konzepten virtueller Dienstleistungsproduktion generische Konzepte zwischenbetrieblicher Interaktion entgegensetzen. Entsprechende

Sprachkonstrukte werden dann zur Spezifikation konkreter *E-Service-Modelle bzw. -Schemata* verwendet. Diese Schemata fügen sich in die GERA-Protokollinfrastruktur ein. Generell dienen diese Modelle zur Beschreibung von VDL und VDPN in verschiedenen Lebenszyklusphasen. Diese Lebenszyklen werden im E-Service-Management durch eine spezifische EEM nachvollzogen, die sich als *E-Service-Management-Vorgehensmodell* darstellt. Im speziellen Fall Service-orient. E-Service-Metamodelle und -Schemata liegt dem ein Service-orient. Entwicklungslebenszyklus für E-Services zugrunde.

Das E-Service-Management-Vorgehensmodell ist durch *E-Service-Management-Werkzeuge* als spezifische EET zu implementieren. Ein wesentlicher Aspekt ist hier der Entwurf von E-Service-Schemata auf Basis von E-Service-Modellierungssprachen. Hierbei sollte es möglich sein, Entwurfsmuster oder wiederverwendbare Komponenten als spezifische PEM zu verwenden, die sich in diesem Kontext als *partielle E-Service-Schemata* darstellen. Ein zweiter Aspekt von E-Service-Management-Werkzeugen ist die Überführung von E-Service-Schemata in spezifische EOS. Solche EOS sind operative zwischenbetrieblich verteilte Anwendungssysteme zur Steuerung der virtuellen Dienstleistungsproduktion im VDPN. Diese Steuersysteme basieren auf Software-Komponenten, die die Interaktionsmuster im VDPN als *E-Service-Interaktionsprozesse* implementieren. E-Service-Interaktionsprozesse bilden die koordinative Ebene zur zwischenbetrieblichen Integration des Steuersystems. Steuersysteme integrieren wiederum Software-Komponenten von Netzwerkunternehmen, die *E-Service-Endpunkte* als spezifische Form von EMO zur Verfügung stellen.

Die Auslegung von E-Service-Management als Enterprise Engineering lässt sich auch auf die Spezialisierung der GERAM-EMEIS ausweiten. Hier kann zwischen *E-Service Model Development Services* und *E-Service Model Execution Services* unterschieden werden. Diese implementieren zum einen Prozesse der E-Service-Management-Werkzeuge in Bezug auf Entwurf, Analyse und Änderung von E-Service-Modellen. Zum anderen implementieren sie Prozesse für die Konstruktion und den Betrieb von VDPN-Steuersystemen. Dies kann etwa die Generierung und Ausführung von E-Service-Interaktionsprozessen beinhalten. Eine weitere Kategorie spezialisierter EMEIS sind *E-Service Shared Services*. Diese unterstützen den Austausch von Modellen zwischen VDPN-Broker, -Providern und -Kunden.

Die hier identifizierten Bestandteile eines auf die spezifische Perspektive des E-Service-Managements spezialisierten Enterprise-Architektur-Rahmenwerks werden im Verlauf der vorliegenden Arbeit realisiert. Dabei kann zwischen konzeptionellen und technischen Bestandteilen unterschieden werden. Die konzeptionellen Bestandteile von VDPN-Architektur, E-Service-Metamodell und E-Service-Management-Vorgehensmodell bilden das konzeptionelle Rahmenwerk zur Unterstützung der Produktionssteuerung im VDU, das in diesem Kapitel beschrieben wird. Die konkreten Bestandteile der E-Service-Modellierungssprachen sowie kompletter und partieller Schemata, E-Service-Management-Werkzeuge, E-Service-Kontaktpunkte, -Interakti-

onsprozesse und entsprechender EMEIS bilden das zugehörige technische Rahmenwerk, das im nächsten Kapitel beschrieben wird. Im Folgenden wird zunächst der verbleibende konzeptionelle Bestandteil des E-Service-Management-Vorgehensmodells für den allgemeinen Fall Service-orient. E-Service-Modelle und für das konkrete Beispiel von FRESCO-E-Services erarbeitet.

4.4.1.2. E-Service-Management-Vorgehensmodelle

E-Services bilden einen technischen Ansatz zur Koordination von Produktionseinheiten und Kunden eines VDPN. Sie sind dabei als Instrument des Produktionsmanagements zur Unterstützung bei Regelung und Steuerung der Produktion ausgelegt. Da E-Services die Interaktionsprozesse individueller VDPN regeln und steuern sollen, müssen sie dabei auch deren Lebenszyklus wiedergeben. Die Eigenschaften des VDPN-Lebenszyklus bilden also die fundamentale Anforderung an ein Vorgehensmodell für E-Service-Management. Aus diesem Grund wird der Lebenszyklus im folgenden Paragraph noch einmal analysiert.

Aus technischer Sicht bilden E-Services den integrativen Kern eines zwischenbetrieblich verteilten Anwendungssystems. Dies beinhaltet zum einen Techniken zur Realisierung von Kommunikationsendpunkten, die eine interoperable Kommunikation unternehmensübergreifender AS-Komponenten ermöglichen. Zum anderen werden Komponenten benötigt, die eine koordinative Ebene zur Orchestrierung der Interaktionen zwischen diesen Kommunikationsendpunkten bilden. Im zweiten Teil des konzeptionellen Rahmenwerks wurde die Realisierung dieser Integrationstechniken auf Basis von SOC-Techniken untersucht. Hier wurden Techniken der WS-Basisarchitektur als Grundlage der Kommunikation und Techniken der WS-Koordination und -Komposition als Grundlage der koordinativen Ebene vorgeschlagen. Folgt man diesem Ansatz weiter, so gründet sich die Entwicklung von E-Service auf Service-orient. Entwicklungsmethoden und -prozesse. Service-orient. Entwicklungslebenszyklen bilden also die Grundlage eines Vorgehensmodells für E-Service-Management. Daher werden im zweiten der nachfolgenden Paragraphen einige aktuelle Ansätze dazu dargestellt.

Schließlich erfolgt die Synthese eines abstrakten Vorgehensmodells für das E-Service-Management auf Basis Service-orient. Entwicklungsmethoden. Dazu erfolgt ein Vergleich und eine Abbildung der Lebenszyklen von Service-orient. Anwendungssystemen und virt. Dienstleistungsproduktionsnetzwerken. Dabei zeigt sich, dass die Lebenszyklen nicht identisch sind und eine Abbildung auf verschiedene Art möglich ist. Auf Basis einer Diskussion verschiedener Möglichkeiten wird ein optimierter Service-orient. E-Service-Entwicklungslebenszyklus entwickelt. Dieser bildet einen abstrakten Rahmen zur Ausgestaltung konkreter E-Service-Management-Vorgehensmodelle für spezifische Service-orient. E-Service-Modelle.

VDPN-Lebenszyklus Der Lebenszyklus virt. Dienstleistungsproduktionsnetzwerke kann aus verschiedenen Perspektiven betrachtet werden. Dazu gehört zunächst die Perspektive allgemeiner Managementaufgaben in Netzwerkorganisationen. Ein weiterer Blickwinkel betrifft den Lebenszyklus der Dienstleistungserbringung. Des Weiteren ist der fundamentale Lebenszyklus virt. Unternehmenen von zentralem Interesse. Schließlich liefert auch GERAM ein generisches Lebenszyklusmodell, das sich hier anwenden lässt. Im Folgenden werden diese Betrachtungsweisen verglichen und als Grundlage für die Ableitung eines Lebenszyklusmodells für VDPN zusammengefasst.[39]

Um die verschiedenen Perspektiven auf den VDPN-Lebenszyklus zusammen zu bringen, eignet sich grundsätzlich das generische Lebenszyklusmodell von GERAM. Dieses erlaubt die Strukturierung von Lebenszyklen verschiedener Komponenten einer Enterprise-Architektur in Form ihrer Lebenshistorie. Für die Betrachtung virt. Dienstleistungsproduktion sind hier primär die Komponenten der Virt. Produktionsnetzwerke und Dienstleistungen von Interesse. Deren zusammenhängende Lebenshistorie wird in Abb. 4.28 dargestellt. Ferner sind Lebenszyklen verschiedener spezieller Perspektiven in den Zusammenhang eingeordnet.

GERAM beschreibt ein generisches Lebenszyklusmodell mit acht Phasen. Jede Komponente der Enterprise-Architektur durchläuft hier die Phasen Identifikation, Konzeption, Anforderungsanalyse, vorläufiger Entwurf, Detailentwurf, Implementierung, Nutzung und Stilllegung. In der Abbildung bilden die entsprechenden Lebenszyklen von VDPN und Dienstleistungen den zentralen Aspekt. Im Verlauf der Lebenszyklen stehen deren Phasen in Beziehung zueinander. Die Lebenszyklen verlaufen jedoch nicht exakt parallel, sondern sind zeitlich versetzt. Zur Kennzeichnung zusammenhängender Phasen werden horizontale Stufen verwendet, die in der Abb. durchnummeriert sind. Die Phasen einer Stufe treten jeweils gleichzeitig auf und stehen in wechselseitigem Zusammenhang.

Der grundlegende Lebenszyklus wird durch Dienstleistungsnetzwerke gebildet.[40] Diese bilden den Kontext eines strategischen Unternehmensnetzwerks, in dem sich spezifische VDPN zur Produktion einzelner Dienstleistungen bilden können. Dienstleistungsnetzwerke entwickeln sich zunächst über die verschiedenen Lebenszyklusphasen bis hin zum Betrieb. Die hierzu notwendigen Vorgänge haben jedoch wenig mit der letztendlichen Dienstleistungsproduktion zu tun und werden daher nicht näher betrachtet. Im Betrieb gehen aus dem Netzwerk bis zur letztendlichen Stilllegung multiple Generationen virt. Dienstleistungsproduktionsnetzwerke hervor.

Der Lebenszyklus eines VDPN beginnt mit dessen *Identifikation*, d. h. mit der Idee zu dessen Entwicklung. Diese Phase, die im GERAM-Modell eher kurz ausgelegt ist, findet sich auch im Lebenszyklusmodell für allgemeine VU nach Mertens [MF97]. Dort umfasst sie jedoch weitere Aspekte, die in GERAM mit *Konzept*, *Anforderungen*

[39] Siehe für Netzwerkmanagement Sektion 2.3.2.1, für VU- und DLU-Lebenszyklen Sektion 3.2.2.3 und für das GERAM-Lebenszyklusmodell Sektion 3.3.1.2

[40] Der Lebenszyklus von Dienstleistungsnetzwerken wird im Folgenden nicht weiter betrachtet und daher in Abb. 4.28 nicht dargestellt.

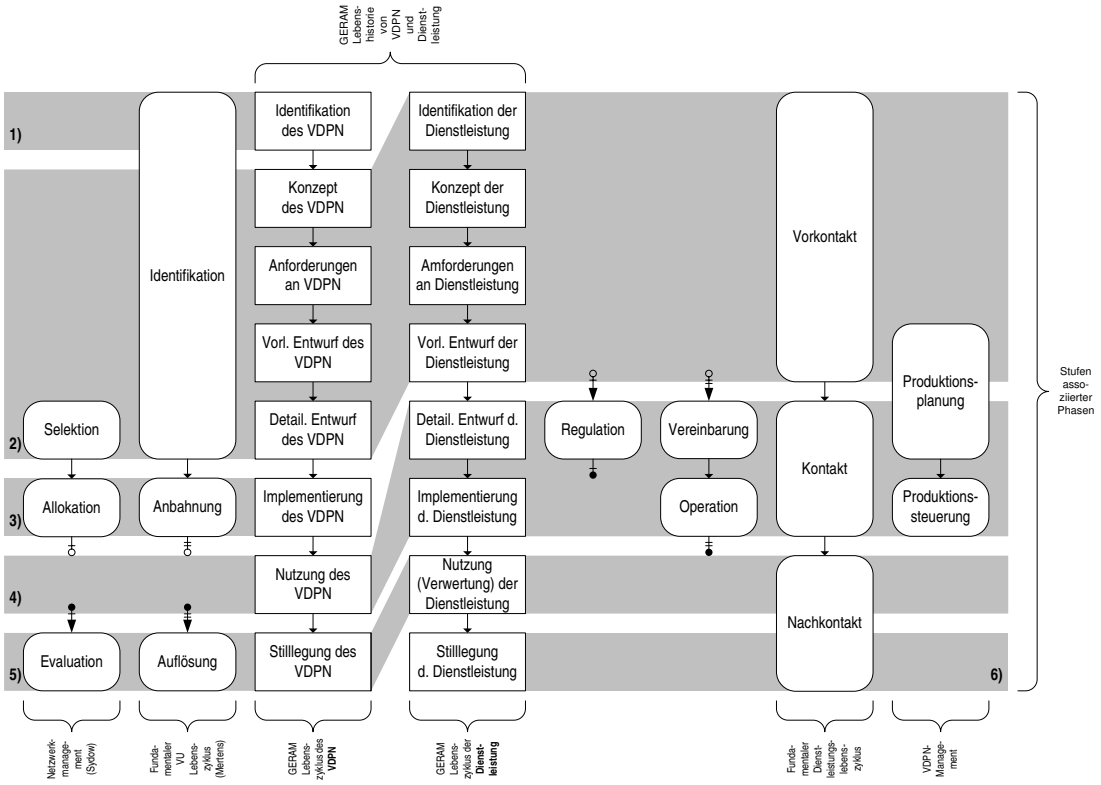

Abbildung 4.28.: Lebenshistorie virt. Dienstleistungsproduktion

und *vorläufigem* sowie *detailliertem Entwurf des VDPN* feiner aufgegliedert werden. Ein wesentlicher Anteil dieser Phasen besteht in der Vorbereitung der im VDPN zu produzierenden Dienstleistung. Im Sinne des grundlegenden Dienstleistungslebenszyklus findet hier die *Vorkontaktphase* statt, in der das Potenzial zur Erbringung der Dienstleistung gebildet wird. Dies beinhaltet *Identifikation*, *Konzept*, *Anforderungen* und *vorläufigen Entwurf der Dienstleistung*. Der vorläufige Entwurf der Dienstleistung bezieht die initiale *Produktionsplanung* auf der Netzwerkebene eines VDPN ein. In Bezug auf die Managementfunktionen für Unternehmensnetzwerke nach Sydow [Syd99, S. 295] bilden diese Vorgaben eine Basis zur *Selektion* von DLU für das strategische Dienstleistungsnetzwerk. Hierauf begründet sich auch das Angebot des Dienstleistungsnetzwerks für Kunden. Dieses kann aber insbesondere auch auftragsinduziert sein. Am Ende von Stufe 2) sind VDPN und Dienstleistung soweit geplant, dass sie für spezifische Fälle realisiert werden können.

In Stufe 3) wird das VDPN für eine spezifische Dienstleistung implementiert. Dies beinhaltet auf der einen Seite die *Anbahnung* durch potenzielle Kunden. Auf der anderen Seite findet eine *Allokation* geeigneter DLU statt. Stehen die Teilnehmer der virt. Dienstleistungsproduktion dann fest, beginnt die *Kontaktphase* der Dienstleistung.

Dabei werden in Stufe 4) zunächst die detaillierten Bedingungen in *Vereinbarung* mit den Kunden und die *Regulation* mit den beteiligten DLU festgelegt. Dies manifestiert sich im detaillierten Entwurf der Dienstleistung. Die Festlegung des kollaborativen DLP und der dabei ablaufenden Interaktionen bildet dabei eine detaillierte *Produktionsplanung* für das VDPN. In der Folge findet eine *Nutzung des VDPN* zur Produktion der Dienstleistung statt, was deren *Implementierung* entspricht. Bei dieser *Operation* des VDPN wird durch *Produktionssteuerung* der geplante Ablauf des DLP durchgesetzt.

Nach Beendigung des DLP ist die Kontaktphase der Dienstleistung beendet und es beginnt in Stufe 5) die *Nachkontaktphase*. In dieser Phase findet eine *Nutzung der Dienstleistung* durch Verwertung der Nutzeffekte statt. Der Nutzen ergibt sich hier primär für den Kunden. Auf der anderen Seite beginnt die *Stilllegung des VDPN*. Dies bezieht sich auf die Konstellation von DLU zur Produktion der spezifischen Dienstleistung. Bei deren *Auflösung* findet eine *Evaluation* des Produktionsverlaufs statt. Die Ergebnisse fließen in die Planung des VDPN für zukünftige Fälle der Dienstleistungsproduktion ein. Schließlich erfolgt in Stufe 6) die Stilllegung der Dienstleistung. Diese eher theoretische Phase beschreibt das Verwerfen der Nutzeffekte einer Dienstleistung auf Kundenseite.

Service-orient. Entwicklungslebenszyklus Die Entwicklung Service-orient. Anwendungssysteme erfordert spezifische Methoden und Entwicklungsprozesse. Diese leiten sich aus den allg. Ansätzen des Software Engineering ab und berücksichtigen darüber hinaus die Anforderungen des Service-orient. Modells. Zusammengenommen bilden sie ein Vorgehensmodell für den zyklischen Entwicklungsprozess Service-orient. Anwendungssysteme, der auch als Service-orient. Entwicklungslebenszyklus bezeichnet wird [PH06].

Das Ziel eines Service-orient. Entwicklungslebenszyklus besteht grundsätzlich in Analyse, Entwurf und Produktion Service-orient. Anwendungssysteme. Dies soll derart geschehen, dass die Geschäftsprozess-Interaktionen von Geschäftspartnern in Hinblick auf ein gemeinsames Ziel wiedergeben werden. Das Vorgehensmodell selbst bezieht eine Mischung von Fähigkeiten, Technologien, Werkzeugen und Qualifikationen ein, die verschiedene Aspekte Service-orient. Software-Entwicklung abdecken. Insbesondere sollen diese das Management des kompletten Lebenszyklus eines Service-orient. Anwendungssystems erlauben, eine Plattform und ein Programmiermodell für Service-orient. Anwendungssysteme zur Verfügung stellen, praktische Verfahren und Werkzeuge umsetzen und die Produktion praktikabler Lösungen in Bezug auf qualitative Merkmale ermöglichen.

Mithilfe dieser Mittel werden im Zuge des Service-orient. Entwicklungslebenszyklus Services identifiziert, zusammen in Hierarchien einfacher und zusammengesetzter Services strukturiert und im Sinne von korrespondierenden Geschäftsprozessen orchestriert. Dabei wird die Entwicklung durch Prozesse und Regeln der geschäftlichen Ebene getrieben. Von dort aus erfolgt sie dann hin zu einer technischen Lösung für die

Automation und Integration von Unternehmen. Auf diese Weise soll die Verfolgung geschäftlicher Anforderungen von den geschäftlichen Zielen über den Software-Entwurf bis hin zu Software-Komponenten und zusammengesetzten Anwendungssystemen ermöglicht werden.

Für das Vorgehensmodell eines derartig charakterisierten Service-orient. Entwicklungslebenszyklus schlagen Papazoglou und van den Heuvel einen iterativen und inkrementellen Prozess mit 8+1 Phasen vor:

- *Planung* – Während der Planungsphase erfolgt eine Untersuchung von Art, Umfang und Machbarkeit einer Service-orient. Lösung im Unternehmen.
- *Analyse* – Die Analysephase beinhaltet die Bestimmung von Anforderungen an das Service-orient. Anwendungssystem.
- *Entwurf* – Die Entwurfsphase leistet eine Service-orient. Spezifikation von Lösungen, die die Anforderungen an das AS berücksichtigen.
- *Konstruktion* – In der Konstruktionsphase wird der Service-orient. Entwurf unter Beachtung der Analyse in ein lauffähiges Software-System überführt.
- *Test* – Die Testphase dient der Validierung und stellt sicher, dass die Anforderungen an das Service-orient. Anwendungssystem umgesetzt wurden.
- *Provision* – In die Provisionsphase fallen technische und geschäftliche Aspekte zur Unterstützung von Aktivitäten der Clients.
- *Deployment* – Die Phase des Deployment betrifft das Ausrollen neuer Prozesse durch Verteilen der Komponenten des Service-orient. Anwendungssystems auf die Teilnehmer.
- *Ausführung* – Die Ausführungsphase umfasst Mittel zu Durchsetzung des Ablaufs von Prozessen der Service-orient. Lösung bei allen Teilnehmern.
- *Monitoring* – Die Phase des Monitorings fasst Mittel zum Messen, Überwachen, Berichten und Verbessern von Qualitätsmerkmalen des Service-orient. Anwendungssystems zusammen.

Die beschriebenen Phasen bilden ein generelles Vorgehensmodell für verschiedene Rollen des Service-orient. Modells. Angewandt für die einzelnen Rollen gliedern sich hier dann noch die verschiedenen Rollenfunktionen ein. Hier sind insbesondere die Rollen von Client, Provider und Aggregator von Interesse. Diese sind anders als z. B. der Broker direkt mit der Erstellung eines Service-orient. Anwendungssystems befasst. Je nach Rolle stellt sich der Entwicklungslebenszyklus dabei unterschiedlich dar.

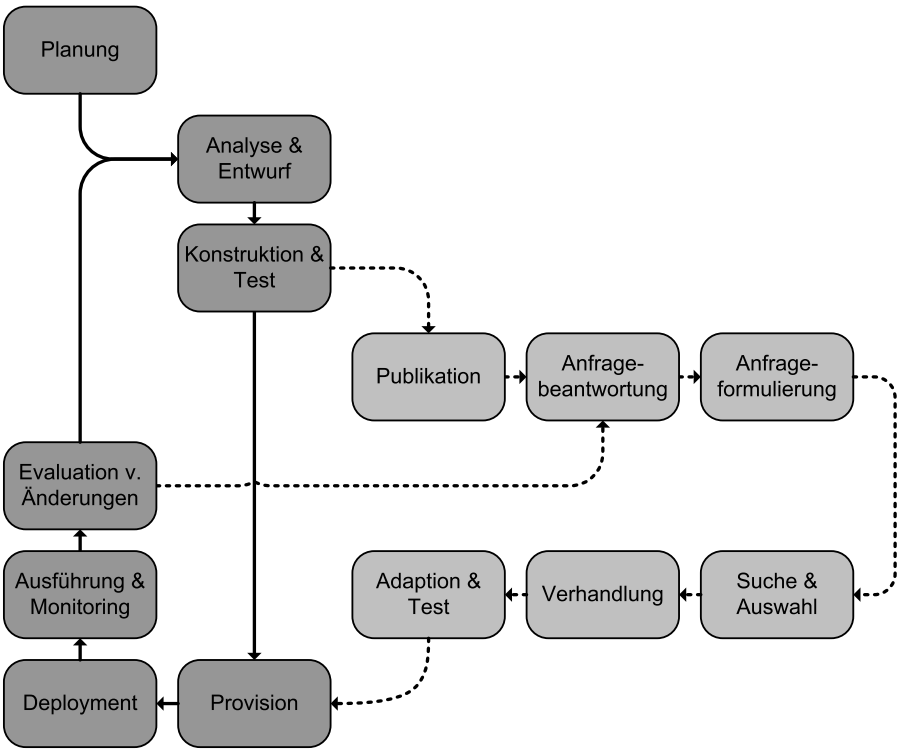

Abbildung 4.29.: Service-orient. Entwicklungslebenszyklus

Abb. 4.29 zeigt die Zusammenhänge der einzelnen Phasen des fundamentalen Service-orient. Entwicklungslebenszyklus. Zudem sind die zusätzlichen Phasen für die Rolle des Aggregators eingefügt. Diese Rolle bezieht die unterschiedlichen Phasen sowohl des Providers als auch des Clients ein und ist für die nachfolgenden Betrachtungen der vorliegenden Arbeit im Zusammenhang mit der E-Service-Technologie von besonderer Bedeutung. Ein Aggregator tritt sowohl als Provider in Bezug auf einen zusammengesetzten Web Service als auch als Client in Bezug auf WS-Komponenten auf. Der Lebenszyklus des zusammengesetzten Web Service startet mit Phasen der *Planung*, gefolgt von *Analyse und Entwurf* sowie *Konstruktion und Test*. Der generische Entwicklungslebenszyklus geht an dieser Stelle zur *Provision* über. Durch die spezielle Betrachtung der Aggregator-Rolle kommen jedoch zunächst noch einige spezifische Phasen hinzu.

Eine Besonderheit besteht darin, dass die Konstruktion an dieser Stelle nicht endgültig, sondern primär zu Testzwecken erfolgt. Das Ziel besteht zu diesem Zeitpunkt lediglich in einer abstrakten Definition der WS-Logik. Diese wird dann *publiziert*, so dass potenzielle Clients den WS finden können. Wenn dann vonseiten eines Clients eine Anfrage eingeht, erfolgt eine Überarbeitung und Konkretisierung des zusammengesetzten Web Service in Bezug auf die spezifischen Anforderungen. Diese Vorgehensweise wird z. B. im *Service Composition Lifecycle* nach Yang und Papazoglou beschrie-

ben [YP04]. Dieser unterscheidet die Phasen des *Planning* mit Synthese der WS-Logik, *Definition* als abstrakte Beschreibung des komponierten WS, *Scheduling* mit Analyse möglicher Verfeinerungen der Komposition im Kontext einer Anfrage, *Construction* als Zusammensetzung des WS für den spezifischen Fall einer Anfrage und *Execution* als Durchsetzung der konkreten Komposition. Eine Abwandlung dieses Verhaltensmusters soll auch in das hier betrachtete Referenzmodell eines Service-orient. Entwicklungslebenszyklus einfließen.

Im hier betrachteten Modell wird nach der *Anfrage* eines Clients eine Verfeinerung der WS-Komposition in Hinblick auf dessen Komponenten vorgenommen. Dazu erfolgt eine *Anfrage* nach, *Suche und Auswahl* von sowie *Verhandlung* über WS-Komponenten. Wenn dann die Komponenten feststehen, können *Adaption und Test* der WS-Komposition erfolgen. Danach liegt das Service-orient. Anwendungssystem vollständig vor und der Entwicklungslebenszyklus kann mit *Provision*, *Deployment*, *Ausführung* und *Monitoring* fortgesetzt werden. Schließlich kann noch eine Evaluation des bisherigen Lebenszyklus in Bezug auf die Notwendigkeit zur Änderung erfolgen. Ist dies der Fall, tritt der Zyklus in eine erneute Runde mit *Analyse und Design*. *Planung* ist in diesem Modell ein einmaliger Vorgang, der nicht wieder durchlaufen wird. Falls keine Änderungen erfolgen sollen, kann direkt die nächste *Beantwortung einer Client-Anfrage* erfolgen.

Service-orient. E-Service-Entwicklungslebenszyklus Nachdem in den vorangegangenen Paragraphen die Lebenszyklen von virt. Dienstleistungsproduktionsnetzwerken und Service-orient. Anwendungssystemen analysiert und in Vorgangsmodellen beschrieben wurden, kann nun die Konzeption eines kombinierten E-Service-Entwicklungslebenszyklus erfolgen. Die zusammenhängenden Lebenszyklen von VDPN und den darin produzierten Dienstleistungen geben hierbei die Anforderungen vor. VDPN basieren wiederum auf E-Services als Mitteln zur Regelung und Durchsetzung der Koordination kooperativer Produktionsvorgänge, d. h. dass der E-Service-Lebenszyklus diejenigen von VDPN und Dienstleistungen direkt beeinflusst. Die Anforderung an ein E-Service-Management-Vorgehensmodell besteht darin, dass diese Beeinflussung in der gewünschten Art geschieht, damit die idealtypischen Verläufe so gut wie möglich abgebildet und unterstützt werden. Die Mittel zur Realisierung des E-Service-Management-Vorgehensmodells sind von den verwendeten E-Service-Techniken und deren Entwicklungsmethoden abhängig. Im Falle Service-orient. E-Services gibt also der Service-orient. Entwicklungslebenszyklus die Mittel vor, die zur Unterstützung der Anwendungsvorgänge zur Verfügung stehen und entsprechend analysiert und angepasst werden müssen. Im Folgenden soll ein generelles Konzept für die Gestaltung von E-Service-Management-Vorgehensmodellen auf Basis des Service-orient. Entwicklungslebenszyklus hergeleitet werden. Das Ziel besteht darin, einen abstrakten Service-orient. E-Service-Entwicklungslebenszyklus zu definieren. Hierin soll ein generelles Prinzip der Anwendung Service-orient. Entwicklungsmethoden zur

Lebenszyklus virt. Unternehmen	**Lebenszyklus Service-orient. Anwendungssysteme**
Identifikationsphase	Planung
	Analyse
	Entwurf
	Konstruktion und Test
	Publikation
Anbahnungsphase	Anfragebeantwortung
	Anfrageformulierung
	Suche und Auswahl
Vereinbarungsphase	Verhandlung
	Adaption und Test
Operative Phase	Provision
	Deployment
	Ausführung und Monitoring
Auflösungsphase	Evaluation in Bezug auf Änderungen

Tabelle 4.4.: Vergleich der Lebenszyklusphasen von VU und Service-orient. AS

Entwicklung von E-Services erfasst werden. Dies soll als konzeptionelle Grundlage zur Ableitung von E-Service-Management-Vorgehensmodellen für spezifische Service-orient. E-Service-Modelle dienen.

Der erste Schritt hin zu einem abstrakten Service-orient. E-Service-Entwicklungslebenszyklus besteht in einer Gegenüberstellung der Lebenszyklusmodelle virt. Dienstleistungsproduktionsnetzwerke und Service-orient. Anwendungssysteme. Tabelle 4.4 veranschaulicht dies in einer Übersicht zusammenhängender Phasen. Die Tabelle stützt sich auf das grundlegende Lebenszyklusmodell für virt. Unternehmen nach Mertens und das erweiterte Lebenszyklusmodell für Service-orient. Anwendungssysteme nach Papazoglou. Das einfache Modell der virtuellen Organisationsform ist auf VDPN als Spezialform eines VU anwendbar und reicht an dieser Stelle aus, um die Zusammenhänge zu beschreiben. Damit wird auf der linken Seite der Tabelle ein Lebenszyklus vorgegeben, der durch denjenigen auf der rechten Tabellenseite abgebildet werden soll. Anschließend wird analysiert ob und wieweit dies gelingt.

Der VU-Lebenszyklus beginnt mit der *Identifikationsphase*. Die Entstehung eines VU ist unmittelbar mit den darin zu produzierenden Produkten verbunden. In dieser Phase entsteht zunächst die Idee zur Realisierung einer bestimmten Produktkategorie. Daraufhin werden die Gegebenheiten des Marktes und die Kompetenzen der Netzwerkunternehmen analysiert. ggf. werden neue Unternehmen in das Netzwerk aufgenommen, um fehlende Kompetenzen zu akquirieren. Als Nächstes erfolgt dann der Entwurf des Produkts und die Planung der Produktion. Im speziellen Fall der virt. Dienstleistungsproduktion im VDPN werden Dienstleistungsprodukte betrachtet. Diese sind im Wesentlichen durch ihren interaktiven Dienstleistungsprozess charakterisiert. Dienstleistungsvirtualisierung bildet den DLP auf E-Services ab. Modelle

und Methoden der E-Service-Techniken machen das Interaktionsmuster des DLP explizit und nutzen es zur Regelung der Koordination von kooperativen Tätigkeiten der Netzwerkunternehmen bei der Dienstleistungsproduktion aus. Im Falle Service-orient. E-Service-Techniken erfolgt dies auf Basis von entsprechenden Service-orient. Entwicklungsmethoden.

Die Vorgänge der VU-Identifikation können durch Aktivitäten verschiedener Phasen des Service-orient. Entwicklungslebenszyklus unterstützt werden. Dies fängt mit *Planung* und *Analyse* des E-Service an. Die Planung und Analyse von E-Services als Service-orient. Anwendungssystem geht hier unmittelbar mit den entsprechenden Aktivitäten der VU-Identifikation einher. Hier können vor allem die prozessorientierten Methoden der Service-orient. Analyse hilfreich sein. Diese beinhalten u. a. Aktivitäten zur Prozessidentifikation, Prozessabgrenzung, geschäftlichen Gap-Analyse auf Basis von „ist"- und „soll"-Prozessen sowie eine Analyse der Prozessrealisierung [PH06]. Die nächste Phase betrifft den *Entwurf* von E-Services. Dieser geht sowohl mit dem Produktentwurf als auch der Produktionsplanung im VU konform. Service-orient. Techniken erlauben hier die explizite Spezifikation der interaktiven Anteile des DLP. Dies beinhaltet im Wesentlichen u. a. die Kommunikationsschnittstellen der Netzwerkunternehmen und die Interaktionsmuster des DLP als Protokolle zur Regelung der Koordination. Zudem geben verschiedene Entwurfsprinzipien in Bezug auf Kopplung, Kohäsion und Granularität Hilfestellung beim Entwurf gut wartbarer und wiederverwendbarer Lösungen. Die anschließenden Phasen beinhalten *Konstruktion* und *Test* von E-Services. Die Implementierung von E-Services während der Identifikationsphase des VU stellt deren schnelle Einsatzbereitschaft in späteren Phasen sicher. Zudem wird dadurch das Testen der Lösungen in Bezug auf verschiedene Aspekte wie Funktion, Leistung, Stress, Volumen, Schnittstellen und Zusammensetzung vor dem eigentlichen Betrieb ermöglicht. Am Ende erfolgt die *Publikation* von E-Services, so dass diese am Markt in Erscheinung treten.

Die nächste Phase des VU betrifft die *Anbahnung*. Hierbei kommt ein Kontakt mit potenziellen Kunden zustande. Dieser Kontakt basiert auf einem Produktangebot das bei Bedarf durch das VU realisiert werden kann. Bei Interesse eines Kunden an dem Produkt werden potenzielle Netzwerkunternehmen bestimmt, die die Produktion im VU auf optimale Weise durchführen könnten. Die Phase beschränkt sich hierbei auf eine Vorauswahl potenzieller Teilnehmer. Übertragen auf die Phasen des Service-orient. Entwicklungslebenszyklus deckt sich dies zunächst mit der *Anfragebeantwortung* für WS-Clients in Bezug auf publizierte E-Service-Spezifikationen. Dies entspricht dem Kontakt zu Kunden des VU. Auf der anderen Seite unterstützt die Möglichkeit zur *Anfrageformulierung* an bzw. *Suche und Auswahl* von WS-Providern die Identifikation potenzieller Netzwerkunternehmen.

Im VU folgt nun die *Vereinbarungsphase*. Diese beinhaltet eine Verhandlung zwischen den Teilnehmern. Verhandlungsgegenstand sind die konkreten Bedingungen und Konditionen der Dienstleistungserbringung. Eine derartige *Verhandlungsphase* ist auch im Service-orient. Entwicklungslebenszyklus bedacht. Dazu existieren lediglich

vereinzelte Ansätze mit direktem Bezug zu den übrigen Service-orient. Techniken. Hierbei werden entweder Service-orient. Techniken als Grundlage zur Durchführung des Verhandlungsvorgangs benutzt oder es wird über die Inhalte Service-orient. Spezifikationen selbst verhandelt. Über solche Ansätze besteht jedoch bislang kein Konsens und es gibt keine entsprechende Standardisierung. Ein weiterer Aspekt der Vereinbarungsphase des VU ist die potenzielle Notwendigkeit zur Änderung des Dienstleistungsentwurfs als Folge ausgehandelter Bedingungen. Dies geht ggf. mit der Notwendigkeit zur Umstrukturierung des VU zur Produktion der Dienstleistung einher. Dieser Umstand wird durch eine Phase für *Anpassung und Test* von E-Services unterstützt.

An dieser Stelle geht das virt. Unternehmen in die *operative Phase* über. Hier beginnt die Produktion der Dienstleistung. Damit gehen Steueraktivitäten auf Netzwerk und Knotenebene einher. Die Unterstützung dieser Aktivitäten ist eine der primären Funktionen der E-Service-Technologie und wird dementsprechend auch durch die Phasen des Service-orient. Entwicklungslebenszyklus unterstützt. Als Vorbereitung der Produktionssteuerung findet ein *Deployment* der E-Service-Komponenten bei den verschiedenen konkreten VU-Teilnehmern statt. Die eigentliche Steuerung fällt dann in die Phase von *Ausführung und Monitoring*. Parallel sieht der E-Service-Lebenszyklus die Unterstützung von Clients durch verschiedene Maßnahmen im Rahmen der *Erbringung* vor. Hierzu gehören generische Aktivitäten, die orthogonale Aspekte wie z. B. Erfassung, Protokollierung, Messung und Berechnung von E-Services betreffen können [PH06].

Am Ende des VU-Lebenszyklus steht die *Auflösungsphase*. Hier werden Informationen über den Verlauf des Lebenszyklus und insbesondere über die operative Phase der Produktion ausgewertet und und verwertet. Die Ergebnisse werden abgelegt und für Folgeaktivitäten wie z. B. den Kundendienst oder die Prüfung von Gewährleistungsfragen sowie als Wissensbasis für weitere Lebenszyklen verwendet. Eine entsprechende Phase findet sich auch im Service-orient. Entwicklungslebenszyklus. Sie wird jedoch bislang durch keine verbreiteten Service-orient. Techniken direkt unterstützt.

Insgesamt zeigt die Gegenüberstellung der korrespondierenden Lebenszyklen eine grundsätzliche Konformität der Abläufe. Zudem lässt sich eine Überdeckung wesentlicher Lebenszyklusphasen virt. Unternehmen durch korrespondierende Phasen im Entwicklungslebenszyklus Service-orient. E-Services feststellen, die die Regelung und Durchsetzung der Koordination kooperativer Produktionstätigkeiten in VDPN unterstützen. Im Einzelnen erfahren besonders die VU-Phasen der Identifikation, Anbahnung und Operation durch die entsprechenden Service-orient. Methoden eine grundlegende Hilfestellung. Hier werden ausdrucksstarke Modelle und Sprachen zur expliziten Spezifikation der Dienstleistungsprodukte und zur Regelung der Koordination kooperativer Produktionstätigkeiten bereitgestellt. Diese bilden die unmittelbare Grundlage für automatisierte Methoden zur Vermittlung entsprechender Kompetenzen von Netzwerkunternehmen und Steuerung der Produktion.

Diese Instrumente bilden ein erhebliches Potenzial, um institutionalisierte Managementfunktionen im virt. Unternehmen zu ersetzen. Gleichwohl ist der generische Service-orient. Entwicklungslebenszyklus nicht als Instrument zur Regelung der virt. Dienstleistungsproduktion konzipiert worden. Entsprechend lassen sich auch verschiedene Aspekte identifizieren, die in Hinblick auf die Unterstützung des VU-Lebenszyklus nicht optimal ausfallen. Defizite sind vor allem in Hinblick auf die Flexibilität von Dienstleistungen und die Agilität des Entwicklungsprozesses im Verlauf von Identifikation, Anbahnung, Vereinbarung und Operation festzustellen. Im VU wird das Produkt und vor allem dessen Produktion in der Identifikationsphase noch nicht endgültig festgelegt. Dies erfolgt erst, wenn nach der Anbahnungsphase die Teilnehmer feststehen und in der Vereinbarungsphase konkrete Konditionen und Bedingungen ausgehandelt wurden. Erst dann können das spezifische Produkt und dessen Produktionsweise im VU im Detail festgelegt werden. Im generischen Service-orient. Entwicklungslebenszyklus erfolgen der Entwurf und die Konstruktion des E-Service schon sehr früh während der Identifikationsphase des VU. Dies ist zum einen notwendig, um eine explizite Spezifikation zur Regelung der Koordination vornehmen zu können, die als Grundlage für die darauf folgende Anbahnung dienen kann. Zum anderen muss nach der Anbahnung und Vereinbarung ein rascher Übergang in die operative VU-Phase erfolgen. Eine Konstruktion des E-Service zu diesem späten Zeitpunkt wäre im allgemeinen Fall zu langwierig. Die frühzeitige Konstruktion des E-Service beugt dem zwar vor, führt dafür aber zu Defiziten bei der Flexibilität. Eine Änderung des bereits entworfenen und konstruierten E-Service kann nämlich einerseits nur in beschränktem Umfang erfolgen. Andererseits kann dieser Vorgang auch wieder langwierig ausfallen. Hier wäre also eine Optimierung des Entwicklungslebenszyklus wünschenswert. Diese Optimierung müsste einerseits die Flexibilität von E-Services und andererseits die Agilität des Entwicklungsprozesses erhöhen.

Auf Basis der Gegenüberstellung und Analyse des zu unterstützenden VU- bzw VDPN-Lebenszyklus und des dazu potenziell anzuwendenden generellen Service-orient. Entwicklungslebenszyklus besteht der zweite und abschließende Schritt hin zu einem abstrakten Service-orient. E-Service-Entwicklungslebenszyklus in der Synthese der unterschiedlichen Aspekte. Dabei werden die identifizierten Defizite des generellen Service-orient. Entwicklungslebenszyklus aufgegriffen und die Perspektiven verschiedener Akteure des konzeptionellen VDPN-Modells differenziert. Das Ergebnis ist in Abb. 4.30 zu sehen.

Für die Konzeption des Service-orient. E-Service-Entwicklungslebenszyklus wird zunächst die betrachtete Lebenshistorie virtueller Dienstleistungsproduktion auf den Lebenszyklus von E-Services ausgeweitet. Dieser wird hierbei zunächst in einer technologieneutralen generischen Form entworfen. Hintergrund sind hierbei die angestrebte Unterstützungsfunktion des Produktionsmanagements und die daraus resultierenden Zusammenhänge mit den Lebenszyklen von VDPN und Dienstleistungen. Die Phasen einer Dienstleistung folgen prinzipiell denjenigen des VDPN mit kurzer Verzögerung.

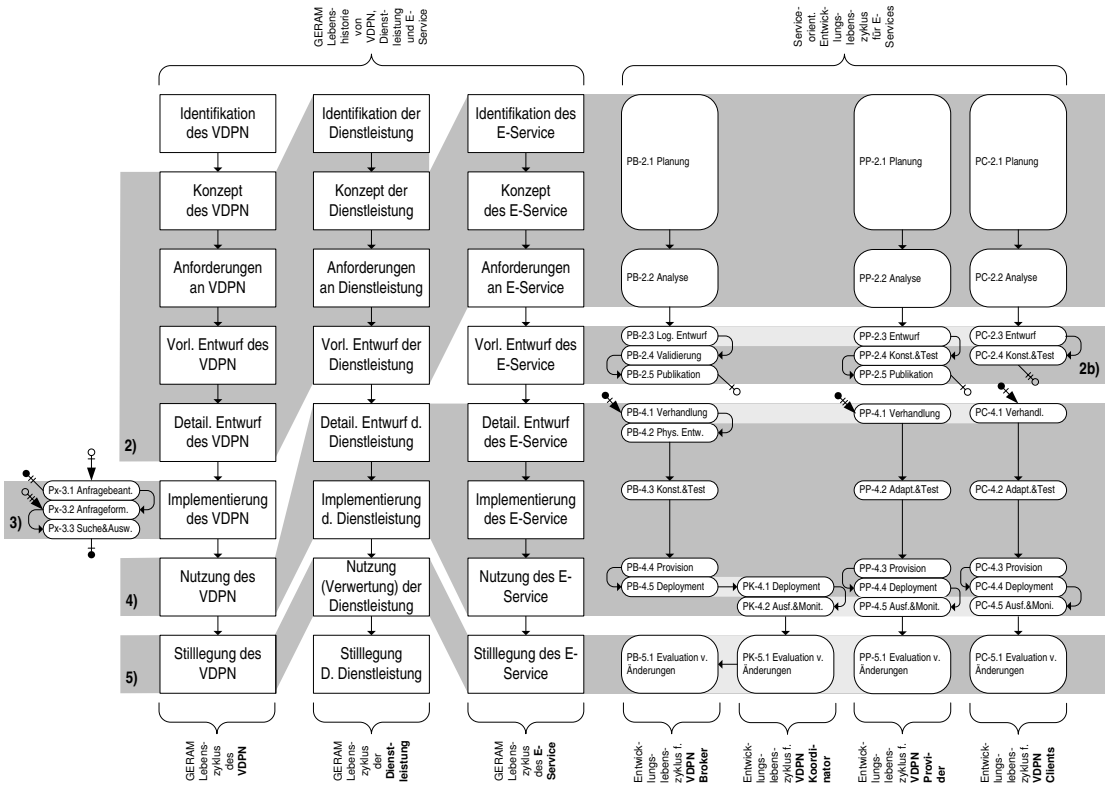

Abbildung 4.30.: Entwicklungslebenszyklus für Service-orient. E-Services

In gleicher Weise sind die Phasen eines entsprechenden E-Service naturgemäß unmittelbar mit denen von Dienstleistungen und VDPN verbunden. Die initialen Phasen der *E-Service-Identifikation*, *-Konzeption* und *-Anforderungsanalyse* gehen direkt aus korrespondierenden Phasen von Dienstleistungen und VDPN auf Stufe 2) hervor. Hier führt die Konzeption des VDPN zur Identifikation der Dienstleistung, deren Konzeption wiederum zur Identifikation des E-Service führt. Im weiteren Verlauf führen die Anforderungen der Dienstleistung zur Konzeption des E-Service. Der vorl. Entwurf der Dienstleistung beschreibt dann deren geschäftlichen Dienstleistungsprozess auf relativ abstrakter Ebene und bildet damit die Anforderungen an den E-Service. An dieser Stelle zeigt sich beim *vorl. E-Service-Entwurf* der erste Nutzeffekt der Dienstleistungsvirtualisierung für das Produktionsmanagement. Auf der erweiterten Stufe 2b) werden nämlich die geschäftlichen Prozesse der Dienstleistung in eine formale Form gebracht und exakt beschrieben. Dies erlaubt erweiterte Methoden: zum einen für den Entwurf von Dienstleistungen und zum anderen für die nachfolgenden Phasen der VDPN-Implementierung bzw. Anbahnung auf Stufe 3). Auf Stufe 4) erfolgt dann eine Nutzung des VDPN für den spezifischen Fall der Dienstleistungsproduktion mit individuellen Clients und Providern. Die Dienstleistung wird dabei für den speziellen

Fall im Detail entworfen und dann durch Ablauf des DLP implementiert. Dies wiederum basiert wesentlich auf den Phasen von *E-Service-Entwurf*, *-Implementierung* und *-Nutzung*. E-Services sind hier zunächst Subjekt der Verhandlung. Danach werden sie derart implementiert, dass letztendlich der Ablauf des DLP durch Ausführung der durch E-Services repräsentierten Interaktionsmuster gesteuert werden kann. Am Ende realisiert die E-Service-Stilllegung einen Aspekt der VDPN-Stilllegung. Konkret werden hier der Ablauf des Interaktionsmusters evaluiert und potenziell notwendige Änderungen am E-Service-Entwurf durchgeführt.

Für den derart vorgegebenen E-Service-Lebenszyklus wird dann ein Service-orient. Entwicklungslebenszyklus aufgestellt. Dieser Entwicklungslebenszyklus berücksichtigt insbesondere die verschiedenen Perspektiven der durch das konzeptionelle VDPN-Modell vorgegebenen Akteure. Namentlich sind das VDPN-Client, -Provider, -Broker und Koordinator.

Der VDPN-Client trägt zur Entwicklung des E-Service primär mit seinen Kommunikationsendpunkten bei, die eine Nachfrage des Clients nach Dienstleistungsinhalten repräsentieren. Diese bilden aber aus seiner Perspektive Komponenten eines kompletten Service-orient. AS. Dadurch werden die Interaktionen im Zuge der von ihm angefragten Dienstleistung mit seiner internen betrieblichen Informationsverarbeitung zusammengeführt. In diesem Sinne vollzieht der Client die komplette Entwicklung eines lokalen Service-orient. Anwendungssystems. Der entsprechende Entwicklungslebenszyklus ist vom allg. Service-orient. Entwicklungslebenszyklus abgeleitet. Eine erste Besonderheit liegt darin, dass die Entwicklung mit denjenigen der anderen Akteure synchronisiert werden muss. Am Ende muss sich nämlich aus den einzelnen Entwicklungen ein gemeinsames kooperatives Service-orient. Anwendungssystem zur Produktionssteuerung des VDPN ergeben. Daher erfolgt eine Koordination der beteiligten Akteure in der Entwicklungsphase.[41] Während der Implementierung des VDPN bzw. in der Anbahnungsphase des VU fragt der Client dann Dienstleistungen an, die zuvor von Brokern publiziert wurden. Er sucht dann eine zu seinen Kommunikationsendpunkten und sonstigen Anforderungen passende Dienstleistung aus. Der Rest der Entwicklung entspricht dem standardmäßigen Vorgehen.

Die Service-orient. Entwicklung beim VDPN-Provider geht weitgehend mit derjenigen des Clients konform. Nur, dass die Kommunikationsendpunkte in diesem Fall keine Nachfrage, sondern ein Angebot von Dienstleistungsinhalten repräsentieren. Auch dabei vollzieht sich die Entwicklung eines kompletten Service-orient. Anwendungssystem im Sinne des generellen Service-orient. Entwicklungslebenszyklus und dient zur Integration der Dienstleistungsinteraktionen mit der internen IV. Zudem muss die Entwicklung wiederum mit derjenigen anderer Teile des kooperativen Anwendungssystems synchronisiert werden. Der wesentliche Unterschied liegt in der VU-Anbahnungsphase bzw. VDPN-Implementierung. Hier publiziert der Provider seine

[41] Synchronisierte Phasen in den Entwicklungslebenszyklen verschiedener Akteure werden in der Abbildung mit hellen Balken hinterlegt.

Kommunikationsendpunkte, damit der VDPN-Broker sie finden und für die Planung des VDPN in Erwägung ziehen kann. Der Provider selbst führt im generellen Fall keine Suche nach WS-Komponenten durch. Der Rest der Entwicklung entspricht dem standardmäßigen Vorgehen.

Der letzte und interessanteste Lebenszyklus betrifft eine kombinierte Vorgehensweise von VDPN-Broker und Koordinator. Das hier entwickelte Service-orient. Anwendungssystem repräsentiert die Interaktionsmuster des DLP und realisiert dadurch die koordinative Ebene des kooperativen Anwendungssystems. Als zentraler Anteil des E-Service übernimmt der Broker mit seinem Entwicklungslebenszyklus eine leitende Rolle. In den Phasen der Planung und Analyse entsteht hier das Gesamtkonzept für ein Interaktionsmuster, das eine auf geschäftlicher Ebene vorgegebene Dienstleistung repräsentiert und dadurch ein kooperatives Anwendungssystem zur Produktionssteuerung im VDPN koordiniert. Die Leitungsfunktion macht sich dann als Erstes in der Entwurfsphase bemerkbar. In dieser gibt der Broker durch den Entwurf der Interaktionsmuster als Koordinationsprotokolle den Kontext für den Entwurf von Kommunikationsendpunkten aufseiten von Clients und Providern vor. Dabei wird der generelle Service-orient. Entwicklungslebenszyklus in Bezug auf die weiter oben identifizierten Defizite abgeändert. Hier erfolgt in der Identifikationsphase noch kein vollständiger Entwurf und überhaupt keine Konstruktion des Service-orient. Anwendungssystems. Stattdessen erfolgt lediglich ein logischer Entwurf, der eine Validierung vitaler Eigenschaften sowie eine Publikation zum Zweck der Koordination und als Dienstleistungsangebot erlaubt. Die Anbahnungsphase verläuft dann weitgehend im Sinne des standardmäßigen Vorgehensmodells. Erst nach den Verhandlungen in der Vereinbarungsphase erfolgen der physikalische Entwurf und die Konstruktion des Service-orient. Anwendungssystems. Dies hat den Vorteil, dass das Interaktionsmuster bis zu seinem endgültigen Einsatz maximal flexibel bleibt. Hierdurch entsteht jedoch ein Problem in Bezug auf die Agilität der Verfahrensweise. Dies muss durch entsprechende Methoden des physikalischen Entwurfs und der Konstruktion kompensiert werden. Die Charakteristik des hier betrachteten Service-orient. Anwendungssystems als Interaktionsmuster, das sich auf Basis von WS-Komposition implementieren lässt, macht diesen Entwicklungsprozess jedoch grundsätzlich möglich. Insbesondere können hier nämlich generative Methoden den physikalischen Entwurf und die Konstruktion erheblich beschleunigen.[42] Im Verlauf des Entwicklungsprozesses findet sich eine weitere Besonderheit. Sie geht darauf zurück, dass der Broker im konzeptionellen VDPN-Modell die Steuerungsfunktion an einen oder mehrere Koordinatoren abgibt. In diesem Sinne durchläuft der Broker lediglich eine Deployment-Phase und übernimmt danach nicht selbst die Ausführung. Der Koordinator seinerseits führt keine Entwicklung im eigentlichen Sinne durch, sondern übernimmt deren Ergebnisse in einer Deployment-Phase vom Broker. Die Koordinatoren führen dann jedoch parallel

[42] Solche Methoden wurden z. B. von Orriens vorgeschlagen [OYP03] und werden im weiteren Verlauf noch eingehend betrachtet.

zu Client und Providern die zentralen integrativen Komponenten des ganzheitlichen kooperativen Anwendungssystems aus und erledigen dabei auch die Überwachung der Abläufe. Am Ende findet eine Evaluation der Abläufe in Zusammenarbeit mit dem Broker statt, der die Teilergebnisse zusammenfasst.

4.4.2. E-Service-Management in FRESCO

In den vorangegangenen Sektionen wurden Grundlagen für die Integration der E-Service-Technologie in den betrieblichen Kontext gelegt. Dies geschah durch Eingliederung von E-Service-Management in den Kontext des Enterprise Engineering und Anwendung des GERAM-Referenzmodells zu dessen Strukturierung. Hierbei zeigte sich die Bedeutung einer ganzheitlichen Repräsentation aller geschäftlich relevanten Aspekte verschiedener Bereiche und Ebenen mit ihren zusammenhängenden Lebenszyklen im Rahmen einer Enterprise-Architektur, um daran Modelle, Methoden und Werkzeuge des Enterprise Engineering auszurichten. In Übertragung auf das E-Service-Management wurden daraufhin die zusammenhängenden Lebenszyklen der wichtigsten Konzepte virt. Dienstleistungsproduktion analysiert. Daraus wurde ein abstrakter Entwicklungslebenszyklus für Service-orient. E-Services als Grundlage für spezifische E-Service-Management-Vorgehensmodelle abgeleitet.

Zur weitergehenden Konkretisierung von Entwicklungsprozessen und -methoden eines Service-orient. E-Service-Management-Vorgehensmodells müssen konkrete VDL-Konzepte und Techniken zu deren Realisierung als Service-orient. E-Services zugrunde liegen. Theoretische Methoden und formale Sprachen innerhalb verschiedener Phasen sowie die Phasenstruktur eines E-Service-Entwicklungslebenszyklus basieren nämlich unmittelbar auf der VDPN-Architektur als spezieller Enterprise-Architektur, die durch ein spezifisches VDL-Modell vorgegeben wird. Sie lassen sich zudem am besten im Zusammenhang mit den Eigenschaften der Techniken und Instrumente konzipieren, die hinter den Kommunikationsendpunkten und Interaktionsmustern zur Realisierung virt. Vorgänge und Abläufe des virt. Dienstleistungsprozesses stehen. Hierauf kann dann eine Werkzeugunterstützung einzelner oder zusammenhängender Lebenszyklusphasen aufbauen. Eine solche Werkzeugunterstützung ist auf Grund der Komplexität der abzubildenden Prozesse und der geforderten Agilität des Entwicklungsvorgangs von entscheidender Bedeutung.

Die Konkretisierung des abstrakter Entwicklungslebenszyklus für Service-orient. E-Services soll nun exemplarisch auf Basis der Ansätze im FRESCO-Projekt veranschaulicht werden. Hierbei werden zunächst in Sektion 4.4.2.1 die notwendigen Spezialisierungen eines Service-orient. Entwicklungslebenszyklus für FRESCO-E-Services abgeleitet. Danach wird in Sektion 4.4.2.2 ein entsprechendes FRESCO-spezifisches E-Service-Management-Vorgehensmodell (FSMV) vorgestellt. Hierbei werden auch die damit einhergehenden Sprachen und Werkzeuge eingeführt. Diese bilden Bestandteile des technischen Rahmenwerks, dessen Entwicklung dann im nächsten Kapitel untersucht wird.

4.4.2.1. Spezialisierung des abstrakten E-Service-Entwicklungslebenszyklus

Im Rahmen des FRESCO-Projekts werden virt. Dienstleistungen und deren Realisierung als E-Services auf Basis Service-orient. Techniken in konkreten konzeptionellen Modellen und Metamodellen exakt definiert. Das konzeptionelle FRESCO-VDL-Modell gibt die Eigenschaften des spezifischen Dienstleistungsbegriffs und dessen Anwendung für das Produktionsmanagement im VDPN vor. Dies stellt eine Spezialisierung der VDPN-Architektur für das FRESCO-Rahmenwerk dar, die sich sowohl auf deren Realisierung als E-Services auswirkt als auch auf ein damit einhergehendes E-Service-Management. Das FRESCO-SOM und das FRESCO E-Service-Metamodell geben die entsprechenden Service-orient. Konzepte vor, die zur Realisierung virt. Dienstleistungen als E-Service verwendet werden sollen. Dadurch werden jedoch nicht nur die Techniken zur Implementierung von E-Service bestimmt, sondern auch ein Rahmen möglicher Entwicklungsprozesse und -methoden vorgegeben. Insgesamt ergeben sich also Anforderungen des zu unterstützenden konzeptionellen Dienstleistungsmodells und Vorgaben des damit einhergehenden Service-orient. E-Service-Modells, die einer Spezialisierung des abstrakten Service-orient. E-Service-Entwicklungslebenszyklus zugrunde liegen.

Organisatorische Anforderungen des VDL-Modells Das konzeptionelle VDL-Modell bzw. FRESCO-Dienstleistungsmodell beruht auf dem Prinzip der Trennung von Dienstleistungsinhalt, -erbringung und -konsum. Dies drückt sich in den entsprechenden Konzepten von Assets, Capabilities und Demands aus, die spezielle Aspekte eines virt. Dienstleistungsprozesses definieren. Der erbringungsorientierte FRESCO-Ansatz fokussiert insbesondere Capabilities als Perspektive der Dienstleistungserbringung. Diese Perspektive stellt die Interaktionsabläufe des VDLP als administrative Vorgänge zum Zugang von Dienstleistungsinhalten auf Basis von Dienstleistungsnachfrage dar. In diesem Zusammenhang spezifiziert das konzeptionelle FRESCO-Dienstleistungsmodell eine spezielle Art der Regelung der Koordination kooperativer Produktionsaktivitäten in VDPN. Hierbei teilt sich die Regelung im Hinblick auf lokale und globale Aspekte in einen kollaborativen Vorgang zwischen FRESCO-Provider und -Broker auf. Provider und Broker übernehmen jeweils für ihren Bereich die Produktionsplanung. Darüber hinaus erben sie die entsprechenden Aufgaben der Produktionssteuerung von der Rolle des Koordinators, die hier in ihrer expliziten Form entfällt. Diese Besonderheiten des FRESCO-Dienstleistungsmodells bilden eine Spezialisierung der VDPN-Architektur, die beim Entwurf eines Entwicklungslebenszyklus für FRESCO E-Services berücksichtigt werden muss.

Die Anforderungen des FRESCO-Dienstleistungsmodells resultieren in der Notwendigkeit zur Verfeinerung der Service-orient. Entwicklungslebenszyklen für E-Services aus den Perspektiven von Provider und Broker. FRESCO-Provider übernehmen in Bezug auf die Entwicklung von Assets zunächst die im abstrakten Lebenszyklus vorgegebene Verfahrensweise zur Entwicklung von Kommunikationsendpunkten. Zu-

sätzlich besteht hier die Anforderung zur Regelung der Koordination kooperativer Produktionsaktivitäten in Bezug auf lokale Assets. Dazu müssen FRESCO-Provider einen zweiten Entwicklungslebenszyklus für Capabilities unterstützen. Die Entwicklung von Capabilities folgt im Wesentlichen der Verfahrensweise des Brokers im abstrakten Lebenszyklus, die sich dort auf die Entwicklung von Interaktionsmustern bezieht. Verfeinerungen dieses Entwicklungslebenszyklus werden nicht nur für den FRESCO-Provider, sondern auch für den FRESCO-Broker benötigt. Dabei muss sich die Verfeinerung zum einen auf die besonderen Eigenschaften von Capabilities als spezifisches Subjekt der Entwicklung beziehen. Zum anderen muss die Kollaboration zwischen Broker und Provider bei der Entwicklung zusammenhängender Capabilities beachtet werden. Hierbei muss die Abstimmung zwischen der Entwicklung globaler und lokaler Capabilities im Entwicklungsprozess spezifiziert werden. Eine weitere Notwendigkeit zur Anpassung ergibt sich daraus, dass FRESCO-Broker und -Provider die generelle Rolle des Koordinators erben. Die Abläufe und Vorgänge zur Nutzung von E-Services durch Ausführung implementierter Interaktionsmuster müssen daher in die Entwicklungslebenszyklen von FRESCO-Broker und Provider integriert werden.

Schließlich können im Kontext der schwerpunktmäßigen Betrachtungen in FRESCO noch einige Vereinfachungen des Entwicklungslebenszyklus von Capabilities vorgenommen werden. Die Aspekte der Planung und Analyse werden in FRESCO als Vorgabe der geschäftlichen Ebene vorausgesetzt und daher komplett ausgeklammert. Verhandlung, Test, Monitoring und Evaluation werden dagegen als orthogonale Aspekte betrachtet, die sich bei Bedarf integrieren lassen, aber nicht explizit behandelt werden. Außerdem ist Provision im FRESCO-Dienstleistungsmodell ein integraler Bestandteil des Capability-Konzepts und entfällt daher als separate Phase im Entwicklungslebenszyklus.

Im Folgenden wird insbesondere der so spezialisierte Entwicklungsprozess für E-Service Capabilities in den Vordergrund gestellt, der sich in Kollaboration von FRESCO-Broker und -Provider vollzieht. Parallele Entwicklungsprozesse von Assets und Demands durch FRESCO-Provider und -Clients folgen im Wesentlichen den generellen Verfahrensweisen Service-orient. Entwicklung und werden daher nur am Rande betrachtet.

Methodische Vorgaben des E-Service-Modells Der Entwicklungslebenszyklus von FRESCO E-Service Capabilities durch Broker und Provider wird auch von den Vorgaben des FRESCO-SOM und FRESCO E-Service-Metamodell bestimmt. Das FRESCO-SOM beschreibt ein SOA-Konzept zur Realisierung von E-Services. Hier werden Capabilities auf Konzepte der WS-Koordination und -Komposition abgebildet. Zum einen werden Capabilities als WS-Interaktionsprotokolle umgesetzt. Diese spezifizieren Interaktionen im Rahmen des von der Capability repräsentierten administrativen Prozesses als WS-Aufrufe. Dadurch bilden sie eine Vorgabe für Entwicklungsprozesse von Assets und Demands und regeln deren Kooperation im Kontext der Dienstleis-

tungsproduktion. Zum anderen werden Capabilities als WS-Orchestrierungsschemata realisiert. Diese implementieren das Interaktionsmuster zwischen Teilnehmern der Capability, das durch das WS-Interaktionsprotokoll vorgegeben wurde. So können die Interaktionen durch Capability-WS ausgeführt werden und es ergibt sich eine automatisierte Produktionssteuerung. Die so skizzierte SOA wird durch das FRESCO E-Service-Metamodell exakt spezifiziert. Hieraus ergeben sich Vorgaben für die Wahl Service-orient. Methoden zur kombinierten Entwicklung von WS-Interaktionsprotokollen und -Orchestrierungsschemata.

Auf Basis der technischen Vorgaben müssen passende Instrumente für die generellen methodischen Kategorien im abstrakten Service-orient. E-Service-Entwicklungslebenszyklus gewählt werden. In Übertragung des generellen Konzepts von FRESCO-E-Services auf die Phasenstruktur des Service-orient. Entwicklungslebenszyklus erfolgt die Regelung der Kooperation im Kontext der Dienstleistungsproduktion durch Methoden zum logischen Entwurf, zur Validierung und zur Publikation sowie später zur Änderung von WS-Interaktionsprotokollen. Die Implementierung und Ausführung von Interaktionen erfolgt hingegen durch Methoden zum physikalischen Entwurf, zur Konstruktion, zum Deployment und zur Ausführung von WS-Orchestrierungsschemata. In Bezug auf diese methodischen Kategorien ergibt sich ein wesentlicher Ansatzpunkt zur Wahl einer konkreten Methodik durch die Bestimmung einer Strategie zur Service-orient. Prozessintegration.[43]

In Bezug auf die Regelung der Kollaboration fällt die Wahl auf eine freie Variante der Regelung von Interaktionen zwischen FRESCO-Broker und -Providern. Hierbei erfolgen ein manueller Entwurf und Abgleich von Koordinationsprotokollen durch Provider und Broker. Dafür ist zunächst eine passende Entwurfsmethode für entsprechende Koordinationsprotokolle zu entwickeln. Zum Abgleich kann dann eine Hilfestellung vonseiten der Validierung erwartet werden. Die Validierung von WS-Koordinationsprotokollen auf Basis von z. B. Konversationen und Choreografien wird zurzeit intensiv untersucht und bildet ein eigenständiges Forschungsumfeld. Eine solche Validierung dynamischer Protokolleigenschaften geht über den Rahmen der Entwicklung in FRESCO hinaus. Es sollten jedoch strukturelle Prüfungen, z. B. in Bezug auf die Syntax der Modellierungssprache, möglich sein. Eine weitere Methode muss die Änderung von Koordinationsprotokollen unterstützen. Dies ist ein wesentlicher Aspekt in der Auflösungsphase von VDPN. Zudem kann eine Änderung vorläufiger Entwürfe im Zuge der Vereinbarungsphase nötig werden. Die hierfür in FRESCO anvisierte Methode soll vor allem eine Änderung von Capabilities ermöglichen, von denen im E-Service-Schema potenziell sehr viele existieren. Entsprechend soll die Methode die automatisierte und systematische Änderung unterstützen. Dazu soll hier ein regelbasierter Transformationsmechanismus zum Einsatz kommen.

In Bezug auf die Implementierung kollaborativer Regelungen soll in FRESCO eine automatisierte Variante gewählt werden. Dies soll insbesondere die Agilität in der

[43] Siehe Diskussion in Sektion 3.4.3.1 und Sektion 4.3.1.2

Vereinbarungsphase des VDPN fördern. Hier können dann nämlich im Anschluss an eine potenzielle Verhandlung noch Änderungen am Koordinationsprotokoll vorgenommen werden, die dann auf Basis automatischer Methoden quasi unmittelbar in eine ausführbare Form überführt werden können. Die operative Phase des VDPN kann dann ohne Verzögerung starten. Die automatische Implementierung soll hier in Form von WS-Kompositionsschemata erfolgen und sich deren Nähe zu den WS-Koordinationsprotokollen zunutze machen. Hierdurch kann der physikalische Entwurf als Modelltransformation vom WS-Koordinationsprotokoll zum WS-Kompositionsschema konkretisiert werden. Die Implementierung muss dann noch Methoden zur Konstruktion und zum Deployment von Software-Komponenten beinhalten, die auf den WS-Kompositionsschemata basieren. Diese Komponenten bilden Engines, die die anschließende Ausführung des E-Service maßgeblich antreiben. Es sind jedoch noch zusätzliche Mechanismen zur Verwaltung von Ressourcen der E-Service-Teilnehmer und verschiedener E-Service-Instanzen zu konzipieren.

4.4.2.2. FRESCO E-Service-Management-Vorgehensmodell

In Zusammenfassung der vorangegangenen Sektion bezieht sich der in FRESCO betrachtete E-Service-Entwicklungslebenszyklus auf spezifische Aspekte der Erstellung, Ausführung und Verwertung von FRESCO E-Service Capabilities durch FRESCO-Broker und -Provider. FRESCO-SOM und FSM geben dabei die Realisierung von Capabilities auf Basis von Web Service-Koordination und -Komposition vor und bestimmen dadurch Vorgaben für die Entwicklungsmethodik. Die Anordnung dieser methodischen Vorgaben zu Phasen eines spezifischen Entwicklungslebenszyklus erfolgt in Bezug auf die Anforderungen des konzeptionellen FRESCO-VDL-Modells. Die Umsetzung einer derartigen Spezialisierung des abstrakten Service-orient. E-Service-Entwicklungslebenszyklus führt zu den spezifischen Entwicklungsprozessen, Entwicklungsmethoden, Modellierungssprachen und Werkzeugen eines konkreten FRESCO E-Service-Management-Vorgehensmodells (FSMV) [ZBL03, ZLP04]. Dieses Vorgehensmodell wird in Abb. 4.31 im Kontext der Lebenshistorie virtueller Dienstleistungsproduktion gezeigt.

Der im FSMV betrachtete E-Service-Entwicklungsprozess startet mit dem kollaborativen Entwurf von Capabilities. Zuvor sind bereits verschiedene Phasen der Identifikation des virt. Dienstleistungsproduktionsnetzwerks durchlaufen worden. Für diese hier nicht explizit betrachteten Phasen wird vorausgesetzt, dass dort eine virtuell zu produzierende Dienstleistung identifiziert, konzipiert und analysiert wird. Insbesondere erfolgt dabei der Entwurf der Dienstleistung als Geschäftsprozess.[44] Dieser bildet die zentrale Anforderung an den FRESCO E-Service. Die darauf folgenden Phasen des E-Service-Entwicklungslebenszyklus werden nun erläutert.

[44]Die Entwicklung dieser Geschäftsprozesse sollte zweckmäßiger weise einem eigenständigen Vorgehensmodell folgen, wie es z. B. in Sektion 2.3.1.2 beschrieben wurde.

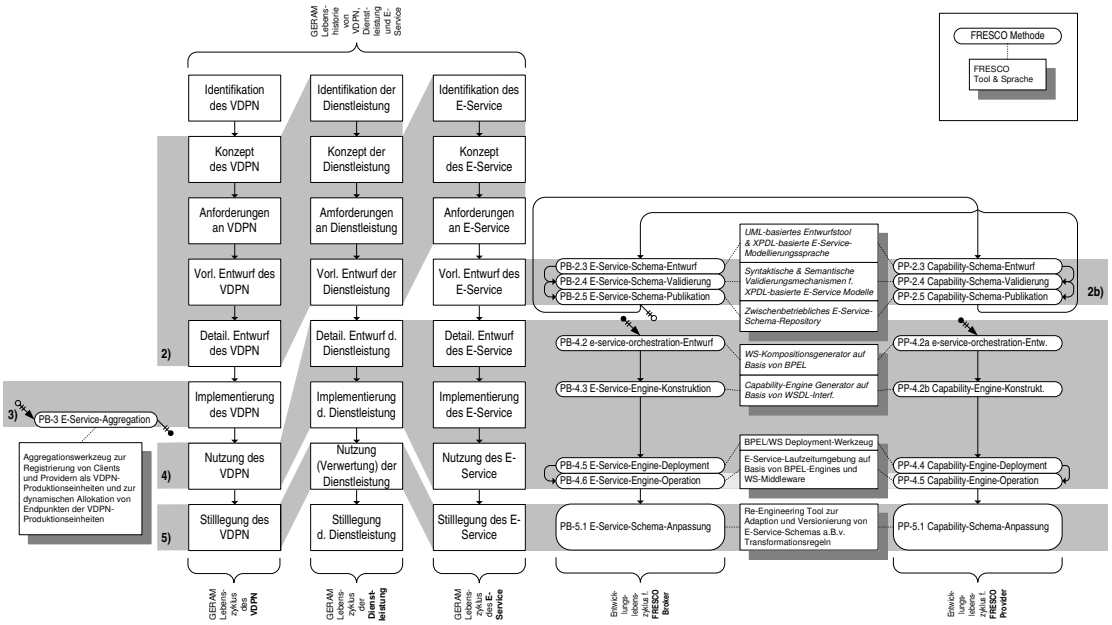

Abbildung 4.31.: FRESCO E-Service-Management-Vorgehensmodell

- *Schema-Entwurf (PB-2.3, PP-2.3)* – Der initiale Schritt der E-Service-Entwicklung besteht in einer Überführung des geschäftlichen Dienstleistungsprozesses in die formale Form von E-Service-Schemata. Konkret werden an dieser Stelle die verschiedenen Interaktionen zwischen Providern und Kunden sowie deren Abläufe als Interaktionsprozesse in Form von E-Service Capabilities spezifiziert. Dabei wird zwischen Interaktionsmustern im Zusammenhang mit der Erbringung einzelner Assets und im Zusammenhang mit der Kombination verschiedener Assets zu einer ganzheitlichen Dienstleistung unterschieden. Der Entwurf von *Capability-Schemata* in Bezug auf Assets erfolgt durch Provider. Der Entwurf globaler Capabilities sowie deren Einbindung in ein gemeinsames *E-Service-Schema* erfolgt durch den Broker. In beiden Fällen dient eine XPDL-basierte Modellierungssprache nach Vorgabe des FRESCO E-Service-Metamodells zur Spezifikation von Capabilities. Für diese Sprache existiert ein UML-Profil. Dies erlaubt den grafischen Entwurf mittels eines UML-Entwurfswerkzeugs.

- *Schema-Validierung (PB-2.4, PP-2.4)* – Im Anschluss an den Entwurf einer Capability erfolgt eine Validierung. Fundamental kann dabei eine Prüfung der Syntax mitsamt grundlegender Konsistenzkriterien erfolgen. Darüber hinaus ist es möglich, die Konformität des Entwurfs mit den strukturellen Vorgaben des FRESCO E-Service-Metamodells zu testen. Der Validierungsvorgang erfolgt in automatisierter Weise durch die FRESCO-Werkzeuge.

- *Schema-Publikation (PB-2.5, PP-2.5)* – Validierte Schemata von E-Services und Capabilities werden zunächst im Dienstleistungsnetzwerk veröffentlicht. Auf Stufe 2b) der virt. Dienstleistungsproduktion wird hierdurch der Abgleich verschiedener Schemata ermöglicht. Der kollaborative Entwurfsvorgang erfolgt dabei in Mikrozyklen zwischen Broker und Providern. Anfangs erfolgt der Entwurf eines E-Service-Schemas durch den Broker auf Basis der Anforderungen des geschäftlichen Dienstleistungsprozesses. Dieses Schema dient als Vorgabe für den Entwurf Asset-spezifischer Capability-Schemata durch die Provider. Provider veröffentlichen ihre Capability-Schemata ihrerseits zur Kenntnisnahme durch den Broker. Der Broker kann dann auf Basis der internen Angebote ggf. das E-Service-Schema modifizieren. Dieser Mikrozyklus setzt sich fort, bis aus Sicht des Brokers ein zufriedenstellendes Ergebnis erreicht ist. Die Veröffentlichung von E-Service- und Capability-Schemata erfolgt mit Werkzeugunterstützung durch das Web-basierte FRESCO Schema-Repository.

- *Service-Aggregation (PB-3)* – Nach Beendigung des E-Service-Schema-Entwurfs geht die virt. Dienstleistungsproduktion auf Stufe 3) über. Sofern es sich bei der Produktion nicht um eine Auftragsfertigung handelt, erfolgt an dieser Stelle eine Veröffentlichung des E-Service-Schemas als Angebot auf einem passenden elektronischen Marktplatz. Dieser Fall wird in FRESCO nicht explizit unterstützt. In jedem Fall erfolgt hier jedoch eine Auswahl von Providern zur Realisierung von Assets und deren Erbringung durch Asset-spezifische Capabilities. Die Registrierung von Teilnehmern als Produktionseinheiten des virt. Dienstleistungsproduktionsnetzwerks geschieht in FRESCO mithilfe eines Aggregationswerkzeugs. Hierbei werden Client und Provider an Rollen in E-Service Capabilities gebunden, durch die deren Assets bzw. Demands festgelegt werden.

- *Orchestrierungsentwurf (PB-4.2, PP-4.2a)* – Nachdem die Teilnehmer des VDPN feststehen, schreitet die virt. Dienstleistungsproduktion auf Stufe 4) fort. Die Stufe beinhaltet zunächst die VU-Vereinbarungsphase mit möglichen Verhandlungen über Dienstleistungsinhalte und -erbringung sowie sonstige Eigenschaften. Solche Verhandlungen werden in FRESCO nicht explizit unterstützt. Deren Ergebnis kann hier jedoch durch Änderung des Schemas einbezogen werden. Der wesentliche Beitrag dieser Phase liegt in der Überführung des vorläufigen bzw. logischen Entwurfs in einen Detailentwurf bzw. physikalischen Entwurf. Während der logische Entwurf auf dem FSM basiert und in Form von generischen E-Service-Schemata erfolgt, basiert der physikalische Entwurf auf einer konkreten WS-Kompositionstechnik und erfolgt in Form von Orchestrierungsschemata. In FRESCO wird BPEL als exemplarische Kompositionstechnik betrachtet. Ein entsprechendes Transformationswerkzeug generiert BPEL-basierte Orchestrierungsschemata aus XPDL-basierten E-Service-Schemata.

- *Engine-Konstruktion (PB-4.3, PP-4.2b)* – Der physikalische Entwurf des E-Service bildet die Basis zur anschließenden Konstruktion eines operativen Systems zur zwischenbetrieblichen Steuerung der virtuellen Dienstleistungsproduktion. Die Architektur dieses Systems wird durch das FRESCO-SOM vorgegeben und beinhaltet im Wesentlichen Asset, Demand und Capability Web Services. Hierbei orchestrieren die Capability Web Services Aufrufe der Asset und Demand Web Services. Asset und Demand Web Services werden als Enterprise Modules von Client und Providern eingebracht. Die verbleibende Konstruktion bezieht sich auf Capability Web Services. Dies geschieht in FRESCO durch Konfiguration von E-Service Engines und Generierung von deren Kommunikationsendpunkten. Exemplarisch basiert dies für den Fall von BPEL-Orchestrierungsschemata auf entsprechenden BPEL Engines. Der Vorgang wird durch ein Werkzeug zum Engine-Management unterstützt, das die komplette Konstruktionsphase automatisiert.

- *Engine Deployment (PB-4.5, PP-4.4)* – Nach Konstruktion der E-Service Engine-Konfigurationen und Kommunikationsendpunkte von Capability Web Services werden diese beim Broker und bei den Providern installiert. Dies erfolgt in FRESCO ebenfalls mit Werkzeugunterstützung. Hierbei leitet wiederum das Engine-Management-Werkzeug den Deployment-Prozess an, so dass am Ende ein unmittelbar ablauffähiges verteiltes Anwendungssystem zur Produktionssteuerung vorliegt.

- *Engine Operation (PB-4.6, PP-4.5)* – An dieser Stelle ist der E-Service implementiert und das VDPN geht in die operative Phase über. Gleichzeitig beginnt die E-Service Engine des Brokers mit der Steuerung der Dienstleistungsproduktion. Hierbei erfolgt eine Orchestrierung von Interaktionen zwischen den Capability Web Services teilnehmender Provider. Die E-Service Engines verschiedener Provider werden dadurch in den Steuerprozess einbezogen. Sie übernehmen die Orchestrierung von Interaktionen zwischen den Assets des Providers und den Demands des Clients. Neben den verschiedenen BPEL-basierten E-Service Engines kommt während des E-Service-Betriebs noch ein Laufzeitmechanismus zur Verwaltung der vielfältigen Kommunikationsendpunkte eines E-Service zum Einsatz. Hierbei erfolgt in FRESCO die Allokation von Kommunikationsendpunkten der Clients und Provider in dynamischer Weise durch das Aggregationswerkzeug. Auf diese Weise wird gleichzeitig die Verwaltung multipler E-Service-Instanzen realisiert.

- *Schema-Anpassung (PB-5.1, PP-5.1)* – Nach Beendigung der Steueraktivitäten in den verschiedenen Engines von Providern und Brokern ist die Erbringung der Dienstleistung abgeschlossen. Die virtuelle Dienstleistungsproduktion geht damit in die abschließende Stufe 5) über. Hier erfolgt zunächst eine Auswertung

der Betriebsphasen von VDPN und E-Service. Diese basiert in FRESCO im Wesentlichen auf den Funktionen der verwendeten E-Service Engines. Insbesondere die E-Service Engine des Brokers kann hier z. B. mit der Durchlaufzeit des gesamten Steuerprozesses wichtige Anhaltspunkte über den Verlauf der Dienstleistungserbringung liefern. Diese Daten bilden die Grundlage zur Evaluation potenzieller Änderungen am E-Service-Schema. Die Durchführungen von Änderungen werden in FRESCO durch ein Re-Engineering-Werkzeug für E-Service-Schemata unterstützt. Hiermit werden Änderungen als Transformationsregeln spezifiziert, die in automatischer und konsistenzerhaltender Weise auf komplexe Schemata mit zahlreichen Capabilities angewandt werden können. Ein weiterer unterstützter Aspekt ist die Versionierung von E-Service-Schemata im Verlauf ihrer Evolution. Hiermit wird sichergestellt, dass Informationen über erbrachte Dienstleistungen auch nach der Auflösung des VDPN erhalten bleiben.

Das so beschriebene FSMV organisiert Provider eines Dienstleistungsnetzwerks zu virt. Dienstleistungsproduktionsnetzwerken, die dem idealtypischen Lebenszyklus eines virt. Unternehmens folgen. Dabei ermöglicht das Service-orient. FRESCO E-Service-Metamodell eine Reihe automatisierter Methoden, mit denen die Produktionsplanung und -steuerung im Sinne der virtuellen Organisationsform flexibel und agil gestaltet werden kann. Das FRESCO E-Service-Management bildet so ein Instrument zum Einsatz im Rahmen eines ganzheitlichen Enterprise Engineering virt. Dienstleistungsunternehmen.

4.5. Zusammenfassung

Im Kapitel 4 wurden grundlegende Ansätze zur virtuellen Produktion von Dienstleistungen in virt. Dienstleistungsunternehmen und zur zwischenbetrieblichen Prozessintegration durch Service-orient. Techniken einer IV-Infrastruktur für virt. Dienstleistungsunternehmen zusammengeführt. Auf dieser Basis wurde ein konzeptionelles Rahmenwerk zur Unterstützung des Managements der Dienstleistungsproduktion in virt. Dienstleistungsunternehmen durch Virtualisierung von Dienstleistungen selbst entwickelt. Hierbei wurde auf der einen Seite eine allgemeine Theorie virtueller Dienstleistungsproduktion in virt. Dienstleistungsproduktionsnetzwerken auf Basis des Entwicklungslebenszyklus virtualisierter Dienstleistungen aufgestellt. Auf der anderen Seite wurde die abstrakte Theorie durch Ausgestaltung der allgemeingültigen Konzepte als konkrete Modelle in eine praktische Form überführt.

Die theoretische Ausarbeitung des konzeptionellen Rahmenwerks leistet einen Beitrag in Bezug auf die grundlegenden Aspekte der Dienstleistungsvirtualisierung in Form von Konzepten für I) die konzeptionelle Modellierung virt. Dienstleistungen und ihre Produktion in virt. Dienstleistungsproduktionsnetzwerken, II) die Realisierung virt. Dienstleistungen durch E-Services und E-Service-Management auf Basis integrativer IV-Techniken im Allgemeinen und SOC-Techniken im Speziellen sowie III) die

Integration des E-Service-Managements in den ganzheitlichen Kontext des Enterprise Engineering auf Basis eines abstrakten Service-orient. E-Service-Entwicklungslebenszyklus. Im Rahmen des konzeptionellen Modells wurden mit VDL und VDPN die zwei grundlegenden Konzepte der Dienstleistungsvirtualisierung definiert und in einen funktionalen Zusammenhang gestellt, der die virtuelle Dienstleistungsproduktion begründet. Mit den Konzepten von E-Service und E-Service-Management wurde eine Strategie zur technischen Realisierung des konzeptionellen Modells entworfen, die auf einer strukturierten Anforderungsanalyse beruht und sich durch allgemeine Richtlinien eines generischen Vorgehensmodells zur VDL-Realisierung sowie durch deren grundsätzliche Anwendung auf das SOC-Paradigma manifestiert. Die Gestaltung des E-Service-Managements als Facette des Enterprise Engineering im Rahmen des GERAM-Referenzmodells begründet die Integration virtueller Dienstleistungsproduktion in den betrieblichen Gesamtkontext und ermöglichte die systematische Erarbeitung eines abstrakten E-Service-Management-Vorgehensmodells durch Optimierung des Service-orient. Entwicklungslebenszyklus.

Die exemplarische Betrachtung eines praktischen Ansatzes zur Dienstleistungsvirtualisierung im Kontext des FRESCO-Projekts leistet einen Beitrag in Bezug auf die Veranschaulichung der theoretischen Betrachtungen und die Bereitstellung detaillierter Konzepte für die praktische Umsetzung virtueller Dienstleistungsproduktion in Form I) eines konzeptionellen Modells virtueller VDL-Produktion, II) eines Service-orient. E-Service-Metamodells sowie III) eines Service-orient. E-Service-Management-Vorgehensmodells. Das konzeptionelle FRESCO-Dienstleistungsmodell liefert einen dienstleistungszentrierten und erbringungsorientierten Ansatz zur Strukturierung virt. Dienstleistungsprozesse. FRESCO-SOM und FRESCO E-Service-Metamodell beschreiben ein detailliertes SOA-Konzept, das den Service-orient. Entwurf von E-Services in einer technikneutralen Weise erlaubt. Mit dem FRESCO E-Service-Management-Vorgehensmodell liegt eine konkrete Anleitung für die Anwendung von Entwicklungsprozessen, -methoden und -werkzeugen für FRESCO-E-Services als praktische Instrumente des Enterprise Engineering virt. Dienstleistungsunternehmen vor.

5. Realisierung einer Service-orient. Plattform für E-Services

5.1. Zielsetzung

In Kapitel 5 geht die Untersuchung der vorliegenden Arbeit von der Konzeption zum ersten Teil der Realisierung über. Nachdem im letzten Kapitel der konzeptionelle Anteil für ein Rahmenwerk zur Unterstützung virt. Dienstleistungsproduktion ausgearbeitet wurde, besteht das Ziel nun in der Realisierung des technischen Anteils dieses Rahmenwerks. Der technische Anteil soll IV-Techniken zur Realisierung von E-Services und E-Service-Management umfassen. Um die Gesamtheit dieser IV-Techniken zu erfassen, soll die Bezeichnung *E-Service-Management-System* wie folgt definiert werden:

Definition 19 (E-Service-Management-System) *Der Begriff des E-Service-Management-Systems (ESMS) bezeichnet ein technisches Rahmenwerk zusammenhängender IV-Techniken zur Unterstützung des Produktionsmanagements in VDU. Dies beinhaltet zum einen eine Ausführungsplattform für die Interaktionsvorgänge und -abläufe eines E-Service als Teil der vertikalen IV-Infrastruktur in VDU. Zum anderen beinhaltet dies Mechanismen und Werkzeuge zur Realisierung von Entwicklungsprozessen und -methoden eines E-Service-Management-Vorgehensmodells als Teil horizontaler Fachfunktionen in VDU.*

Im weiteren Verlauf soll sich die Untersuchung der vorliegenden Arbeit auf die Realisierung Service-orient. ESMS konzentrieren. Diese gliedert sich in die Aspekte des technischen Entwurfs und der Implementierung. Sektion 5.2 thematisiert zunächst den technischen Entwurf von ESMS. Dieser beinhaltet den Entwurf von E-Service-Entwicklungsmechanismen und -Ausführungsplattformen. Beides soll jeweils für die generelle Form Service-orient. E-Services sowie die konkrete Form von FRESCO E-Services untersucht werden. Für den Entwurf von Entwicklungsmechanismen wird in Hinblick auf die Anforderungen des E-Service-Entwicklungslebenszyklus[1] ein möglichst integrierter Ansatz von E-Service-Entwurf und -Implementierung angestrebt. Das Ziel ist es, hier einen effizienten Übergang auf Basis generativer Mechanismen zu entwerfen, der eine agile Entwicklung begünstigt. Für die resultierenden Service-orient. E-Service-Implementierungen ist dann eine Ausführungsplattform zu entwerfen. Dabei ist die Interaktion zwischen VDPN-Teilnehmern mittels Web Services und die Steuerung virt. Dienstleistungsprozesse durch Web Service-Kompositionen

[1] Siehe Sektion 4.4.2.1.

einzubeziehen. Konkret sollen hier auf Grund der besseren Repräsentation von Teilnehmerressourcen Grid-basierte WS-Erweiterungen untersucht und in Hinblick auf ein E-Service Grid eingesetzt werden.

Als zweiter Schritt der ESMS-Realisierung sind die Entwürfe als Software-System zu implementieren. Hierbei soll eine fokussierte Untersuchung für den Fall von FRESCO-E-Services erfolgen. Sektion 5.3 dokumentiert die prototypische Implementierung der FRESCO E-Service-Plattform und -Methodik als FRESCO Toolkit (FRESCO-TK). Dies betrifft zunächst Basistechniken der horizontalen IV-Infrastruktur im virt. Dienstleistungsunternehmen, die so weit wie möglich auf COTS-Komponenten basieren soll. Als wesentliche Bestandteile müssen hier eine WS-basierte OGSA-Infrastruktur sowie eine Ausführungsumgebung für BPEL-Kompositionsschemata eingebunden werden. Diese Basisinfrastruktur soll dann zu einem technischen Rahmenwerk für die Entwicklung der spezifischen Grid Service-Komponenten von E-Service-Implementierungen erweitert werden. Zudem sollen auf dieser Ebene Metaprotokolle zur Regelung der Interaktion von Komponenten der FRESCO-Plattform und Werkzeugen der FRESCO-Methodik vorgegeben werden. Auf Basis dieser horizontalen Infrastruktur sollen dann die vertikalen Fachfunktionen der FRESCO-Entwurfsmethodik und Plattformmechanismen als Komponenten und -Werkzeuge erfolgen.

5.2. Entwurf eines E-Service-Management-Systems (ESMS)

Die Betrachtungen zur Realisierung Service-orient. E-Service-Management-Systeme beginnen mit der Untersuchung des technischen Entwurfs solcher Systeme. Der technische Entwurf erfolgt dabei in Bezug auf die davon abzugrenzenden konzeptionellen Entwürfe für Service-orient. E-Services und das darauf aufbauende E-Service-Management, die im vorherigen Kapitel ausführlich beschrieben wurden. Die dort ausgearbeiteten Modelle bilden also die Grundlage des technischen Entwurfs von ESMS. Die konzeptionellen Modelle virtueller Dienstleistungsproduktion definieren die Anwendungsperspektive und bilden einen generellen Rahmen der Akteure und Vorgänge, die durch das System wiederzugeben sind. Dies spiegelt sich in den Service-orient. E-Service-Metamodellen und E-Service-Management-Vorgangsmodellen wider. Hierin wird in groben Zügen der generellen Betrachtung bzw. in detaillierten Modellen des FRESCO-Ansatzes festgelegt, wie das Konzept virt. Dienstleistung und einer darauf basierenden Produktionssteuerung in virt. Dienstleistungsproduktionsnetzwerken auf Technikkategorien bzw. konkrete Techniken des Service Oriented Computing abgebildet werden soll.

Auf Basis der konzeptionellen Vorgaben soll zunächst in Sektion 5.2.1 der Entwurf Service-orient. E-Service-Entwicklungsmechanismen untersucht werden. Das Ziel besteht in der Ausarbeitung eines technischen Konzepts zur Umsetzung von Entwicklungsprozessen und -methoden Service-orient. E-Service-Management-Vorgehensmodelle. Hierfür wird zunächst ein genereller Ansatz zur Integration von

Entwicklungsmechanismen auf Basis von Modelltransformation erstellt. Dann werden entsprechende Mechanismen zum Entwurf und zur Transformation von E-Service- und WS-Kompositionsschemata in FRESCO vorgestellt.

Als zweiter Aspekt wird in Sektion 5.2.2 der Entwurf einer IV-Infrastruktur zur Ausführung der entwickelten E-Services untersucht. Eine solche Infrastruktur steht in direkter Beziehung zu den Modellen für virt. Dienstleistungen und E-Services auf konzeptioneller Ebene. Die im konzeptionellen VDL-Modell bestimmten Kernaspekte virtueller Vorgänge und Abläufe müssen durch einen technischen Entwurf von Infrastrukturmechanismen im Sinne der im E-Service-Metamodell vorgegebenen Realisierungskonzepte durch WS-basierte Interaktion, Koordination und Komposition als integrierte Plattform zur Ausführung von E-Services umgesetzt werden. Hierfür erfolgt zunächst eine Untersuchung der beiden Kernaspekte der E-Service-Ausführung und deren Umsetzung als Service-orient. Systemplattform. Dann erfolgt als Beispiel der Entwurf einer Ausführungsplattform für FRESCO-E-Services.

5.2.1. Entwurf von E-Service-Entwicklungsmethoden

Die Untersuchung des ESMS-Entwurfs beginnt im Folgenden mit Mechanismen und Werkzeugen zur Implementierung der Entwicklungsmethodik im Rahmen von E-Service-Management-Vorgangsmodellen. Dies ist insofern sinnvoll, als die hierbei betrachteten Methoden den Service-orient. E-Service-Entwicklungslebenszyklus einleiten und auch über dessen weiteren Verlauf wesentliche Eckpunkte markieren. Die im Anschluss untersuchten Mechanismen zur Ausführung von E-Services bilden hingegen einen einzelnen zentralen Punkt des Entwicklungslebenszyklus. Es muss jedoch beachtet werden, dass Entwicklungsmechanismen und Ausführungsplattform naturgemäß eng zusammenhängen und zwar sequenziell betrachtet werden können, aber nicht voneinander isoliert werden dürfen.

Der Entwurf von E-Service-Entwicklungsmechanismen beginnt in Sektion 5.2.1.1 zunächst mit der Untersuchung eines Konzepts, das einen generellen Lösungsweg zur Erfüllung der konzeptionellen Anforderungen liefert. Aus konzeptioneller Sicht übernimmt der E-Service-Entwicklungsprozess die Rolle eines Plans zur Regelung der Koordination kooperativer Produktionsaktivitäten virt. Dienstleistung. Hierbei bilden Flexibilität und Agilität wesentliche nicht-funktionale Anforderungen im Anwendungskontext virt. Dienstleistungsunternehmen. Diese Eigenschaften sollen durch eine modellgetriebene Entwicklung von E-Services erreicht werden.

Das generelle Konzept modellgetriebener Entwicklung Service-orient. E-Services wird dann in Sektion 5.2.1.2 als Methodik für das FRESCO E-Service-Management exemplarisch umgesetzt. Hierbei werden integrierte Mechanismen für den logischen Entwurf von E-Service-Modellen und die automatische Generierung physikalischer Entwürfe in Form von WS-Kompositionsschemata entwickelt. Ferner werden Mechanismen zur Änderung von E-Service-Modellen durch regelbasierte Transformation entworfen.

5.2.1.1. Integrierte E-Service-Entwicklungsmethodik

Enterprise-Architekturen basieren generell auf einer komplexen hierarchischen Struktur vielfältiger Modelle zur Beschreibung von Unternehmen. Dies gilt auch für die in der vorliegenden Arbeit betrachtete Facette betrieblicher Architekturmuster für VDPN. Hierfür existieren verschiedene Modelle von VDPN auf organisatorischen und technischen sowie konzeptionellen und konkreten Meta- und Instanzebenen, die im logischen, kausalen und temporalen Zusammenhang der Lebenszyklen verschiedener VDPN-Bestandteile stehen. Auf den Modellen basieren Methoden zur dispositiven und operationalen Unternehmensführung, deren Anwendung durch Vorgehensmodelle geregelt wird. In diesem Sinne wurden im vorangegangenen Kapitel auf konzeptioneller Ebene Modelle virt. Dienstleistungen und Methoden zur VDPN-Produktionssteuerung sowie auf technischer Ebene Modelle von E-Services und Methoden des E-Service-Management erarbeitet. In einem entsprechenden E-Service-Management-Vorgehensmodell wurden die Zusammenhänge von Modellen und Methoden aus dem generellen Service-orient. Entwicklungslebenszyklus entwickelt. Für die im Vorgehensmodell beschriebenen Entwicklungsmethoden und Prozesse sollen nun Mechanismen und Werkzeuge entworfen werden.

Auf Grund der bei Enterprise-Architekturen stark ausgeprägten Modellhierarchie bietet sich hierfür unmittelbar eine modellgetriebene Vorgehensweise der Entwicklung an. Hierbei bildet Modellierung das zentrale Element der Entwicklung. Die schrittweise Verfeinerung von Modellen bis hin zu ausführbarem Code basiert dann ganz wesentlich auf (teil-)automatisierten Transformationsmechanismen. Durch diese Art der Konstruktion wird der Entwicklungsprozess stark beschleunigt. Außerdem können Änderungen auf verschiedenen Modellebenen mit geringem Konstruktionsaufwand umgesetzt werden. Insgesamt fördert eine modellgetriebene Vorgehensweise also die Agilität und Flexibilität der Entwicklung. In Hinblick auf die Anforderungen der virtuellen Organisationsform nach schneller Realisierung und häufiger Änderung der Organisationsstrukturen bietet sich der modellgetriebene Ansatz unmittelbar zur Implementierungen der Methoden des E-Service-Managements an. Im Folgenden wird daher zunächst ein kurzer Überblick modellgetriebener Entwicklung gegeben. Danach erfolgt die Ausarbeitung einer generellen Methodik zur modellgetriebenen E-Service-Entwicklung.

Modellgetriebene Entwicklung Der Begriff der modellgetrieb. Entwicklung (MDD) bezeichnet eine spezifische Art der Software-Entwicklung auf Basis von a) domänenspezif. Modellierungssprache mit hohem Abstraktionsgrad und b) Transformationsmechanismen und Generatoren zur automatisierten Auswertung von Modellen und Synthese von Artefakten eines Software-Systems [Sch06].

Eine domänenspezif. Modellierungssprache (DSML) gibt die Struktur, das Verhalten und die Anforderungen einer Klasse von Anwendungssystemen unmittelbar durch ihre Sprachkonstrukte vor. DSMLs basieren auf Metamodellen, in denen die wesentlichen

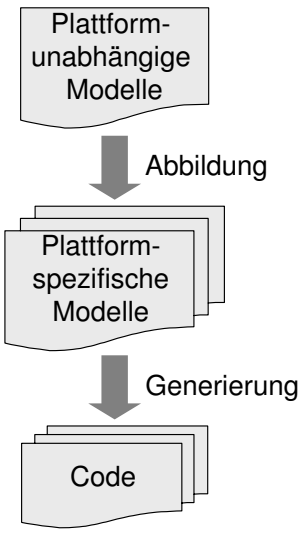

Abbildung 5.1.: Modellgetriebener Entwicklungsprozess

Konzepte und Beziehungen einer Anwendungsdomäne mitsamt ihrer Semantik und ihren Einschränkungen spezifiziert werden. Im Entwicklungsprozess werden DSML dann zum Entwurf von Anwendungssystemen verwendet, wobei die Systemeigenschaften in deklarativer Weise festgelegt werden. Die daraus hervorgehenden Modelle repräsentieren Anwendungssysteme nicht nur auf dem hohen Abstraktionsniveau der (z. B. geschäftlichen) Anwendungsdomäne, sondern sind auch weitgehend unabhängig von konkreten Techniken. Sie bilden die zentralen Bestandteile der Entwicklung und sind als langlebige, gut wartbare und wiederverwendbare Artefakte ausgelegt.

Die Implementierung dieser Modelle erfolgt oft mithilfe von Transformationsmechanismen und Generatoren. Derartige Mechanismen werten die Modelle von Anwendungssystemen aus und synthetisieren vielfältige Artefakte zur Konstruktion ausführbarer Software-Systeme. Dazu gehören z. B. Code, Deployment-Anweisungen oder auch alternative Repräsentationen der Modelle. Durch die automatisierte Konstruktionsmethoden wird sichergestellt, dass die resultierenden Software-Systeme den aus der Analyse hervorgegangenen und im Modell deklarierten funktionalen und nichtfunktionalen Eigenschaften entsprechen. Hierbei werden Fehler der Konstruktion komplett ausgeschlossen. Fehler bei der Modellierung können in einigen Fällen durch Verifikation von Modellen identifiziert werden. Zudem wird der Konstruktionsvorgang wesentlich beschleunigt.

Die Methoden und Vorgehensweisen der modellgetriebenen Entwicklung sind u. a. durch die modellgetriebene Architektur (MDA) der Object Management Group (OMG) [MM01] standardisiert worden.[2] In deren Terminologie können bei modellgetrieb.

[2]Im Folgenden wird der MDA-Standard zur Erläuterung grundlegender Konzepte und in Bezug auf dessen Terminologie verwendet. Es wird in der vorliegenden Arbeit jedoch kein strikt

Entwicklung grundsätzlich plattformunabhäng. Modelle (PIM) und plattformspezif. Modelle (PSM) unterschieden werden (Abb. 5.1). PIM beschreiben ein Anwendungssystem unter kompletter Abstraktion der technischen Plattform zu deren späteren Implementierung. Auf diese Weise sind PIM zwischen Plattformen verschiedener Organisationen sowie verschiedenen Plattform-Generationen der gleichen Organisation portierbar. Entsprechend eigenen sich PIM zur Erfassung von Entwürfen, die als dauerhafte Standards in oder zwischen Organisationen ausgelegt sind. Zur Implementierung von PIM erfolgt zunächst die Transformation in ein plattformspezif. Modell. Diese Transformation geschieht nach vorgegebenen Regeln der Abbildung, die in Bezug auf die PIM- und PSM-Metamodelle definiert werden. Dabei findet eine Konkretisierung der generischen Beschreibung des Anwendungssystems bezüglich einer technischen Plattform statt, auf der die Implementierung basieren soll. Im PSM wird dann spezifiziert, wie die Konzepte des PIM mit den Konzepten der technischen Plattform zu realisieren sind. Auf Basis der konkreten Beschreibung im PSM kann schließlich eine Generierung von Code-Artefakten erfolgen, auf deren Basis das Anwendungssystem ausgeführt werden kann.

Modellgetriebene E-Service-Entwicklung Die skizzierte Verfahrensweise modellgetriebener Entwicklung soll nun auf die E-Service-Entwicklung übertragen werden. Hierzu müssen die spezifischen Metamodelle und Modelle sowie Abbildungen und Transformationen im Sinne des Service-orient. E-Service-Entwicklungslebenszyklus konkretisiert werden. Eine entsprechende Modellhierarchie wird in Abb. 5.2 gezeigt.

Der im vorangegangenen Paragraphen beschriebene Entwicklungsprozess findet sich in der mittleren Zeile wieder, in der die Modellebene dargestellt wird. Das plattformunabhäng. Modell ist hier durch ein E-Service-Modell gegeben. Hierin wird der virt. Dienstleistungsprozess ohne Bezug auf die zu seiner Realisierung verwendeten integrativen IV-Techniken spezifiziert. Wie dieses Modell zu entwerfen ist, wird durch ein entsprechendes E-Service-Metamodell definiert, dessen Instanz das E-Service-Modell ist. Für das E-Service-Metamodell wird auch eine Abbildung auf das Metamodell eines plattformspezif. Modell definiert. In der vorliegenden Arbeit ist dies ein Metamodell Service-orient. Interaktion und insbesondere von WS-Koordination und -Komposition. Auf Basis der Abbildung kann eine Transformation des E-Service-Modells in WS-Koordinations- und -Kompositionsmodelle erfolgen, die dann Instanzen des Service-orient. PSM-Metamodells darstellen. Im weiteren Verlauf können PSM dann in konkrete Artefakte in Form von WS-Schnittstellenbeschreibungen sowie -Koordinationsprotokollen und -Kompositionsschemas transformiert werden. Je nach E-Service-Management-Vorgehensmodell gehen die Koordinationsprotokolle wiederum als Vorgabe für Provider oder Broker in deren Entwicklungsprozess ein. Schnittstellenbeschreibungen und Kompositionsschemata können hingegen direkt als Artefakte zur Ausführung von Interaktionsprozessen genutzt werden. Nachdem bisher

MDA-konformer Ansatz verfolgt.

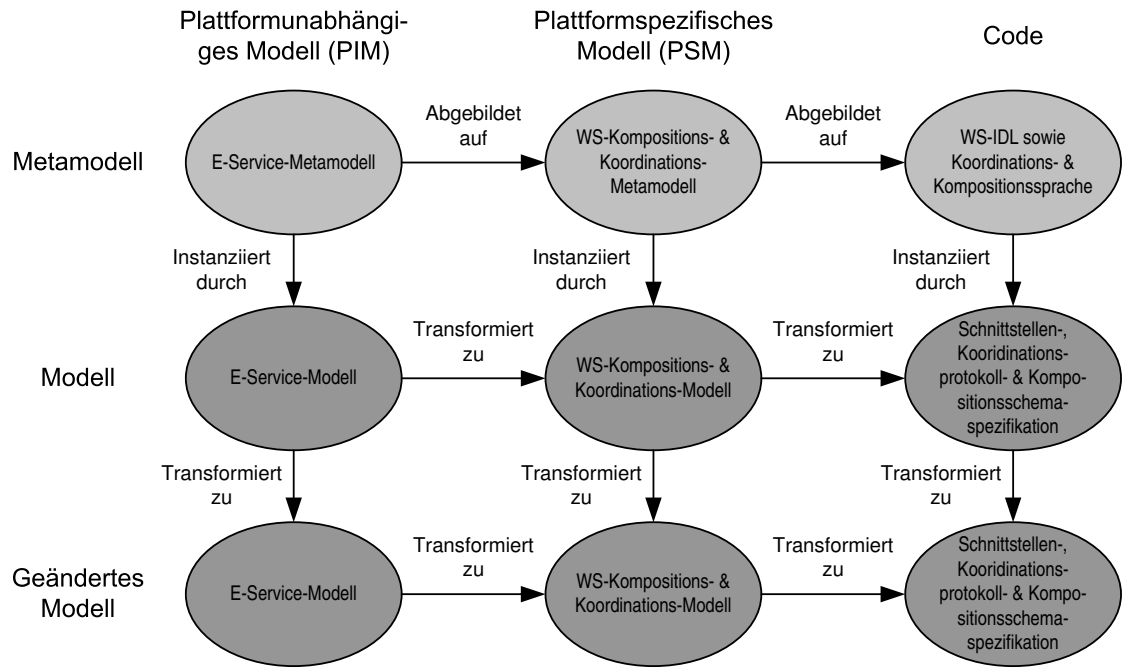

Abbildung 5.2.: E-Service-Entwicklung als integrierte Modelltransformation

die Phasen von Entwurf und Konstruktion abgedeckt wurden, besteht eine weitere wichtige Phase des Service-orient. E-Service-Entwicklungslebenszyklus in der Änderung. Diese kann bei modellgetriebener Entwicklung wiederum durch Transformation erfolgen. Änderungen sind hierbei als Transformationsregeln für Modelle zu spezifizieren und resultieren dann in geänderten Modellen. Dies kann auf verschiedenen Abstraktionsebenen geschehen. Auf Code-Ebene ist die Transformation für ad hoc-Änderungen denkbar. Langfristige Änderungen sollten hingegen auf Basis einer PIM-Transformation vorgenommen werden. Schließlich bietet sich eine Transformation von PSM an, wenn Änderungen im Zusammenhang mit der technischen Plattform vorgenommen werden sollen.

In Bezug auf den allg. Service-orient. E-Service-Entwicklungslebenszyklus[3] stellen der Modellentwurf sowie die Transformationsmechanismen und Generatoren Entwicklungsmethoden dar. Diese konkretisieren die methodischen Kategorien des abstrakten Entwicklungslebenszyklus in wesentlichen Phasen. Der Entwurf von E-Service-Modellen schließt sich an die Analyse an. Zusammen mit einer möglichen Validierung und der Transformation in Koordinationsprotokolle, die publiziert und zwischen Broker und Providern ausgetauscht und abgeglichen werden können, ergibt sich eine Methodik für die Phase des vorläufigen E-Service-Entwurfs (Stufe 2b der VDPN-Lebenshistorie). Die Transformation von E-Service-Modellen in Service-orient.

[3]Vgl. Sektion 4.4.1.2.

Kompositionsmodelle sowie die Generierung von Kompositionsschemata sowie von weiteren Artefakten einer spezifischen technischen Plattform der Web Service-Komposition bilden dann eine Methodik des physikalischen Entwurfs und automatisierten Konstruktion (Stufe 4 der VDPN-Lebenshistorie). Die Ausführung beruht dann auf den Mechanismen der technischen Ausführungsplattform, die in Sektion 5.2.2 untersucht werden. Schließlich bilden Transformationsmechanismen zur Änderung von Modellen auf verschiedenen Abstraktionsebenen einen Mechanismus zur Umsetzung der E-Service-Änderung (Stufe 5 der VDPN-Lebenshistorie). Insgesamt liefert der modellgetriebene Ansatz somit Mechanismen für die wesentlichen methodischen Kategorien des E-Service-Entwicklungslebenszyklus. Der Ansatz soll daher im Folgenden am Beispiel des FRESCO E-Service-Management-Vorgehensmodells konkret ausgeführt werden.

5.2.1.2. FRESCO-Methodik

In FRESCO werden die Modellstrukturen durch das FRESCO E-Service-Metamodell spezifiziert. Das FSM gibt zunächst durch sein E-Service-Metamodell die konzeptionellen Elemente eines E-Service-Modells vor. Ferner wird die Beziehung der E-Service-Modellelemente zu den Elementen Service-orient. Koordinations- und Kompositionsmodelle definiert, die im Service Coordination Metamodel und Service Composition Metamodel beschrieben werden. Schließlich stehen die E-Service-, Service Coordination- und Service Composition-Metamodelle in fundamentaler Beziehung zum Workflow Reference Model. Hierdurch wird nicht nur das grundlegende Prozessmodell festgelegt, sondern auch die Repräsentation von Modellen auf Basis der XML-basierten Workflow-Spezifikationssprache XPDL. Daneben wird das generische Service Composition Metamodel in der vorliegenden Arbeit exemplarisch mit dem BPEL-Ansatz in Beziehung gesetzt, dessen Spezifikation auf Basis eines XML-Schemas vorliegt. Übertragen auf die Hierarchie der modellgetriebenen E-Service-Entwicklung entsprechen E-Service-Modelle dem plattformunabhäng. Modell. Service-orient. Koordinations- und Kompositionsmodelle bilden hingegen plattformspezif. Modelle. Die Code-Ebene der Modellhierarchie wird maßgeblich durch WSDL und BPEL bestimmt.

Abbildung 5.3 fasst die Übertragung der modellgetriebenen E-Service-Entwicklung auf die konkreten FRESCO-Ansätze zusammen. Im oberen Teil sind die beschriebenen Metamodelle dargestellt. Darunter werden die bei der modellgetriebenen Entwicklung von FRESCO-E-Services verwendeten spezifischen Modelle, Formate, Transformatoren und Generatoren gezeigt. Hier sind auch die durch Rechtecke markierten Funktionsbereiche zu sehen, deren Mechanismen in FRESCO durch eigenständige Werkzeuge realisiert werden. Dies betrifft zunächst Mechanismen in den gestrichelten Rechtecken. Diese erlauben den logischen Entwurf von E-Service-Modellen und deren Erweiterung zu Service-orient. Koordinations- und Kompositionsmodellen auf grafischer Basis von UML. Des Weiteren sind hier Transformationsmechanismen enthalten, die zur Umwandlung von UML-basierten E-Service-Modellen in XPDL-basierte textuelle Re-

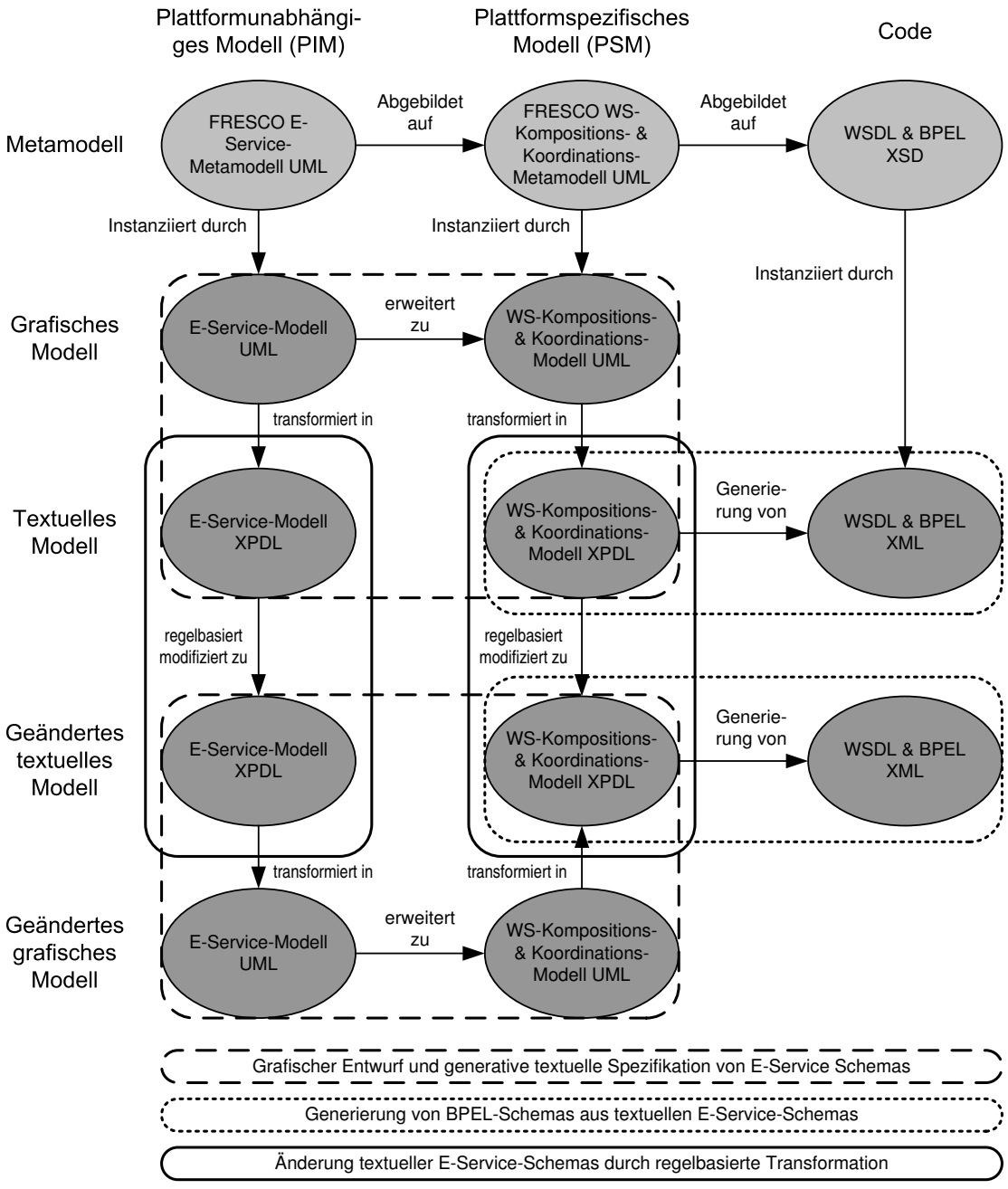

Abbildung 5.3.: Modellgetriebene Methodik der FRESCO E-Service-Entwicklung

präsentationen dienen. Während der interaktive Entwurf von Modellen in grafischer Form mit UML erfolgt, bietet XPDL ein alternatives textuelles Format, das sich auf Grund der WfMC-Standardisierung zum organisationsübergreifenden Austausch eignet. Zudem eignet sich das Format auch gut zur automatisierten Verarbeitung. XPDL-basierte Modelle dienen daher als Eingabe der anderen Werkzeuge. Hier ist zunächst ein BPEL-Generator zu nennen, der durch gepunktete Rechtecke markiert ist. Dieser enthält Mechanismen zum automatisierten physikalischen Entwurf, die XPDL-basierte E-Service-Modelle in ausführbare WS-Kompositionsschemata auf Basis von BPEL transformieren. Das verbleibende Werkzeug, das durch ein solides Rechteck gekennzeichnet ist, dient der Änderung XPDL-basierter PIM und PSM. Hierzu kommen Transformationsregeln zur Anwendung, die eine Änderung in systematischer automatisierter Weise erlauben. Derart geänderte PSM können direkt als Eingabe des BPEL-Generators dienen. Geänderte PIM werden erst in das grafische Format überführt, um eine interaktive Erweiterung zu PSM vornehmen zu können. Die drei Werkzeuge der beschriebenen modellgetriebenen Entwicklungsmethodik werden im Folgenden noch einmal einzeln und in ausführlicher Form beschrieben.

Grafischer Entwurf und generative Spezifikation von VDLP In der Vorkontaktphase einer Dienstleistung besteht eine der wesentlichen Aufgaben der E-Service-Technologie in der exakten Erfassung der Interaktionen bei der Dienstleistungsproduktion. Die formale Beschreibung des virt. Dienstleistungsprozesses macht hier Interaktionsmuster zwischen Produktionseinheiten von Kunden und Providern explizit und dient als Koordinationsprotokoll zur Publikation und wechselseitigen Abstimmung. *Entwurf und Spezifikation von E-Services* sind daher als wichtige Entwicklungsmethoden im FSMV berücksichtigt. Gemäß diesem Vorgehensmodell erfolgt der vorläufige Entwurf von E-Services zunächst auf eine interaktive Weise nach Vorgabe der Anforderungen eines Geschäftsprozessmodells. Anfangs wird dabei ein plattformunabhäng. Modell erstellt, das den E-Service in einer abstrakten Form beschreibt. Für den kollaborativen Entwurf muss das PIM zu plattformspezif. Modellen erweitert werden, die zusätzliche Eigenschaften eines Service-orient. Koordinationsmodells erfassen. Danach erfolgt eine strukturelle Verifikation der Modelle. Schließlich wird das Modell in das Austauschformat transformiert und publiziert, damit es von potenziellen Produktionseinheiten eingesehen werden kann, die ihrerseits PIM und PSM entwerfen und dabei die Vorgaben des Koordinationsmodells einfließen lassen. In FRESCO wird die Semantik von PIM und PSM durch das FRESCO E-Service-Metamodell beschrieben. Hier werden die Konzepte und Strukturen auf den einzelnen Abstraktionsebenen hierarchisch definiert und basieren letztendlich auf einem fundamentalen Workflow-Metamodell. Entwurf und Spezifikation von E-Services basieren auf Modellierungssprachen, die für die Konzepte des Metamodells eine syntaktische Repräsentation zur Verfügung stellen. Für diese Repräsentation ist in FRESCO ein dualer Ansatz vorgesehen, der für Entwurf und Spezifikation

zwei alternative Modellformate berücksichtigt. Ein grafisches Format auf Basis von UML dient zum interaktiven Entwurf von Modellen. Ein textuelles Format auf Basis von XPDL erlaubt den interoperablen Austausch von Modellen als Protokollspezifikationen zwischen Produktionseinheiten und dient als Eingabe für automatisierte Transformationen durch andere Mechanismen des E-Service-Managements. Diese kombinierte Methode wird in FRESCO durch ein E-Service-Entwurfswerkzeug unterstützt [AK05]. Dessen eigener Entwurf beruht auf einer Spezialisierung genereller UML-basierter Modellierung. Hierbei wird der generische Sprachumfang von UML um die spezifischen Konzepte des FRESCO E-Service-Metamodells ergänzt. Dadurch stehen UML-Elemente mit FSM-spezifischer Semantik zum Entwurf von E-Service-Modellen, WS-Koordinations- und -Kompositionsmodellen sowie Workflow-Modellen zur Verfügung. Auf Basis des formal spezifizierten UML-Metamodells können diese UML-Modelle automatisch verarbeitet werden. Durch die zusätzliche formale Auszeichnung der FSM-spezifischen Semantik kann auch diese bei der Verarbeitung berücksichtigt werden. Dies wird zur Transformation der UML-basierten Modelle in XPDL-basierte textuelle Repräsentationen genutzt. Umgekehrt ist es so auch möglich, die Elemente einer XPDL-basierten Modellrepräsentation in Modellstrukturen entsprechender UML-Elemente zu überführen. Im Folgenden wird zunächst die Spezialisierung der UML-Semantik durch ein FRESCO UML-Profil vorgestellt. Danach wird die Verwendung des Profils zum E-Service-Entwurf besprochen. Schließlich erfolgt eine Diskussion der Transformationsmechanismen.

Die Erweiterung von UML ist durch die Sprache selbst u. a. auf Basis von *UML-Profilen* vorgesehen [OMG03, S. 2-73]. Diese basieren auf den drei Konzepten der *Stereotypen*, *Tagged Values* und *Constraints*. Durch Stereotypen können UML-Elemente in Bezug auf eine spezifische Semantik ausgezeichnet werden. Tagged Values erlauben die Annotation derart spezialisierter UML-Elemente durch Name-Wert-Paare. Constraints sind Ausdrücke auf Basis der Object Constraint Language (OCL), mit denen Bedingungen für die Verwendung von Stereotypen bei der Modellierung formuliert werden können. UML-Profile fassen verschiedene Stereotypen, Tagged Values und Constraints zusammen, die gemeinsam die Aspekte von Modellen einer spezifischen Domäne festlegen. Die Erweiterung von UML in Bezug auf den Entwurf von E-Services erfolgt durch ein solches UML-Profil. Dieses *FRESCO UML-Profil* führt im Wesentlichen die im FSM beschriebenen Konzepte als Stereotypen ein. In Entsprechung der Abstraktionsebenen im FSM ist das Profil in vier hierarchische Bereiche mit Stereotypen für E-Service-, Koordinations-, Kompositions- und Workflow-Modelle gestaffelt. Die fundamentale Ebene wird durch ein Profil zur Modellierung WfMC-konformer Workflows gebildet. Dieses Workflow-Profil folgt grundsätzlich dem im FSM integrierten WfMC-Workflow-Metamodell. Zur Spezifikation von PSM ist dies jedoch nicht detailliert genug. Stattdessen wird daher ein verfeinertes Metamodell auf Basis des XML-Schemas von XPDL verwendet. Hierin sind neben den im FSM verwendeten Basiselementen eine Reihe zusätzlicher Konzepte zu deren Konkretisierung

PIM		PSM				UML Elemente	Anmerkung
	E-Service-Model Stereotypen	Service Coordination Model Stereotypen	Service Composition Model Stereotypen	WfMC-Modell Stereotypen			
Rollen	eService Interaction Coordinator	Conversation Orchestrator	Web Service Aggregator	Participant		Actor, Class	ParticipantType=SYSTEM
	eService Client	Conversation Party	Web Service Provider	Participant		Actor, Class	ParticipantType=ROLE
	eService Provider	Conversation Party	Web Service Provider	Participant		Actor, Class	
	eService Capability Provider	Conversation Party	Web Service Provider	Participant		Actor, Class	
	–	–	fresco.CorrelationSet	ExtendedAttribute		Class	Liste von Typdeklarationen
Endpunkte	eService Interaction Service	Conversation Orchestrator Receptor	Web Service Composition Operation	Application		Package, Class	Immer <<Web Service Description Reference>>
	eService Demand Contact	Conversation Party Listener	Web Service Component Operation	Application		Package, Class	
	eService Asset Access	Conversation Party Listener	Web Service Component Operation	Application		Package, Class	
	eService Capability Service	Conversation Party Listener	Web Service Component Operation	Application		Package, Class	
	–	→	Web Service Description Reference	ExternalReference		Class	
	–	→	WSDL Document Reference	–		Attribute	Location= WSDL URI
	–	→	WSDL Port/Operation Reference	–		Attribute	xref= port/operation name
Interaktionen	eService Client Feedback	Party Statement	Web Service Composition Interaction	Activity		Package, Class, Activity	Type:Implementation, Implementation-Type:Tool, <<Tool>> referenziert Endpoint, <<fresco.communication>> mit Value=in
	eService Resource Directive	Party Statement	Web Service Composition Interaction	Activity		Package, Class, Activity	
	eService Sub-Capability Outgoing Response	Party Statement	Web Service Composition Interaction	Activity		Package, Class, Activity	
	eService Sub-Capability Outgoing Request	Party Statement	Web Service Composition Interaction	Activity		Package, Class, Activity	
	eService Flow Handover	Party Statement	Web Service Composition Interaction	Activity		Package, Class, Activity	
	eService Client Statement	Orchestration Directive	Web Service Component Interaction	Activity		Package, Class, Activity	Type:Implementation, Implementation-Type:Tool, <<Tool>> referenziert Endpoint, <<fresco.communication>> mit Value=out
	eService Provider Statement	Orchestration Directive	Web Service Component Interaction	Activity		Package, Class, Activity	
	eService Sub-Capability Incoming Response	Orchestration Directive	Web Service Component Interaction	Activity		Package, Class, Activity	
	eService Sub-Capability Incoming Request	Orchestration Directive	Web Service Component Interaction	Activity		Package, Class, Activity	
	eService Flow Takeover	Orchestration Directive	Web Service Component Interaction	Activity		Package, Class, Activity	
	–	–	Provider Correlation Map	–		Attribute	Performer=Provider, Parameter Index
	–	–	Aggregator Correlation Map	–		Attribute	Performer=Aggregator, Parameter Index
	→	→	fresco.communication	ExtendedAttribute		Class	Value=in/out
E-Service Struktur	eService Capability Interaction Sub Process	–	–	Activity		Package, Class, Activity	Type:Implementation, Implementation-Type:SubFlow
	eService Capability Sub Process	–	–	Activity		Package, Class, Activity	
	eService interaction Context	→	Web Service Interaction Context	Package		Package	
	eService Capability Interaction Pattern	Web Service Composition Protocol	Web Service Composition	Package		Package	GraphConformace=LOOP_BLOCKED
	eService Initial Capability	Web Service Composition Protocol	Web Service Composition	Package		Package	GraphConformace=LOOP_BLOCKED
	eService Shell	–	–	Package		Package	Id=E-Service Name, GraphConformace=LOOP_BLOCKED
	eService Capability Interaction Flow	Orchestrated Multi-Party Conversation	Web Service Orchestration Process	WorkflowProcess		Package, Activity Diagram	
	eService Capability	–	–	WorkflowProcess		Package, Activity Diagram	degeneriert; nur <<Capab. Inter. Sub Process>>
	eService Capabilities	–	–	WorkflowProcess		Package, Activity Diagram	Name:"Capabilities", degeneriert; nur <<eService Capability Sub Process>>
	–	→	fresco.targetNamespace	ExtendedAttribute		Class	Value=namespace URI
Daten	eService Data Format	→	Web Service Message Type Declaration	TypeDeclaration		Package, Class	Immer <<XML Schema Type Reference>>
	→	→	Orchestration Data	DataField		Package, Class	<<DataType>> ref. immer <<Web Service Message Type Declaration>>
	→	→	→	DataType		Dependency	
	–	→	XML Schema Type Reference	ExternalReference		Class	Location= XSD URI, xref= Typname
	–	–	Web Service Coordination Metadata	DataField		Package, Class	Konstante Prozessvariable
Prozessdefinition	*opt*	*opt*	*opt*	ExtendedAttribute		Class	Nur informal
	→	→	→	ActivitySet		Package, Activity Diagram	
	→	→	→	Activity		Package, Class	Nur Endpoints oder Routes
	–	→	→	Tool		Package, Class	Nur i.B.a. Endpoints
	–	–	–	SubFlow		Package, Class	Nur i.F.v. <<eService Capab. Inter. Sub Process>>, <<eService Capab. Sub Process>>
	→	→	→	ActualParameter		Class	Immer <<Orchestration Data>>
	–	–	–	Formal Parameter		Package, Class	Spezifikation erfolgt in WSDL
	→	→	→	Transition		Package, Class, Transition	
	→	→	→	Condition		Package	keine Exceptions
	→	→	→	Xpression		Class	Ausdruck in XPath 1.0 Notation
	→	→	→	Join		Class	
	→	→	→	Split		Package	
	→	→	→	TransitionRestriction		Package	
	→	→	→	TransitionRef		Class	
Metadaten	*opt*	*opt*	*opt*	RedefinableHeader		Class	
	–	–	→	Script		Class	Nur XPath 1.0
	opt	*opt*	*opt*	PackageHeader		Class	
	opt	*opt*	*opt*	ProcessHeader		Class	
	opt	*opt*	*opt*	Deadline		Class	
	opt	*opt*	*opt*	SimulationInformation		Class	
Spezifikationsstruktur	→	→	→	WorkflowProcesses		Package	
	→	→	→	ExtendedAttributes		Package	
	→	→	→	ExternalPackages		Package	
	→	→	→	ExternalPackage		Package	
	→	→	→	Type Declarations		Package	
	→	→	→	DataFields		Package	
	→	→	→	Participants		Package	
	→	→	→	Applications		Package	
	–	–	–	Formal Parameters		Package	Spezifikation erfolgt in WSDL
	→	→	→	ActivitySets		Package	
	→	→	→	Activities		Package	
	→	→	→	ActualParameters		Package	
	→	→	→	Transitions		Package	
	→	→	→	TransitionRestrictions		Package	
	→	→	→	TransitionRefs		Package	

Tabelle 5.1.: Stereotypen im FRESCO UML-Profil

und Strukturierung enthalten, die detailliert genug sind, um Workflow-Schemata zu spezifizieren. Auf dieser Basis enthält das UML-Profil Stereotypen, Tagged Values und Constraints, die den Entwurf von Workflows als UML-Modelle reglementieren. Tabelle 5.1 gibt einen Überblick des Profils. Hierin sind alle Stereotypen mit ihren hierarchischen Beziehungen aufgelistet. Dies beginnt links auf der höchsten Abstraktionsebene der E-Service-Modelle und geht nach rechts bis zur Ebene der WfMC-Modelle, deren Stereotypen mit ihren UML-Basiselementen gezeigt sind.

Der Entwurf des Profils zielt zunächst darauf ab, die Komplexität XPDL-orientierter Workflow-Modelle für den interaktiven Entwurf handhabbar zu machen. Hierzu werden die zahlreichen Konzepte, die nur zur Strukturierung von Modellen dienen, im Wesentlichen durch UML-Pakete abgebildet. Deren vereinzelte Attribute werden meist durch Tagged Values erfasst. Tragende Konzepte, die auch im FSM erfasst sind, erhalten neben einem UML-Paket meist eine zusätzliche UML-Klasse zur Erfassung komplexerer Inhalte. Daneben werden zum Teil Constraints definiert, um Kardinalitäten und Wertebereiche einzugrenzen. Die resultierende Pakethierarchie strukturiert die zahlreichen Elemente komplexer Modelle in homogene Teilbereiche. Sie ist allerdings weniger zur Visualisierung in Diagrammen als in Tabellen oder Bäumen von Entwurfswerkzeugen geeignet. Während die Modellorganisation auf diese Weise in den Hintergrund tritt, werden einige Schlüsselaspekte hervorgehoben, die für den Entwurf von besonderer Bedeutung sind. Dazu gehört zuallererst die Prozessstruktur. Diese wird durch UML-Aktivitätsdiagramme visualisiert, in denen Aktivitäten und Transitionen dargestellt werden. Ein Ausschnitt aus dem Profil mit Stereotypen im Zusammenhang von Workflow-Prozessen und deren Beziehung zu den Elementen von Aktivitätsdiagrammen ist in Abb. 5.4 dargestellt.[4]

Auf die Stereotypen für WfMC-Modelle bauen die verfeinerten Stereotypen der abstrakteren Ebenen auf. Die unmittelbare Hierarchie der Konzepte ist in den linken Spalten von Tabelle 5.1 dargestellt. Die Modellstrukturen der einzelnen Ebenen sind im FSM spezifiziert. Hier finden sich alle Stereotypen als Konzepte der einzelnen Metamodelle wieder und sind dort miteinander in Beziehung gesetzt. Die entsprechenden UML-Assoziationen im FSM sind hierbei wiederum mit Stereotypen versehen, die ihre Realisierung auf Basis von Konzepten des WfMC-Metamodells vorgeben, d. h. z. B. dass in Bezug auf Abb. 4.23 (S. 273) zur Modellierung der Zuordnung von ≪`E-Service Capability Interaction Flow`≫ als Realisierung von ≪`E-Service Capability Interaction Pattern`≫ Elemente des Stereotyps ≪`WorkflowProzesses`≫ verwendet werden. Weitere Hinweise zum Entwurf sind in der Tabelle angegeben.

Der Entwurfsvorgang von E-Services auf Basis des FRESCO UML-Profils startet auf PIM-Ebene mit der Erstellung eines E-Service-Modells. Hierbei kommen primär die Stereotypen des E-Service-Metamodells zum Einsatz. Ein wichtiger Aspekt ist hier zunächst die Modellierung des E-Service Core mit den darin enthaltenen Assets und

[4]Die Darstellung des Profils basiert auf einer UML-basierten Notation. Deren genaue Beschreibung sowie das komplette UML-Profil der Workflow-Ebene finden sich in [AK05, S. 41 ff.].

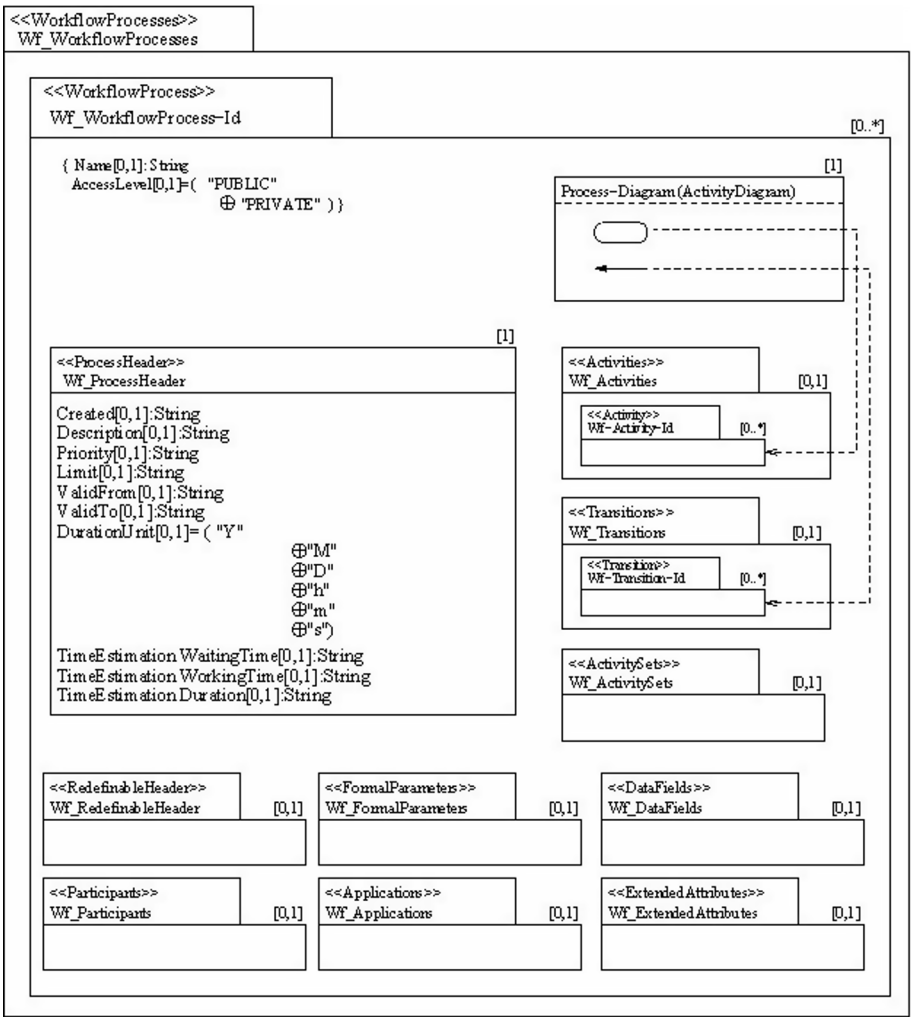

Abbildung 5.4.: Workflow-Prozess-Stereotyp im FRESCO UML-Profil [AK05, S. 53]

Asset-spezifischen Capabilities. Weiterhin ist die geplante Aktivität von Kunden zu antizipieren. All diese Aspekte werden durch Rollen, Endpunkte und Datenformate in einem ≪`E-Service Interaction Context`≫-Paket beschrieben. Hierauf basiert dann die Modellierung der E-Service-Shell durch ein zentrales ≪`eService Shell`≫-Paket. Darin werden im Wesentlichen die Capabilities des E-Service erfasst. Diese werden zunächst als ≪`eService Capabilities`≫ aufgelistet und durch ≪`eService Capability`≫-Elemente abstrakt beschrieben. Die detaillierte Modellierung als ≪`eService Capability Interaction Pattern`≫ und vor allem ≪`eService Capability Interaction Flow`≫ kann im Verlauf eines kollaborativen Entwurfsvorgangs schrittweise erfolgen. Capabilities sollten anfangs nur in Bezug auf Metadaten und Interaktionen, aber ohne komplizierte Prozessstruktur beschrieben werden. Letztere wird in Kollaboration der beteiligten Akteure schrittweise verfeinert.

Vor allem in den detaillierteren Modellen kommen auch Stereotypen anderer Metamodellebenen zum Einsatz. Z. B. werden zur Beschreibung eines ≪eService Capability Interaction Flow≫ auch ≪Transition≫ und andere Elemente einer Prozessbeschreibung verwendet. Jedoch sollte der Detaillierungsgrad von E-Service-Modellen nicht bis auf plattformspezifische Details herunterreichen. Solche Details bilden den Inhalt von Koordinations- und Kompositionsmodellen. Deren Entwurf erfolgt als Verfeinerung eines bestehenden E-Service-Modells. Hierbei werden die dort lediglich abstrakt vorgegebenen Interaktionsprozesse mit ihren Rollen, Endpunkten, Datenformaten und Prozessstrukturen in Hinblick auf eine Service-orient. Ausführungsplattform konkretisiert. Dazu gehört z. B. die Deklaration von Datentypen und entsprechende Prozessvariablen mit XML-Schema sowie die Formulierung von Transitionsbedingungen durch Ausdrücke auf Basis der XML Path Language (XPath). Insbesondere erfolgt an dieser Stelle auch eine Verknüpfung von Endpunkten mit WSDL-Schnittstellen. Zudem sind verschiedene andere Konkretisierungen, z. B. in Hinblick auf die Korrelation und Prozessmetadaten, durchzuführen, die im Zusammenhang mit der Ausführungsplattform stehen und auf die bei der Implementierung noch eingegangen wird. Welche Konzepte auf welcher Modellebene zu spezifizieren sind, kann im Einzelnen aus Tabelle 5.1 entnommen werden. Stereotypen werden dort auf höheren Modellebenen entweder durch spezifischere Konzepte verfeinert, werden übernommen (→), sind optional (*opt*), oder entfallen (–).

Auf beiden Ebenen von PIM und PSM kann eine Transformation in Bezug auf das Format der Modelltransformation erfolgen. Hierbei wird im einen Fall das UML-Modell in die textuelle Repräsentation eines XPDL-Dokuments überführt. Im anderen Fall wird das XPDL-Dokument in die grafische Repräsentation eines UML-Modells gewandelt. Auf PIM-Ebene erfolgt die Transformation von UML nach XPDL im Wesentlichen als Austauschformat im organisationsübergreifenden Kontext des Dienstleistungsnetzwerks. Ein weiterer Aspekt ist die Verarbeitung des Modells durch einen regelbasierten Transformationsmechanismus. In jedem Fall ist danach eine Rückwandlung in das grafische Format notwendig, in dem dann auch eine interaktive Verfeinerung von der PIM- zur PSM-Ebene erfolgen kann. Auf der PSM-Ebene dient die Transformation von UML nach XPDL dann in erster Linie als Zwischenformat, das durch Generatoren in ein Ausführungsformat weiterverarbeitet werden kann. In FRESCO ist ein solcher Generator in exemplarischer Weise für das BPEL-Format entwickelt worden. Vor der Eingabe in den Generator kann auch hier eine Verarbeitung durch regelbasierte Transformation erfolgen. Hierdurch können im Einzelfall ad hoc-Änderungen am E-Service vorgenommen werden.

Die Transformation zwischen UML- und XPDL-Format basiert auf dem Workflow-Metamodell. Dieses bildet in direkter Weise die Spezifikationen im XML-Schema von XPDL auf UML Stereotypen ab, d. h. dass eindeutige Beziehungen zwischen typisierten UML-Elementen und Abschnitten in XPDL-Dokumenten bestehen. Auf dieser Basis können einfache Regeln spezifiziert werden, die für jedes Workflow-Konzept jeweils

die Beziehung zwischen einem UML Stereotyp und einem XPDL-Element beschreiben. Solche Regeln beinhalten in erster Linie die Überführung der zu den Konzepten gehörigen Informationen in den jeweiligen Repräsentationen. Die Transformation von UML-Modellen in XPDL-Dokumenten kann durch diese Abbildung unmittelbar erfolgen. Die umgekehrte Transformation von XPDL-Dokumenten in UML-Modelle erfordert dagegen eine weitergehende Auswertung auf inhaltlicher Ebene. In dieser Richtung kann zunächst ganz entsprechend dem umgekehrten Fall eine Generierung von UML-Elementen mit den Stereotypen des Workflow-Metamodells erfolgen. Die Semantik des Workflow-Modells auf den höheren Metamodellebenen geht jedoch aus der XPDL-Repräsentation nicht unmittelbar hervor. Die Konzepte höherer Ebenen von PIM und PSM müssen daher aus den Mustern der XPDL-Spezifikation abgeleitet werden. Initial lässt sich bei FSM-konformer Spezifikation die ≪`eService Shell`≫ eindeutig erkennen. Hieraus lassen sich dann wiederum der ≪`eService Interaction Context`≫ und die einzelnen ≪`eService Capability`≫-Pakete ableiten. Entsprechend können aus den Strukturen eines XPDL-Elements alle höherwertigen Konzepte abgeleitet werden. Hierfür liegt dann aber noch kein grafisches Layout vor. Ein solches Layout kann zwar automatisch erstellt werden, führt aber oft nicht zu der gewünschten Übersichtlichkeit. An dieser Stelle ist dann ggf. ein manueller Eingriff notwendig.[5]

Abbildung von VDLP-Spezifikationen auf BPEL-Schemata Der vorläufige Entwurf eines FRESCO E-Service beinhaltet den grafischen Entwurf von PIM, deren Erweiterung zu PSM und Transformation in das textuelle XPDL-Format. Im FSMV folgt an dieser Stelle die VDPN-Implementierung (Stufe drei), in der potenzielle Kunden und Provider in Bezug auf den vorläufigen E-Service-Entwurf identifiziert werden. Am Anfang der darauffolgenden Stufe vier) können sich dann als Resultat von Verhandlungen zwischen den Teilnehmern Änderungen am vorläufigen E-Service-Entwurf ergeben. Diese werden ebenfalls mit den Methoden des grafischen Entwurfs und zusätzlich mit Methoden der E-Service-Änderung umgesetzt. Ist dies geschehen, besteht der nächste Schritt der E-Service-Entwicklungsmethodik im physikalischen E-Service-Entwurf. Hierfür wird in FRESCO eine automatisierte Methode zur Generierung von WS-Kompositionsschemata auf Basis des PSM-Entwurfs bereitgestellt. Konkret wurde dazu eine Transformation XPDL-basierten E-Service-PSM in BPEL-Schemata entworfen [BP04, S. 72 ff.].

Der physikalische Entwurf von FRESCO-E-Services spezifiziert ein Service-orient. Anwendungssystem nach dem Architekturmuster des FRESCO-SOM in Bezug auf standardisierte Service-orient. Basistechniken.[6] Dies beinhaltet zum einen die Spezifikation von Asset, Demand und Capability Services als WS-Schnittstellen. Zum

[5] Wenn für ein Modell schon eine frühere Version im UML-Format vorliegt, kann unter Umständen auf deren Layout zurückgegriffen werden. Ein derartiges „Round-Trip-Engineering“ erfordert allerdings komplexe Mechanismen, die im Folgenden nicht weiter verfolgt werden sollen.

[6] Vgl. Sektion 4.3.2.1, S. 258.

anderen wird die Implementierung von Capability Services als WS-Kompositionsschemata spezifiziert. Die Spezifikationen bilden die Grundlage für Konstruktion und Deployment der Services auf Basis einer E-Service-Ausführungsplattform.[7] Der physikalische E-Service-Entwurf ist eine Konkretisierung des logischen E-Service-Entwurfs, der durch ein E-Service-PSM vorgegeben wird. Das FRESCO E-Service-PSM gibt die Kommunikationsendpunkte von Demands, Assets und Capabilities sowie die Interaktionsmuster von Capabilities als konkretes globales Koordinationsprotokoll in einem neutralen Austauschformat vor. Dabei sind die Interaktionsmuster von Capabilities nach Vorgabe des FRESCO E-Service-Metamodell in XPDL-Format spezifiziert. Die Kommunikationsendpunkte einzelner Demands, Assets und Capabilities werden hierin durch XPDL-Rollen zusammengefasst und in Form von WSDL-Schnittstellen beschrieben. Auf dieser Basis erfolgt der physikalische E-Service-Entwurf in Form von BPEL-Schemata und entsprechend erweiterter WSDL-Schnittstellen. Dies bildet die Grundlage für Konstruktion und Deployment von Capability Services auf Basis BPEL-basierter WS-Middleware, die als COTS-Komponente verfügbar ist.

Die Methode des physikalischen Entwurfs basiert in FRESCO auf automatischer Generierung. Hierbei wird der logische Entwurf des VDLP in Form XPDL-basierter E-Service-PSM durch einen Generatormechanismus auf eine Menge von BPEL-Schemata abgebildet. Nach Vorgabe des FSM besteht jede `eService Capability` aus einer Menge separater `eService Capability Interaction Flows`. Diese werden im E-Service-PSM zu `Web Service Orchestration Processes` konkretisiert und als einzelne `WorkflowProcesses` im XPDL-Format repräsentiert. Im Zuge des physikalischen E-Service-Entwurfs wird für jeden `eService Capability Interaction Flow` einer `eService Capability` ein BPEL-Schema generiert und dem zugehörigen `eService Coordinator` zugeordnet. Der hierzu entworfene Generatormechanismus bildet den jeweils korrespondierenden `Web Service Orchestration Process` mitsamt den WSDL-Spezifikationen der hierin bereitgestellten und angesprochenen WS-Operationen in ein BPEL-Schema und dessen erweiterte WSDL-Schnittstelle ab. In formaler Darstellung entspricht dies einer Abbildung:

$$XPDL^{flow} \times WSDL^{flow} \times (WSDL^{resource})^n \rightarrow BPEL^{flow} \times WSDL^{flow}$$

Hierbei werden Schnittstellen, die durch den `Web Service Orchestration Process` angesprochen werden, als $WSDL^{resource}$ bezeichnet. Die Schnittstelle, die der Orchestrierungsprozess selber bereitstellt, wird als $WSDL^{flow}$ gekennzeichnet. Konkret beinhaltet der Abbildungsprozess eines `Web Service Orchestration Process` i mit n einbezogenen Ressourcen eine Transformation:

$$(x_i^{flow}, w_i^{flow}, w_1^{resource}, \ldots, w_n^{resource}) \longmapsto (b_i^{flow}, w_i^{flow'})$$

Hierbei besteht der Definitionsbereich aus Tupeln mit:

[7] Eine ausführliche Beschreibung der E-Service-Ausführungsplattform in FRESCO folgt in Sektion 5.2.2.

- `Web Service Orchestration Process` in XPDL-Repräsentation x_i^{flow}
- WSDL-Spezifikation w_i^{flow} der `Web Service Composition Operations` von x_i^{flow}
- WSDL-Spezifikationen $w_1^{resource}, \ldots, w_n^{resource}$ der `Web Service Component Operations` von n verschiedenen `Web Service Providern`

Der Bildbereich besteht aus Tupeln mit:

- BPEL-Prozess b_i^{flow} entsprechend x_i^{flow}
- WSDL-Spezifikation $w_i^{flow'}$ des durch b_i^{flow} implementierten Web Service

Im Folgenden werden die grundlegenden Prinzipien dieser Transformation beschrieben. Dazu erfolgt eine Gegenüberstellung verschiedener Konzepte und Strukturen von XPDL und BPEL sowie eine Ableitung entsprechender Abbildungen.[8] Die Betrachtung behandelt dabei nacheinander Aspekte des Kontroll- und Datenflusses sowie des Managements von Ressourcen und Instanzen.

In einem FRESCO `Web Service Orchestration Process` wird der Ablauf von Web Service-Interaktionen als Kontrollfluss eines XPDL `WorkflowProcess` spezifiziert. Für die angestrebte Transformation ist daher zunächst eine Abbildung der grundsätzlichen Kontrollstrukturen von XPDL und BPEL zu leisten.

Ein fundamentaler Aspekt dieser Fragestellung ist die Überbrückung der unterschiedlichen Grundkonzepte für Kontrollstrukturen in XPDL und BPEL. In XPDL basiert die Beschreibung des Kontrollflusses auf einer Graphstruktur, die sich an die Semantik von Flussdiagrammen anlehnt. Hierbei bilden Aktivitäten die Ecken und Transitionen die Kanten eines gerichteten Graphen. Der Kontrollfluss verläuft zwischen Aktivitäten entlang der Transitionen. Er wird durch Bedingungen für die Transition von Kanten und Restriktionen für die Verzweigung (Split) und Zusammenführung (Join) in Knoten gesteuert. In BPEL basiert die Beschreibung des Kontrollflusses hingegen primär auf einer Blockstruktur, die durch Prozessalgebra motiviert ist. Wie schon in Sektion 3.4.2.3 beschrieben, basiert diese Struktur auf einer Auswahl strukturierter Aktivitäten, die sich an den Operatoren des π-Calculus orientieren. Sie bestimmen die Ausführungsreihenfolge der darin enthaltenen Aktivitäten und lassen sich beliebig verschachteln. Konkret beinhaltet das die sequenzielle, konditionale, wiederholte, ereignisbedingte und nebenläufige Ausführung von atomaren und strukturierten Aktivitäten.

Im Vergleich bilden die in Blockstruktur darstellbaren Abläufe eine Untermenge derjenigen, die sich durch die Graphstruktur darstellen lassen. Konkret lassen sich asymmetrisch verschachtelte Splits und Joins nicht in Blockstruktur darstellen. Gleiches gilt für Schleifen zwischen Zweigen unterschiedlicher Split/Join Paare. Bei der Abbildung von Graphstrukturen auf Blockstrukturen muss also eine Beschränkung

[8] Die Betrachtung erfolgt auf Basis von XPDL Version 1.0 und BPEL Version 1.1.

der Graphstrukturen erfolgen. In XPDL können in diesem Sinne Konformitätsklassen definiert werden, die die Prozessstruktur reglementieren. Die Klasse `FULL_BLOCKED` schreibt dabei Graphstrukturen mit symmetrischen Split/Join Paaren und ohne Zyklen vor, die sich unmittelbar als Blockstruktur darstellen lassen.

Auf eine derart strikte Beschränkung soll in der vorliegenden Arbeit jedoch verzichtet werden. Die Unterschiede der resultierenden Prozessstrukturen zu den auf organisatorischer Ebene verbreiteten Flussdiagrammen würden eine zu große Einschränkungen für den E-Service-Entwurf bedeuten. Ein Ausweg bietet sich auf Basis von BPEL `Flow`-Aktivitäten an, die durch Definition von `Links` die Spezifikation einer Graphstruktur erlauben. Ein Link verbindet hier immer genau zwei Aktivitäten und kann mit einer Bedingung verknüpft werden. Aktivitäten können mehrere aus- oder eingehende Links besitzen, die mit Bedingungen zur Zusammenführung verknüpft werden können. Auf diese Weise kann die Graphstruktur eines XPDL-Prozesses auf die Graphstruktur innerhalb einer zentralen BPEL `Flow`-Aktivität abgebildet werden. Dabei kann die Einschränkung der XPDL-Konformitätsklasse auf `LOOP_BLOCKED` reduziert werden. Hierdurch werden lediglich Zyklen ausgeschlossen, die in `Flow`-Aktivitäten nicht vorkommen dürfen.

Zur Herleitung einer Abbildung von XPDL auf BPEL-Graphstrukturen muss als Nächstes die Semantik der unterschiedlichen Kontrollstrukturen verglichen werden. Dies erfolgt auf Basis fundamentaler Kontrollmuster, die in beiden Sprachen unterstützt werden. Hierzu wird eine Auswahl der von van der Aalst vorgeschlagenen *Workflow Patterns* verwendet [AHK+03] und deren Anwendung auf XPDL [Aal03b] und BPEL [WAD+03] zugrunde gelegt. Vergleich und Abbildung beschränken sich bei dieser Betrachtung explizit auf den Aspekt des Kontrollflusses. In diesem Sinne zeigt Tabelle 5.2 die Realisierung der Muster durch XPDL und BPEL auf Basis einer Pseudonotation für Knoten a_x und Kanten t_y der jeweiligen Graphstrukturen. Auf XPDL-Seite repräsentiert ein Knoten $[X]a[Y]$ eine `Activity` a mit `TransitionRestriction` X für Join und Y für Split ($X, Y = AND \vee XOR$). Kanten $t(a_x \rightarrow a_y)[A]$ repräsentieren eine `Transition` t von a_x nach a_y mit `Condition` A. Auf BPEL-Seite repräsentiert ein Knoten $(t_1, t_2 \rightarrow)[A]a(\rightarrow t_3[B], t_4[C])$ eine `activity` a, die `target` von `link` t_1 und t_2 mit `joinCondition` A sowie `source` von `link` t_3 und t_4 mit `transitionCondition` B und C ist. Eine Kante t repräsentiert einen `link`. Die untersuchten Muster werden mit WP1-WP13 bezeichnet und beziehen sich auf die entsprechenden Workflow Patterns [Aal03b].

Das fundamentale Muster der *sequence* (WP1) wird in beiden Sprachen mit leicht unterschiedlicher Syntax unterstützt. Zur Abbildung muss nur die Referenzierung zwischen Knoten und Kanten des Graphen von den Transitionen (XPDL `from`, `to`) in die Aktivitäten (BPEL `source`, `target`) verschoben werden. Die folgenden Muster der Tabelle betreffen die Verzweigung des Kontrollflusses. Hierzu bietet XPDL `AND`- und `XOR`-Splits mit Transitionsbedingungen. Bedingungen können logische Ausdrücke sein oder durch `OTHERWISE` einen Standardpfad ausweisen. Sie werden im Falle von `XOR` in Folge abgearbeitet, bis eine zutrifft. XPDL bietet hier nur die Möglichkeit, `source links` mit

Muster	**XPDL**	**BPEL**
WP1: Sequence	a_1; $t_1(a_1 \rightarrow a_2)$; a_2	$a_1(\rightarrow t_1)$; t_1; $(t_1 \rightarrow)a_2$;
WP2: Parallel Split	$a_1[AND]$; $t_1(a_1 \rightarrow a_2)$; $t_2(a_1 \rightarrow a_3)$; $t_3(a_1 \rightarrow a_4)$	$a_1(\rightarrow t_1, t_2, t_3)$; t_1; t_2; t_3
WP4: Exclusive Choice (a)	$a_1[XOR]$; $t1(a_1 \rightarrow a_2)[A]$; $t2(a_1 \rightarrow a_3)[B]$; $t3(a_1 \rightarrow a_4)[C]$	$a_1(\rightarrow t_1[A], t_2[\overline{A}B], t_3[\overline{(A \vee B)}C])$; t_1; t_2; t_3
WP4: Exclusive Choice (b)	$a_1[AND]$; $t_1(a_1 \rightarrow a_2)[C1]$; $t_2(a_1 \rightarrow a_3)[C2]$; $t_3(a_1 \rightarrow a_4)[C3]$ (disjunkt)	$a_1(\rightarrow t_1[C1], t_2[C2], t_3[C3])$; t_1; t_2; t_3 (disjunkt)
WP6: Multi Choice	$a_1[AND]$; $t_1(a_1 \rightarrow a_2)[A]$; $t_2(a_1 \rightarrow a_3)[B]$; $t_3(a_1 \rightarrow a_4)[C]$	$a_1(\rightarrow t_1[A], t_2[B], t_3[C])$; t_1; t_2; t_3
XPDL: Otherwise	$[\ldots]a_1[\ldots]$; $t_1(a_1 \rightarrow a_2)[A]$; $t_2(a_1 \rightarrow a_3)[B]$; $t_3(a_1 \rightarrow a_4)[C]$; $t_4(a_1 \rightarrow a_5)[otherwise]$	$a_1(\underline{\rightarrow} t_1[\ldots], t_2[\ldots], t_3[\ldots], t_4[\overline{(A \vee B \vee C)}])$; t_1; t_2; t_3; t_4
WP3: Synchronisation	$t_1(a_2 \rightarrow a_1)$; $t_2(a_3 \rightarrow a_1)$; $t_3(a_4 \rightarrow a_1)$; $[AND]a_1$	$(t_1, t_2, t_3 \rightarrow)[L(t_1) \vee L(t_2) \vee L(t_3)]a_1$; t_1; t_2; t_3 (implizit)
WP5: Simple Merge	$t_1(a_2 \rightarrow a_1)$; $t_2(a_3 \rightarrow a_1)$; $t_3(a_4 \rightarrow a_1)$; $[XOR]a_1$	$(t_1, t_2, t_3 \rightarrow)[L(t_1) \vee L(t_2) \vee L(t_3)]a_1$; t_1; t_2; t_3 (implizit)
WP7: Synch. Merge	$t_1(a_2 \rightarrow a_1)$; $\langle t_2(a_3 \rightarrow a_1) \rangle$; $t_3(a_4 \rightarrow a_1)$; $[AND]a_1$	$(t_1, t_2, t_3 \rightarrow)[L(t_1) \vee L(t_2) \vee L(t_3)]a_1$; t_1; $\langle t_2 \rangle$; t_3 (implizit)
WP8,9: Multi-Merge, Discriminator	$t_1(a_2 \rightarrow a_1)$; $\langle t_2(a_3 \rightarrow a_1) \rangle$; $t_3(a_4 \rightarrow a_1)$; $[XOR]a_1$	–

Tabelle 5.2.: Vergleich und Abbildung von Kontrollflussmustern in XPDL und BPEL

Transitionsbedingungen zu versehen, die entweder als logische Ausdrücke vorliegen oder implizit als wahr evaluiert werden. Mit diesen Mitteln werden nun die Muster realisiert. Als Erstes wird der *parallel split* (WP2) betrachtet. Dieses Verhalten wird in XPDL durch `AND` Split erreicht und bildet in BPEL das Standardverhalten bei multiplen ausgehenden `links`. Im *exclusive choice*-Muster (WP4) wird genau ein ausgehender Pfad gewählt. Dies lässt sich in XPDL (a) durch einen `XOR`-Split erreichen, wobei der erste Pfad mit zutreffender `Condition` eingeschlagen wird. Alternativ lässt sich (b) ein `AND`-Split mit disjunkten `Conditions` verwenden. Ein Fall (a) entsprechendes Verhalten kann in BPEL durch Umformung der `Condition`-Ausdrücke erreicht werden. Für die alternative Form (b) müssen nur die `Conditions` übernommen werden. Das *multi-choice*-Muster (WP6) betrifft die Auswahl beliebiger ausgehender Pfade. In XPDL wird dazu ein `AND`-Split mit Transitionsbedingungen verwendet. In BPEL entspricht dies multiplen `source` links mit entsprechenden Transitionsbedingungen. Für die Semantik von XPDL `OTHERWISE` ist kein unmittelbar passendes Muster vorhanden. Eine Abbildung nach BPEL lässt sich aber durch Synthese einer Transitionsbedingung für den Standardpfad erreichen, der dann eingeschlagen wird, wenn keine andere Bedingung zutrifft.

Die verbleibenden Muster der Tabelle betreffen die Zusammenführung des Kontrollflusses. Hierzu bietet XPDL `AND` und `XOR` Joins. Deren Semantik in Bezug auf die Ausführung eingehender Pfade geht aus der Spezifikation nicht eindeutig hervor und bedarf der Interpretation im Einzelfall. BPEL bietet hier die Möglichkeit, Bedingungen zu formulieren, die mittels `linkStatus` von `target links` die Komplettierung bzw. den „Tod“ eingehender Pfade berücksichtigen können. Das erste Join-Muster ist *synchroni-*

sation (WP3) als Gegenstück von WP2. Dies wird in XPDL durch ein `AND`-Join erreicht. In BPEL kann hierzu eine `JoinCondition` als Konjunktion des `linkStatus` aller `target links` formuliert werden. Das *simple merge*-Muster (WP5) führt alternative Äste zusammen, von denen jeweils nur einer aktiv ist. In XPDL lässt sich dies durch einen `XOR`-Join ausdrücken. In BPEL entspricht dies einer Disjunktion des `linkStatus` von `target links`, was auch ohne `JoinCondition` implizit angenommen wird. Im *synch. merge* Muster (WP7) werden komplettierte Pfade synchronisiert und „tote“ Pfade ignoriert, die im Prozessverlauf nicht aktiviert wurden. Durch die unklaren Ausführungen der XPDL-Spezifikation kann dies ebenfalls als Verhalten eines `AND`-Join interpretiert werden. In BPEL wird dies wiederum durch implizite Disjunktion der Zustände von `target links` erreicht, denn eine Auswertung dieser `JoinCondition` findet erst statt, wenn alle Zustände vorliegen und dadurch also eine Synchronisation stattgefunden hat. Hierdurch entsteht ein Konflikt der Abbildung von XPDL-`AND`-Joins nach BPEL. Dies wird dadurch aufgelöst, das WP3 in BPEL ebenfalls als implizite Disjunktion der (`link-`)Zustände interpretiert wird. Dies ist auf Grund der synchronisierenden Wirkung der Auswertung von `JoinConditions` gerechtfertigt. Weitere Muster sind *multi-merge* (WP8) und *discriminator* (WP9). In WP8 wird die Join-Aktivität für verschiedene eingehende Zweige in beliebiger Reihenfolge ausgeführt. In WP9 wird sie hingegen nur bei Komplettierung des ersten Zweiges ausgeführt und alle weiteren ignoriert. Die XPDL-Spezifikation lässt eine entsprechende Semantik von `XOR`-Joins offen und in BPEL wird diese überhaupt nicht unterstützt. Entsprechend werden keine Rückschlüsse auf die Abbildung gezogen. Neben den bislang diskutierten Mustern können in XPDL noch einige weitere realisiert werden. Dazu gehört das *arbitrary cycles*-Muster (WP10), das aber schon per XPDL-Konformitätsklasse ausgeschlossen wurde. Das *implicit termination*-Muster (WP11) wird dahingehend berücksichtigt, dass weder in XPDL noch in BPEL explizite Endaktivitäten notwendig sind und der Prozess in beiden Fällen nach Abarbeitung aller Aktivitäten terminiert. Das *Multiple activity instances without synchronisation*-Muster (WP12) erschien für den E-Service-Entwurf nicht relevant und wurde daher nicht explizit untersucht. Gleiches gilt für das *Multiple activity instances with a-priori design time knowledge*-Muster (WP13). Dieses Muster beruht jedoch auf der Kopie von Aktivitäten in parallele Äste eines *parallel split* mit anschließender *synchronisation* und kann daher abgebildet werden.

Als Ergebnis dieser Gegenüberstellung lässt sich festhalten, dass für eine Menge fundamentaler Muster Abbildungen zwischen XPDL- und BPEL-Kontrollstrukturen identifiziert wurden. Hierdurch lassen sich die wesentlichen Fälle von Sequenzen, Splits und Joins eines XPDL-basierten `Web Service Orchestration Process` in Graphstrukturen auf Basis einer BPEL `flow activity` überführen. Im Falle von Splits wurden für alle XPDL-Kontrollstrukturen außer den nicht betrachteten `Exception Conditions` Abbildungen formuliert. Für Joins wird eine Synchronisation-, Simple Merge- oder Synch. Merge-Semantik angenommen. Die Abbildung kann dann Joins komplett ignorieren, da das entsprechende Verhalten in BPEL jeweils implizit gegeben ist.

XPDL Activity					BPEL activities
Type	Implementation-type	Extended-Attribute	Performer	Actual-Parameters	
Route	–	–	–	–	empty
BlockActivity	–	–	–	–	–
Implementation	No	–	–	–	empty
Implementation (Web Service Composition Interaction)	Tool	fresco.communication=in	Aggregator Correlation Map	Datafields ggf. Ausdrücke	sequence: receive, assign, reply
Implementation (Web Service Component Interaction)	Tool	fresco.communication=out	Provider Correlation Map	Datafields ggf. Ausdrücke	sequence: assign, invoke, assign
Implementation	Subflow	–	–	–	–

Tabelle 5.3.: Abbildung von XPDL-Aktivitäten nach BPEL

Für einen Transformationsalgorithmus von XPDL- auf BPEL-basierte Graphstrukturen fehlt nun lediglich noch eine Abbildung von Aktivitäten, die in Tabelle 5.3 dargestellt ist. Hierbei werden die XPDL-Aktivitäten `Route` und `Implementation/No`, die nicht mit der Ausführung einer Aufgabe bzw. deren Kontrolle verbunden sind, als BPEL `empty`-Aktivitäten abgebildet. XPDL-Aktivitäten des Typs `Implementation/Tool` steuern die Ausführung einer Aufgabe durch Aufruf eines Anwendungssystems. In einem `Web Service Orchestration Process` wird hierdurch die Interaktion zwischen Web Service-Aggregator und Providern modelliert. Diese Interaktionen werden auf eine Kombination von BPEL `assign`-Aktivitäten zur Datenmanipulation und Interaktionsdirektiven `invoke` sowie `receive` und `reply` abgebildet. Die genaue Abbildung wird bei der Untersuchung des Datenflussaspekts weiter unten behandelt. Schließlich werden die XPDL-Aktivitäten `BlockActivity` und `Implementation/Subflow` für den Entwurf von E-Services ausgeschlossen. Da die Semantik von `BlockActivities` aus der XPDL-Spezifikation nicht eindeutig hervorgeht, sollen durch deren Ausschluss Mehrdeutigkeiten im Entwurf vermieden werden. Durch den Verzicht auf potenziell asynchrone Unterprozesse soll hingegen die dadurch entstehende Komplexität im Kontrollfluss vermieden werden.

Auf Basis der Abbildungsvorschriften für Kontrollstrukturen und Aktivitäten lässt sich nun der Transformationsalgorithmus für den Prozessgraphen formulieren. Das Prinzip besteht darin, für den XPDL-basierten `Web Service Orchestration Process` zunächst eine BPEL `flow activity` zu generieren. Hierin werden dann für alle XPDL `Activities` und `Transitions` BPEL `activities` und `links` erstellt. Schließlich erfolgt die Anpassung von Transitionsbedingungen der BPEL `source links`. Der entsprechende Algorithmus wird wie folgt definiert:

1. Erstelle eine BPEL `Flow`-Aktivität F als Prozess-Container.

2. Erstelle für jede XPDL `Transition` mit `Id`=t ein BPEL `link` Element mit `name`=t und füge dieses zu F hinzu.

3. Erstelle für jede XPDL `Activity` mit `Id`=a eine BPEL-Aktivität entsprechend der Abbildungsforschriften für Aktivitäten mit `name`=a und füge diese zu F hinzu.

4. Füge für jede XPDL-Transition t^X von Aktivität a^X (d. h. mit `from`=a^X.`Id`) zu der damit korrespondierenden BPEL-Aktivität a^B ein `source` Sub-Element mit `linkName`=t^X.`name` hinzu. Falls t^X eine Transitionsbedingung besitzt, füge dem `source` Element eine entsprechende Transitionsbedingung hinzu.

5. Füge für jede XPDL-Transition t^X zu Aktivität a^X (d. h. mit `to`=a^X.`Id`) zu der korrespondierenden BPEL-Aktivität a^B ein `target` Sub-Element mit `linkName`=t^X.`name` hinzu.

6. Für jede XPDL-Aktivität a^X: falls a^X mehrere abgehende Transitionen $t_1^X, \ldots, t_n^X$ besitzt:
   - Falls t_i^X eine Default-Transition (`OTHERWISE`) ist und alle $t_j^X (j \neq i)$ eine Transitionsbedingung c_j^X besitzen: dann ändere die korrespondierende BPEL-Transitionsbedingung zu $\widetilde{c_i^B} := \overline{\bigvee_{j \neq i} c_j^B}$.
   - Falls a^X eine Transitionsrestriktion in Form einer `XOR`-Verzweigung enthält und mehrere Transitionen (ohne die Default-Transition) mit Transitionsbedingungen $c_1^X, \ldots, c_n^X$ abgehen: ersetze die korrespondierenden BPEL-Transitionsbedingungen $c_j^B (j = 2 \ldots n)$ mit $\widetilde{c_j^B} := \overline{\left(\bigvee_{k=1}^{j-1} c_k^B\right)} \wedge c_j^B$

Neben dem Kontrollfluss ist der Datenfluss ein zweiter wichtiger Aspekt im XPDL-basierten `Web Service Orchestration Process`. Die wesentliche Aufgabe des Orchestrierungsprozesses besteht in der Steuerung von Interaktionen mit Web Service-Providern. Bei den einzelnen Interaktionen werden jeweils Nachrichten zwischen Aggregator und Provider ausgetauscht. Empfangene Nachrichten von Providern werden im Orchestrierungsprozess als `Orchestration Data` zwischengespeichert. Im weiteren Verlauf dient `Orchestration Data` dann zum einen zur Steuerung des Kontrollflusses durch Auswertung von Transitionsbedingungen. Zum anderen gehen daraus die Inhalte von Nachrichten hervor, die in späteren Interaktionen wiederum an Provider versandt werden. In diesem Sinne kann zwischen Intra- und Inter-Prozess-Datenfluss unterschieden werden. Der Intra-Prozess-Datenfluss beinhaltet die Definition von `Web Service Message Type Declarations` und `Orchestration Data` sowie deren Zugriff und Manipulation. Der Inter-Prozess-Datenfluss beinhaltet die Definition von WS-Nachrichten und deren Umwandlung von bzw. in `Orchestration Data`. Beide Aspekte des Datenflusses sind von XPDL- auf BPEL-basierte Orchestrierungsprozesse abzubilden und werden im Folgenden diskutiert.

Der Intra-Prozess-Datenfluss ergibt sich in XPDL durch typisierte Prozessvariablen, die im `Web Service Orchestration Process` als `Orchestration Data` und allg. als `Datafields` gegeben sind. Zur Typisierung von `Datafields` bietet XPDL verschiedene

Möglichkeiten. Zum einen existiert ein proprietäres Typsystem mit grundlegenden und komplexen Datentypen sowie der Möglichkeit zur Typdeklaration. Zum anderen können externe Typsysteme referenziert werden. In einem `Web Service Orchestration Process` werden `Orchestration Data`-Elemente auf Basis von `Web Service Message Type Declarations` typisiert. Diese spezialisieren XPDL-Typdeklarationen derart, dass hier ausschließlich Referenzen auf externe XML-Schema-Typen erlaubt sind. Zu Beginn von Prozessen können `Datafields` mit Standardwerten initialisiert werden. Im weiteren Verlauf sind sie mit den aktuellen Parametern von Aktivitäten verknüpft. Ansonsten ist keine Datenmanipulation vorgesehen. XPDL `Datafields` korrespondieren zu BPEL-Variablen, die wahlweise auf XML-Schema oder WSDL-Nachrichtentypen basieren. BPEL-Variablen können durch `assign`-Aktivitäten auf verschiedene Weise manipuliert oder kopiert werden. Zur Abbildung des Intra-Prozess-Datenflusses wird jedes XPDL `Datafield` einer BPEL-Variable zugeordnet. Für diese wird der in XPDL durch die `Web Service Message Type Declaration` referenzierte XML-Schema-Typ übernommen. Die Initialisierung von `Datafields` mit Standardwerten wird in BPEL durch `assign` Aktivitäten erreicht, die per `sequence` hinter alle potenziellen Startaktivitäten gestellt werden.

Für die Deklaration von Transitionsbedingungen wird in einem `Web Service Orchestration Process` die Verwendung der XML Path Language (XPath) 1.0 angenommen [CD99]. Auf Basis derselben Sprache können auch Ausdrücke formuliert werden, die die Verwendung von `Orchestration Data` in aktuellen ausgehenden Parametern von Aktivitäten beschreiben. Transitionsbedingungen sind XPath-Ausdrücke mit boolschem Wert. Parameter können durch Ausdrücke beschrieben werden, die Zeichenketten, Zahlen oder boolsche Werte ergeben. In allen Ausdrücken können Referenzen auf `Datafields`, logische Standardfunktionen und Literale mittels verschiedener hierarchischer Operatoren verknüpft werden. Da BPEL ebenfalls XPath 1.0 unterstützt, können die Ausdrücke bei der Abbildung im Wesentlichen übernommen werden. Lediglich die Referenzierung von Variablen ist anzupassen.

Der Inter-Prozess-Datenfluss wird in einem `Web Service Orchestration Process` auf Basis spezialisierter `Activities`, `Applications` und `Participants` definiert.[9] Das Empfangen einer Nachricht erfolgt durch `Web Service Composition Interactions`. Dies sind Aktivitäten mit Implementierung durch genau ein `Tool`, das eine `Web Service Composition Operation` referenziert. Außerdem wird immer ein `Web Service Aggregator` als `Performer` der Aktivität referenziert. `Web Service Composition Operations` referenzieren eine WSDL-Spezifikation sowie eine `request-response` oder `one-way Operation` innerhalb eines `Ports`. Sie definieren dadurch die ein- und ggf. ausgehenden WSDL-Nachrichten der Interaktion. Die aus- und ggf. eingehenden aktuellen Parameter der XPDL-Aktivität werden den `Parts` der ein- und ggf. ausgehenden Nachrichten der WSDL-Operation in lexikalischer Reihenfolge zugeordnet und müssen in Anzahl und Typ mit diesen übereinstimmen. Der `Web Service Aggregator` ist ein künstlicher XPDL

[9] Vergleiche hierzu das FRESCO Service Composition Metamodel Abb. 4.21.

Participant (Typ SYSTEM), der zur Definition eines fresco.correlationSet dient. Dieses legt Datentypen fest, mit denen die Zuordnung eingehender Nachrichten zu Prozessinstanzen erfolgen soll. Der Performer-Ausdruck der XPDL-Aktivität beinhaltet eine Aggregator Correlation Map, die neben dem Namen des Aggregators die Positionen von Parametern in der Parameterliste festlegt, die für das Instanz-Routing verwendet werden sollen. Die derart referenzierten Parameter müssen in Reihenfolge, Anzahl und Typ dem fresco.correlationSet entsprechen. Das Versenden einer Nachricht erfolgt in weitgehender Entsprechung zum Empfang durch Web Service Component Interactions. Dies sind XPDL-Aktivitäten mit Implementierung durch genau ein Tool, das eine Web Service Component Operation referenziert. Es wird immer ein Web Service Provider als Performer der Aktivität referenziert. Die Referenzierung von Schnittstelle, Operation und Nachrichten einer WSDL-Spezifikation durch eine Web Service Component Operation und der Zusammenhang der WSDL-Nachrichten mit den Parametern der Aktivität entspricht dem obigen Fall. Der Web Service Provider definiert hier jedoch eine Rolle, die zusammenhängende Operationen bündelt und deren Zuordnung zu einem Teilnehmer zwecks Adressierung von Aufrufen erlaubt. Ferner wird für jeden Web Service Provider ebenfalls ein fresco.correlationSet definiert. Dieses legt Datentypen fest, mit denen ein Instanz-Routing auf Seite des Providers erfolgt. Der Performer-Ausdruck der XPDL-Aktivität beinhaltet entsprechend eine Performer Correlation Map zur Bestimmung der Parameter zum Instanz-Routing.

In BPEL erfolgt der Empfang von Nachrichten durch zusammenhängende receive- und ggf. reply-Aktivitäten. Das Versenden von Nachrichten erfolgt durch eine invoke-Aktivität. Ein- und ausgehende Nachrichten können hier jedoch nicht unmittelbar mit den Prozessvariablen assoziiert werden. Stattdessen muss eine Variable mit dem entsprechenden WSDL-Nachrichtentyp verwendet werden. In diese werden vor dem Versand die Inhalte verschiedener Variablen durch assign-Aktivitäten kopiert. Nach dem Erhalt erfolgt entsprechend das Kopieren von Inhalten in verschiedene Variablen. Für die Abbildung von XPDL nach BPEL bedeutet dies zum einen, dass für alle ein- und ausgehenden Nachrichten BPEL-Variablen der entsprechenden WSDL-Nachrichtentypen vorzusehen sind. Zum anderen werden die XPDL-Interaktionen auf Mengen strukturierter BPEL-Aktivitäten abgebildet. Konkret wird eine Web Service Composition Interaction auf eine BPEL sequence-Aktivität mit bis zu fünf Schritten wie folgt abgebildet:

1. receive – ggf. Initialisierung des BPEL-Prozesses; Empfang der eingehenden Nachricht und Kopie in Nachrichtenvariable

2. assign – Bei neuem Prozess: Zuweisung von konstanten Werten zu Variablen entsprechend den Standardwerten von XPDL Datafields

3. assign – Kopie von Teilen der eingehenden Nachricht in Variablen entsprechend der Parameterliste der XPDL-Aktivität

4. `assign` – Bei `request-reply` Operation: Erstellung der Inhalte von Teilen der ausgehenden Nachricht durch Kopie aus Variablen und Auswertung von Ausdrücken entsprechend der Parameterliste der XPDL-Aktivität

5. `reply` – Bei `request-reply` Operation: Versand der ausgehenden Nachricht

Entsprechend wird eine `Web Service Component Interaction` auf eine BPEL `sequence`-Aktivität mit bis zu drei Schritten wie folgt abgebildet:

1. `assign` – Erstellung der Inhalte von Teilen der ausgehenden Nachricht durch Kopie aus Variablen und Auswertung von Ausdrücken entsprechend der Parameterliste der XPDL-Aktivität

2. `invoke` – Versand der ausgehenden Nachricht; bei `request-reply` Operation: Empfang der eingehenden Nachricht und Kopie in Nachrichtenvariable

3. `assign` – Bei `request-reply` Operation: Kopie von Teilen der eingehenden Nachricht in Variablen entsprechend der Parameterliste der XPDL-Aktivität

Zur Komplettierung der Abbildung fehlen nun noch die Konstrukte zur Bestimmung der Rollen von Teilnehmern und der Korrelation von Prozessinstanzen. In einem XPDL-basierten `Web Service Orchestration Process` definieren `Web Service Provider` Rollen, die im `Performer` Ausdruck von `Web Service Component Interactions` referenziert werden. Alle Interaktionen einer Rolle betreffen den gleichen Teilnehmer und werden allesamt an dessen WSDL Port $w_j^{resource}$ adressiert. In BPEL wird dieses Verhalten durch die Spezifikation von `partnerLinks` erreicht, die in den verschiedenen Interaktionen referenziert werden können. `partnerLinks` sind durch `partnerLinkTypes` mit WSDL Ports Verbunden, die von den Partnern zur Verfügung gestellt werden müssen. Bei der Abbildung wird für jeden XPDL `Web Service Provider`, der im `Performer` Ausdruck einer `Web Service Component Interaction` vorkommt, ein BPEL `PartnerLink` in x_i^{flow} erstellt. Dieser wird über einen `partnerLinkType` in w_i^{flow} mit seinem WSDL Port in $w_j^{resource}$ verknüpft. Die entsprechenden `invoke` oder `reply activities` werden mit dem `PartnerLink` verknüpft. Die Bindung konkreter Teilnehmer an eine Rolle r_j im Orchestrierungsprozess und die Adressierung von Instanzen des Web Service $w_j^{resource}$ auf Basis von Korrelationsinformationen lassen sich in BPEL 1.1 nicht zufriedenstellend abbilden. Hierfür wird in FRESCO stattdessen eine separate Komponente der Ausführungsumgebung verwendet.[10]

`Web Service Aggregators` definieren in dem auf XPDL basierenden `Web Service Orchestration Process` für alle `Web Service Composition Interactions` durch ein `fresco.correlationSet` diejenigen Anteile eingehender Nachrichten, die mit der Prozessinstanz korrelieren. In BPEL wird dieses Verhalten durch entsprechende `CorrelationSets`

[10]Die Beschreibung des in FRESCO zu diesem Zweck entworfenen Aggregationsmechanismus erfolgt in Sektion 5.2.2.2.

erreicht. Bei der Abbildung wird für jeden XPDL `Web Service Aggregator` ein BPEL `CorrelationSet` in x_i^{flow} erstellt und nach diesem benannt. Alle BPEL `receive activities` werden dann mit ihren zugehörigen `CorrelationSets` verknüpft und als Startpunkte neuer Instanzen definiert. Schließlich werden die BPEL `CorrelationSets` entsprechend der Definitionen korrespondierender `fresco.correlationSets` und `fresco.correlationMaps` durch BPEL `properties` und `propertyAliases` in w_i^{flow} mit Teilen der eingehenden WSDL-Nachrichten verknüpft.

Mit diesen Schritten wird die Abbildung vervollständigt. Auf dieser Basis kann ein Generator konstruiert werden, der `Web Service Orchestration Processes` automatisch in einen BPEL-Prozess transformiert. So ein BPEL-Prozess kann dann im weiteren Verlauf in einer BPEL Engine der FRESCO-Ausführungsplattform ablaufen.

Regelbasierte Änderung von VDLP-Spezifikationen Das FRESCO E-Service-Management-Vorgehensmodell sieht im Wesentlichen an zwei Stellen die Änderung von E-Service-Entwürfen vor. In der Reihenfolge von Phasen des Entwicklungslebenszyklus ist dies zunächst am Anfang des detaillierten E-Service-Entwurfs (Stufe 4) der Fall. An dieser Stelle finden ggf. Verhandlungen zwischen konkreten VDPN-Teilnehmern statt, deren Ergebnisse eine Änderungen am vorläufigen E-Service-Entwurf notwendig machen. Da die Potenzialerstellung in der Vorkontaktphase der Dienstleistung bereits abgeschlossen ist und die Dienstleistungserbringung unmittelbar bevorsteht, können solche Änderungen nicht fundamentaler Natur sein. Änderungen betreffen daher weniger Dienstleistungsinhalte, sondern mehr die Art der Erbringung. So können z. B. Dienstleistungsmerkmale die schon in Form von Assets vorliegen, hinzu- oder abgewählt werden, indem deren Interaktionsmuster in den globalen VDLP ein- oder ausgegliedert werden. Diese Änderungen müssen am logischen E-Service-Entwurf vorgenommen werden, bevor der physikalische Entwurf beginnen kann.

Die zweite Situation, in der eine Änderung zum Tragen kommt, ist die Stilllegungsphase (Stufe 5). Hier geht ein konkreter Fall der Dienstleistungserbringung in einer spezifischen Konstellation des VDPN zu Ende. Dabei wird die E-Service-Instanz zu dessen Steuerung verworfen. Dies gilt jedoch nicht für den zugrunde liegenden E-Service-Entwurf. Dieser Entwurf geht als bleibendes Wissen in das Dienstleistungsnetzwerk ein. Zum einen ist dies nötig, um Audit-Informationen, die ggf. während der Dienstleistungserbringung gesammelt wurden, auch später noch interpretieren zu können. Zum anderen gehen vorhandene E-Service-Entwürfe in neue Entwicklungslebenszyklen für nachfolgende VDPN über. Für den vorläufigen logischen E-Service-Entwurf dieses Lebenszyklus besteht eine wesentliche Entwurfsmethode in der Änderung der vorhandenen Entwürfe. Da E-Services in diesem Fall in Bezug auf ihre funktionalen Anteile erhalten bleiben sollen, beziehen sich solche Änderungen in der Regel auf nicht-funktionale Aspekte. Hinweise können hier z. B. durch Evaluation der vorangegangenen E-Service-Ausführung gewonnen werden. Die hierbei gemessenen qualitativen und quantitativen Größen sind dabei mit entsprechenden Kennzahlen

abzugleichen. Bei Abweichungen müssen u. a. Änderungsstrategien für den Entwurf in Betracht gezogen werden. Hier können manche nicht-funktionalen Eigenschaften durch Änderungen der Ablaufstruktur im Interaktionsmuster beeinflusst werden. Z. B. kann durch Einführung von Empfangsbestätigungen die Zuverlässigkeit der Dienstleistungserbringung erhöht werden.

Die Änderung von E-Service-Modellen ist eine nicht-triviale Aufgabe, die Effizienz, Präzision und Konsistenz der Methodik fordert. Die Modelle sind potenziell sehr komplex. Der ganzheitliche virt. Dienstleistungsprozess setzt sich in der Regel aus einer Vielzahl von `eService Capabilities` zusammen, die wiederum aus Mengen von `eService Capability Interaction Flows` bestehen. Hinzu kommt, dass die einzelnen Teilprozesse nicht nur innerhalb einer Capability, sondern auch dazwischen eng zusammenhängen. Änderungen müssen sich daher in der Regel auf mehr als einen Punkt der VDLP-Struktur beziehen. Dies beinhaltet dann Änderungen an verschiedenen Punkten der internen Ablaufstruktur von multiplen Capabilities. Hierbei ist die exakte Identifikation der relevanten Capabilities und der darin zu ändernden Punkte entscheidend. Änderungen an anderer Stelle würden zu Fehlern im Entwurf führen. Alles in allem sind an eine Methode zur Änderung von E-Services die folgenden Anforderungen zu stellen:

- *Präzision* – Bei der Durchführung einer Änderung muss ein exakter Zugriff erfolgen, was die Strukturen und deren Kontext betrifft. Der Zugriff auf Strukturen darf keine Änderungen an anderen als den geplanten Strukturen bewirken.

- *Konsistenz* – Die Änderung eines konsistenten Systems darf nicht zu einem inkonsistenten System führen. Änderungen an zusammenhängenden oder abhängigen Strukturen müssen sich auf all diese Strukturen beziehen und ggf. auswirken, so dass deren Beziehung bzw. Abhängigkeit nach der Änderung erhalten bleibt.

- *Effizienz* – Bei einer komplexen Änderungsstrategie muss sich eine große Zahl von Änderungen an einer großen Zahl von Strukturen durchführen lassen. Die Durchführung muss sich in einer absehbaren und praktikablen Zeitspanne vollziehen.

Als Basis für eine solche Änderungsmethode wurde in FRESCO das Konzept der *regelbasierten E-Service-Transformation* [ZP04] entwickelt. Hierbei wird die Änderungslogik explizit als Menge von Regeln erfasst. Diese Regeln können dann in systematischer Weise auf alle Interaktionsprozesse des virt. Dienstleistungsprozesses angewendet werden. Die Anwendung der Regeln auf die Prozesse wird durch ein Software-Werkzeug unterstützt. Das Konzept für Transformationsregeln basiert auf der Suche und direkten Ersetzung von Mustern in Prozessen. Beim E-Service-Entwurf werden die zu ändernden Punkte innerhalb der Prozessstruktur als Muster spezifiziert. Zudem werden Änderungen von Prozessen festgelegt, die die Muster

enthalten. Das klare Prinzip der Änderungsmethode fördert deren Anwendbarkeit und Präzision, da sie einfach zu erlernen und anzuwenden ist. Zudem fördert die explizite Formalisierung der notwendigen Änderungen am E-Service-Entwurf die Konsistenz des Vorgehens. Ein vollständiger Überblick der Änderungslogik ermöglicht eine systematische Sicht auf die Auswirkungen dieser Änderungen für den E-Service-Entwurf. Beziehungen zwischen verschiedenen Arten der Änderung lassen sich besser erkennen und Konflikte sowie Synergien werden sichtbar. Eine kohärente Sicht auf den Änderungsplan vereinfacht eine Validierung von Änderungen an den Prozessen in Bezug auf die Anforderungen an den E-Service-Entwurf.

In FRESCO liegen E-Service-Entwürfe als PIM und PSM auf Basis des FSM vor. Im Sinne der FSM-Spezifikation basieren E-Service-PSM und Service-orient. -PIM auf Konzepten eines Workflow-Metamodells und lassen sich mit den weiter oben beschriebenen Methoden als Workflow-Beschreibung im XPDL-Format darstellen. Die Umsetzung des Konzepts regelbasierter Transformation beruht daher auf der Anwendung einer regelbasierten Sprache zur Workflow-Transformation [BP04, S. 54 ff.].

Diese Sprache erlaubt die Beschreibung genereller Änderungsstrategien für Workflow-Konfigurationen, die für eine Menge konkreter Workflow-Prozesse global umgesetzt werden können. Hierbei kann der Effekt der Änderung individuell auf einen geeigneten Kontext eingeschränkt werden (z. B. auf Capabilities bestimmter Provider). In diesem Kontext werden Prozesse mit einer unerwünschten Konfiguration identifiziert. Änderungen werden durch Transformationsanweisungen von der Basiskonfiguration in eine erwünschte Zielkonfiguration beschrieben. Diese Transformationen werden auf alle identifizierten Prozesse in systematischer und automatisierter Weise angewendet. Für transformierte Prozesse wird ein fundamentaler Konsistenztest durchgeführt. Dieser erkennt offensichtliche Verletzungen der Prozessstruktur und macht die Transformation im Fehlerfalle rückgängig.

Änderungsstrategien werden als Mengen individueller Transformationsregeln strukturiert. Jede Regel enthält zunächst einen Teil zur Prozessselektion (*Prozessselektor*), um den Anwendungskontext einzuschränken. Ein Suchteil (*Suchmuster*) dient zur Identifikation der Basiskonfiguration in den Prozessen des Anwendungskontextes. Ein Ersetzungsteil (*Ersetzungsvorschrift*) beinhaltet die gewünschten Transformationen.

Transformationsregeln basieren fundamental auf Workflow-Mustern, die die Identifikation spezifischer Strukturen in einer Prozessmenge erlauben. Generell beinhaltet ein Muster eine Aufzählung und Benennung von Workflow-Elementen unter Berücksichtigung einer Menge von Konditionen zu deren Identifikation (siehe Listing 5.1).

```
<element type>{<cardinality>} <binding name> :
  <Boolean expression>,
    ...
  <Boolean expression>;
```

Listing 5.1: Schematisches Workflow-Muster

Jede Kondition beschreibt Anforderungen für eine Untermenge von Workflow-Elementen. Workflow-Elemente, die alle Anforderungen erfüllen, werden für die weitere Verarbeitung berücksichtigt. Konkret wird ein Muster durch eine *Suchspezifikation* realisiert, die Workflow-Elemente an spezifische Namen bindet und Anforderungen in Bezug auf Typ, Kardinalität und Zustand beinhaltet:

- *Typanforderungen* beschränken Elemente auf einen bestimmten Typ. Typen beziehen sich dabei auf Klassen von Workflow-Elementen nach den Vorgaben des XML-Schemas von XPDL [WfM02].

- *Kardinalitätsanforderungen* schränken gültige Treffer auf eine spezifische Anzahl von Elementen ein. Im Standardfall wird genau ein Element gesucht. Andere Bereiche können mithilfe verschiedener Modifikatoren spezifiziert werden. Der *Optionalität* (`?`) Modifikator führt zu nicht-obligatorischen Vergleichsbedingungen. Ein *Asterisk* (`*`) bewirkt multiple und ggf. auch keine Bindungen. Dies resultiert in einer Menge aller passender Elemente außer denen, die schon durch vorangegangene Vergleichsbedingungen des Musters gefunden wurden. *Existenz* (`+`) ähnelt dem Asterisk, wobei hier die Elementmenge nicht leer sein darf. *Ausschluss* (`-`) dient als Schutz, da die Suche abbricht, falls so ein Element gefunden wird. Der Ausschluss kann auch für eine Konjunktion von Unterbedingungen definiert werden.

- *Zustandsbedingungen* werden als boolsche Ausdrücke in Bezug auf Attribute und Assoziationen von Elementen spezifiziert. Ein Ausdruck bildet eine *Kondition* der Suchspezifikation. Die Liste aller Konditionen wird als *Klausel* bezeichnet und als Konjunktion der Konditionen interpretiert. In einzelnen Ausdrücken können alle Elemente referenziert werden, die im Muster gebunden wurden. Deren Attribute können gelesen, ausgewertet und durch verschiedene Operatoren kombiniert werden.

In Transformationsregeln bildet der initiale *Prozessselektor* eine spezifische Suchspezifikation, die als *Selektionsmuster* bezeichnet wird. Das Selektionsmuster kombiniert stets die Typanforderung `Process` mit der Kardinalitätsanforderung *Existenz* (`+`). In der Zustandsanforderung können Konditionen formuliert werden, die die zu berücksichtigenden Prozesse beschreiben.

Zur Veranschaulichung soll hier ein Beispiel dienen, das sich auf das in Kapitel 6 realisierte Logistikszenario bezieht. Hierbei soll die bedingte Integration einer zusätzlichen Bestätigung in den VDLP bewirkt werden. In dem Beispiel werden alle Prozesse vorselektiert, deren Attribut `name` die Zeichenkette „Handover“ beinhaltet (siehe Listing 5.2). Außerdem müssen sie Teil eines `Package` sein, das ab dem Jahr 2004 erstellt wurde. Das Beispiel demonstriert dabei die vielfältigen Möglichkeiten zur Formulierung von Ausdrücken. In diesem Fall ist zur Spezifikation u. a. eine *Typzuweisung* notwendig.

```
RULE "transport: conditionally add insurer acknowledgement"
  FOR_ALL_PROCESSES p
    p.name contains "Handover",
    date(p.package.packageHeader.created)
    > date("01/01/2004 00:00");

MATCH_ALL
  Activity start_compensation:
    start_compensation.name contains "start_compensation_process";
  EXCLUSION "excludeIfInformed"
    Activity not_contact_insurer:
      not_contact_insurer.name == "notify_insurer_replaced" or
      not_contact_insurer.name == "inform_insurer_replaced";
    Transition notify_compensate:
      notify_compensate.from == not_contact_insurer,
      notify_compensate.to == start_compensation;;

REPLACE_WITH
  start_compensation;
  Activity acknowledge_compensation:
    id = "acknowledge_compensation",
    name = "Acknowledge start of compensation",
    implementation = No();
  Transition compensation_notify:
    id = "compensation to acknowledge",
    from = start_compensation,
    to = acknowledge_compensation;
```

Listing 5.2: Transformationsregel mit Selektionsmuster (oben) Suchmuster (mittig) und Erstzungsteil (unten)

Der Suchteil einer Transformationsregel wird durch ein Suchmuster spezifiziert, das beliebige Suchspezifikationen beinhaltet. Das Suchmuster schreibt eine spezifische Konfiguration beliebiger Workflow-Elemente vor, deren Existenz bzw. Nichtexistenz eine Bedingung für die weitere Bearbeitung darstellt. Zusätzlich deklariert das Muster die Bindung von Elementen, die aus Übereinstimmungen mit den Suchspezifikationen resultieren, für die weitere Verarbeitung. Eine Elementmenge, die für jede Suchspezifikation des Musters eine Bindung übereinstimmender Elemente beinhaltet, wird als *Trefferkandidat* bezeichnet. Eine Suchspezifikation kann ggf. auf mehrere Arten mit Elementen im Prozess in Übereinstimmung gebracht werden, d. h. dass verschiedene Trefferkandidaten existieren, die verschiedene Permutationen typkompatibler Elemente im Prozess repräsentieren. In diesem Zusammenhang sind zwei Formen des weiteren Vorgehens möglich: Entweder werden die verbleibenden Regeln für alle Trefferkandidaten individuell angewendet (MATCH_ALL-Semantik) oder sie werden genau einmal angewendet (MATCH_ONCE-Semantik).

Das Beispiel in Listing 5.2 zeigt ein Muster mit MATCH_ALL-Semantik und Ausschluss einer Bedingung. Es beschreibt eine spezifische Aktivität „start_compensation“, die

nur dann ausgewählt wird, wenn nicht eine andere Aktivität „not_contact_insurer“ existiert, die durch eine entsprechende Transition vorgeschaltet ist. Wenn diese Konfiguration in einem Prozess mehrfach vorkommt, werden die restlichen Regeln auf alle diese Vorkommnisse individuell angewendet.

Transformationsregeln werden durch einen Ersetzungsteil abgeschlossen, indem definiert wird, wie ein Trefferkandidat transformiert werden soll. Hierin sind Ersetzungsspezifikationen für alle Elementbindungen eines Trefferkandidaten enthalten. Die Standardoperation ist das Entfernen der Elemente, d. h. dass alle Elemente, die hier keine Erwähnung finden, im Prozess entfallen und ein leerer Ersetzungsteil den gesamten Trefferkandidaten entfernt. Um die Elemente einer Suchspezifikation zu bewahren, müssen sie im Ersetzungsteil explizit referenziert werden, wobei alle Elementattribute modifiziert werden können. Zusätzlich können auch neue Elemente in den Prozess eingefügt werden. Bei der Neuerstellung von Elementen sind deren Attribute entweder leer oder werden mit Kopien der Attributwerte typgleicher Elemente initialisiert. Danach können sie beliebig modifiziert werden. Syntaktisch besteht die Neuerstellung von Elementen aus einem Namen, einem Typ mit optionaler Referenz auf ein zu kopierendes Element und verschiedenen Wertezuweisungen an Attribute (siehe Listing 5.3).

```
<binding name> = <element type> ( <binding name> ) :
  <attribute name> = <expression>,
    ...
  <attribute name> = <expression>;
```

Listing 5.3: Schematischer Ersetzungsteil

Um die Platzierung von Elementen innerhalb des Prozesses zu beeinflussen, wird ein spezielles Attribut „`container`“ eingeführt. Dieses erlaubt für alle Elemente den Zugriff und die Änderung des übergeordneten Elements. Wenn Referenzen auf optionale Elemente des Trefferkandidaten verwendet werden, dann muss auch die Transformationsoperation als optional deklariert werden.

Listing 5.2 zeigt ein Beispiel, in dem der Trefferkandidat nur ein einzelnes Element beinhaltet. Der beispielhafte Ersetzungsteil bewahrt diese Aktivität dadurch, dass diese am Anfang referenziert wird. Zusätzlich wird die Erstellung einer weiteren Aktivität sowie einer Transition angewiesen, wobei die Transition die beiden Aktivitäten verbindet. Die Details der Aktivität zum Versand einer Bestätigung sind im Beispiel ausgelassen.

Nachdem die Semantik der regelbasierten Transformationssprache anhand von Beispielen eingeführt wurde,[11] soll nun anhand der Syntax ein Überblick des kompletten Sprachumfangs gegeben werden. Die exakte Spezifikation der Syntax liegt als

[11] Für weitere Beispiele siehe [BP04, S. 54 ff.].

```
RuleMain ::= ( RuleSet | Rule )
RuleSet ::= 'RULESET' Name Rule+
Rule ::= 'RULE' Name ProcessSel? MatchSpec ReplaceSpec
Name ::= QuotedStr


ProcessSel ::= 'FOR_ALL_PROCESSES' Identifier
               EqualsExpr (',' EqualsExpr)* ';'
MatchSpec ::= ( 'MATCH_ONCE' | 'MATCH_ALL' )
              ( MatchElemSpec ';' )+
MatchElemSpec ::= SingleMatchElemSpec | ExclusionGroup
ExclusionGroup ::= 'EXCLUSION' (Name)?
                   ( SingleMatchElemSpec ';' )+ ';'
SingleMatchElemSpec ::= XpdlElementWithCard Identifier
                        ( 'IN' Identifier )? ':' EqualsExpr
                        ( ',' EqualsExpr )*
ReplaceSpec ::= 'REPLACE_WITH'
                ( (ReplaceElementSpec ';' )+ | ';' )
ReplaceElementSpec  ::= ( 'FOR_EACH' Identifier 'IN' )?
                         Identifier CardOrPrefix
                         ( ':' ReplAssignList )?
                         | XpdlElement Identifier
                         ':' ( ReplAssignList )?
ReplAssignList ::= Assign ( ',' Assign )*
CardOrPrefix ::= ( Cardinality | Prefix )?
XpdlElementWithCard ::= XpdlElement Cardinality?
Cardinality ::= ( '{' ('+' | '-' | '*' | '?') '}' )
Prefix ::= "$prefix"


Assign ::= PropertyPath '='
           ( EqualsExpr | XpdlElementCreation )
EqualsExpr ::= AndOrExpr (
               ( '==' | '<' | '>' | '<=' | '>=' | '!=' )
               AndOrExpr )*
AndOrExpr ::= AddSubExpr
              ( ('and' | 'or') AddSubExpr )*
AddSubExpr ::= MultDivExpr
               ( ('+' | '-') MultDivExpr )*
MultDivExpr ::= ContainsExpr
                ( ('*' | '/') MultDivExpr )*
ContainsExpr ::= InnerExpr ( 'contains' InnerExpr )*
```

```
InnerExpr ::= ListExpr | PropertyPath | QuotedStr | Function
              ( ( (UnarySign)? (Digits | ParenthExpr) )
              | ( 'not'? TrueOrFalse | 'not' ParenthExpr ) )
              | Function
TrueOrFalse ::= 'true' | 'false'
ListExpr ::= ( '[' ( EqualsExpr (',' EqualsExpr)* )? ']' )
Function ::= ( 'str' | 'int' | 'bool' | 'date' )
             '(' EqualsExpr ')'
UnarySign ::= '+' | '-'
ParenthExpr ::= ( '(' EqualsExpr ')' )
PropertyPath ::= PathElem ( '.' PathElem )*
PathElem ::= ( Identifier ( Key )? ) | XpdlElement
Key ::= '[' AddSubExpr ']'
XpdlElementCreation ::= XpdlElement '(' (innerExpr)? ')'


Identifier ::= ( Character | '_' )
               ( Character | '_' | DIGIT )*
Character ::= ( 'a' | ... | 'z' ) | ( 'A' | ... | 'Z' )
Digits ::= '0' | ... | '9'
XpdlElement ::= 'Activities' | 'Activity' | 'ActivitySet'
| 'ActivitySets' | 'ActualParameter' | 'ActualParameters'
| 'Application' | 'Applications' | 'ArrayType'
| 'BasicType' | 'BlockActivity' | 'Condition'
| 'ConformanceClass' | 'DataField' | 'DataFields'
| 'DataType' | 'DeclaredType' | 'EnumerationType'
| 'ExtendedAttribute' | 'ExtendedAttributes'
| 'ExternalPackage' | 'ExternalPackages'
| 'ExternalReference' | 'FormalParameter'
| 'FormalParameters' | 'Join' | 'ListType' | 'Member'
| 'No' | 'PackageHeader' | 'Participant'
| 'ParticipantType' | 'Participants' | 'ProcessHeader'
| 'RecordType' | 'RedefinableHeader' | 'Responsibles'
| 'Route' | 'SchemaType' | 'Script'
| 'SimulationInformation' | 'Split' | 'SubFlow'
| 'TimeEstimation' | 'Tool' | 'Tools' | 'Transition'
| 'TransitionRef' | 'TransitionRefs'
| 'TransitionRestriction' | 'TransitionRestrictions'
| 'Transitions' | 'TypeDeclaration' | 'TypeDeclarations'
| 'UnionType' | 'XpdlPackage' | 'XpdlProcess'
| 'XpdlProcesses'
QuotedStr ::= '"' ( ESC | ~('"'|'\\') )* '"'
```

Listing 5.4: EBNF-Grammatik der FRESCO-Transformationsregelsprache

Grammatik in EBNF vor und wird der Vollständigkeit halber in Listing 5.4 komplett dargestellt.[12] Im Folgenden werden anhand dieser die wichtigsten Aspekte erläutert.

Die Basis der Sprache bildet eine eindeutig benannte Regel (`Rule`) oder Regelmenge (`RuleSet`). Zur besseren Identifizierung der dadurch repräsentierten Änderungsschritte und -strategien sind deren Namen (`Name`) Zeichenketten inkl. Leerzeichen, die auch längere und aussagekräftigere Bezeichnungen erlauben. Einzelne Regeln sind, wie im Beispiel beschrieben, aus Selektor (`ProzessSel`), Suchmuster (`MatchSpec`) und Ersetzung (`ReplaceSpec`) zusammengesetzt. Deren syntaktische Grundstrukturen und insbesondere der Aufbau von Suchspezifikationen (`SingleMatchElemSpec`) und Ersetzungsspezifikationen (`ReplaceElementSpec`) wurden schon weiter oben eingeführt. Letztere können sich neben einzelnen auch auf Aufzählungen gebundener Elemente beziehen (`FOR_EACH`). Werden in diesem Zusammenhang neue Elemente erzeugt, so wird mit der `$prefix` Option die eindeutige Benennung durch Voranstellen eines Zählers sichergestellt. Zur Erzeugung einzelner leerer Elemente existiert eine alternative kompakte Schreibweise.

[12] In der Grammatik ist die Deklaration von ESC ausgelassen. Dies entspricht den üblichen ESC-Anweisungen in Zeichenketten. Der EBNF-Dialekt macht geringfügige Anleihen an die Syntax des ANTLR Translator Generators [Par06].

In den Ausdrücken von Such- und Ersetzungsspezifikationen können zusätzlich zu Elementvergleichen (`==,!=`), numerischen Vergleichen (`<,>,<=,>=`), arithmetischen (`+,-,*,/`), unären (`+,-`) und boolschen (`and, or, not`) Operatoren auch Zeichenkettenoperationen (`contains,+`) verwendet werden. Die Anwendung dieser Operationen kann auf atomare Werte wie Listen, Zeichenketten/Zahlen, boolsche Werte `true` und `false` erfolgen. Werte, die im Elementbaum des Prozesses über ihren Pfad (`PropertyPath`) referenziert werden, sind ebenfalls verwendbar. Zusätzlich sind Prozesselemente über ihre obligatorischen `ID` Attribute oder einen Index zugreifbar (z. B. `myProcess.activities['start'].name`). Typen der Regelsprache beinhalten XPDL-Elemente, boolsche und ganzzahlige Werte, Datumsangaben und Zeichenketten sowie deren Listen. Typzuweisungen (`str`, `int`, `bool`, `date`) erlauben deren Konvertierung.

Zur Durchführung von Änderungen, die durch Transformationsregeln in der oben beschriebenen Syntax spezifiziert wurden, wird die aggregierte Regelmenge T auf eine Menge von XPDL Workflow-Prozessen R wie folgt angewendet:

1. Transformationsregeln werden einzeln und in der Reihenfolge ihrer Spezifikation aus der Regelmenge T entnommen und individuell auf die Prozessmenge R angewendet. Für eine Regel, deren Selektionsteil n Konditionen C_i beinhaltet, wird das Selektor-Prädikat $PS = C_1 \wedge C_2 \wedge \ldots \wedge C_n$ für jeden Prozess in R evaluiert. Dies resultiert in der Teilmenge S aller Prozesse, für die die Bedingung zutrifft.

2. Im Vergleichsteil der Regel werden alle Suchspezifikationen mit Existenzmodifikator (`+`) durch einen obligatorischen und eine Menge beliebiger weiterer Vergleiche (`*`) ersetzt. Der Suchalgorithmus identifiziert dann für jede der modifizierten Suchspezifikationen eine Basismenge B_{MS}. Jede Basismenge enthält alle Elemente, die dem Typ der Suchspezifikation entsprechen und im aktuellen Prozess sichtbar sind.

3. Auf Basis des Suchmusters werden nun Suchspezifikationen selektiert. Dabei werden beliebige Vergleiche (`*`) zunächst ignoriert. Die s obligatorischen und o optionalen (`?`) Vergleiche werden in der Reihenfolge ihrer Spezifikation zu einer Liste MS_{s+o} hinzugefügt. Auf Basis dieser Liste und der korrespondierenden Liste von Basismengen $[B_1, B_2, \ldots, B_{s+o}]$ werden die Permutationen der Elemente $[e_1, e_2, \ldots, e_{s+o}]$ mit $e_1 \in B_1, e_2 \in B_2 \backslash \{e_1\}, e_3 \in B_3 \backslash \{e_1, e_2\}, \ldots, e_{s+o} \in B_{s+o} \backslash \{e_1, e_2, \ldots, e_{s+o-1}\}$ in einer Menge MC_{s+o} von *Trefferkandidaten* zusammengeführt. Jeder Trefferkandidat ist also eine Permutation von $s+o$ Elementen, die jeder Suchspezifikation MS_i ein Element e_i passenden Typs zuordnet.

4. Durch jede Kombination der Suchspezifikationen MS_{s+o} mit einem Trefferkandidaten $mc \in MCs + o$ wird ein individueller Namensraum definiert, in dem jeder Elementname einer Suchspezifikation ein individuelles Element des Prozesses referenziert. Dieser Namensraum beinhaltet auch den betrachteten

Prozess, der durch seinen Bezeichner im Selektionsteil referenziert wird. Die Konditionen jeder Suchspezifikation werden auf gleiche Weise wie die Konditionen des Selektionsteils evaluiert. Dabei wird der Namensraum dazu genutzt, die Elementnamen zu dereferenzieren, die in den Suchspezifikationen verwendet werden.

Wenn die Konjunktion dieser Konditionen für einen Trefferkandidaten mc zutrifft, muss als Nächstes geprüft werden, ob hierfür eine Suchspezifikation mit Ausschlussmodifikator (-) vorliegt. Die Überprüfung von Ausschlüssen vollzieht sich entsprechend der oben beschriebenen Suche. Für einen einfachen Ausschluss bzw. eine Gruppe von Ausschlüssen mit x Suchspezifikationen und entsprechende Basismengen $B_{ex_1}, \ldots, B_{ex_x}$ werden alle Permutationen $[ex_1, \ldots, ex_x]$ mit $ex_1 \in B_{ex_1}, \ldots, ex_x \in B_{ex_x} \backslash \{ex_1, \ldots, ex_{x-1}\}$ betrachtet. Die Permutationen bilden jeweils eine Erweiterung des Namensraums, der durch Trefferkandidat mc definiert wird und erlauben eine Evaluation der Ausschlusskonditionen. Wenn die Konjunktion all dieser Bedingungen bei einer der Permutationen und für mindestens eine der Ausschlüsse wahr ist, wird mc verworfen.

Wenn kein Ausschluss zutrifft, wird die Liste der beliebige Vergleiche ($*$) in der Reihenfolge ihrer Spezifikation abgearbeitet. Alle Elemente, die sich in der Basismenge der Suchspezifikation befinden und für die deren Klausel zutrifft, werden zur Ergebnismenge der Suchspezifikation hinzugefügt; es sei denn, sie sind schon im Kontext des Trefferkandidaten gebunden. Am Ende wird der Trefferkandidat zur *Treffermenge* M hinzugefügt, wobei jedes Element (oder Elementmenge) in einem Tupel zusammen mit seinem Namen aus der Suchspezifikation abgelegt wird.

5. Im nächsten Schritt wird die Menge der optionalen Suchspezifikationen durch Entfernung des letzten Elements reduziert. Dann wird der Suchalgorithmus aus Schritt 4 erneut angewendet. Ein Treffer wird aber nur dann zu M hinzugefügt, wenn M nicht schon einen identischen Treffer enthält, der lediglich mehr optionale Elemente beinhaltet. Dies wird wiederholt bis $o = 0$. Auf diese Weise werden am Ende nur noch die obligatorischen Suchspezifikationen als minimale Bedingung des Suchmusters evaluiert. Wenn im Prozess irgendeine Kombination von Elementen vorkommt, die zumindest die obligatorischen Suchspezifikationen und darüber hinaus ggf. optionale Suchspezifikationen erfüllt, so ist zu diesem Zeitpunkt mindestens ein Trefferkandidat in M.

6. Wenn die Menge M überschneidende Trefferkandidaten enthält (d. h. verschiedene Treffer enthalten teilweise die gleichen gefundenen Elemente) und die Suchvariante `MATCH_ALL` ist, gibt der Algorithmus einen Fehler aus und bricht die Verarbeitung der Regel für diesen und alle anderen vorselektierten Prozesse ab.

7. Wenn die Menge M mehr als einen Trefferkandidaten enthält und die Suchvariante `MATCH_ONCE` ist, gibt der Algorithmus eine „Nicht-Determinismus"-Warnung aus. Er wählt jedoch eine Variante aus und setzt die Verarbeitung der Regel fort.

8. Die letztendliche Ersetzung erfolgt in drei Schritten. Zunächst wird ein Namensraum gebildet, der alle Namen der im Trefferkandidaten enthaltenen Elemente umfasst. Als nächstes werden alle neuen Elemente der Ersetzungsspezifikationen erzeugt, initialisiert und zum Prozess hinzugefügt. Name und neues Element werden ebenso dem Namensraum hinzugefügt, um sie in weiteren Zuweisungen referenzieren zu können. Schließlich werden Elemente des Trefferkandidaten, die in den Ersetzungsspezifikationen erwähnt werden, erneut zugewiesen. Solche Elemente, die nicht in den Ersetzungsspezifikationen vorkommen, werden aus dem Prozess entfernt.

Die Analyse des Algorithmus zeigt, dass die Evaluation aller zu einem Mustern gehörigen Bindungskombinationen von Elementen zu erheblicher Komplexität führen kann. Konkret kann die obere Grenze für zu evaluierende Trefferkandidaten wie folgt angegeben werden:

$$\sum_{i=0}^{o} |B_1| \cdot |B_2| \cdots |B_{s+i}|$$

Hierbei wird angenommen, dass alle in den Suchspezifikationen geforderten Typen unterschiedlich sind. Dies kann nur eintreffen, solange die Anzahl von Suchspezifikationen kleiner ist als die Anzahl der Elementtypen. Bei Überschneidungen der Elementtypen in den Suchspezifikationen verringert sich diese Anzahl. Im besten Fall sind alle Typen identisch, wobei für die untere Grenze gilt:

$$\sum_{i=0}^{o} |B_1| \cdot (|B_2| - 1) \cdots (|B_i| - (s + i - 1))$$

Als Ergebnis lässt sich festhalten, dass der Skalierung Grenzen gesetzt sind. Im Falle sehr großer Prozesse und komplexer Transformationsregeln, in denen die Basismengen hunderte Elemente enthalten und dutzende Suchspezifikationen zu evaluieren sind, ist der Zeitaufwand unter Umständen kritisch. Die Situation wird partiell durch die Verwendung des Prozess-Selektionsteils in der Transformationsregel gemildert, was in einer Verringerung der zu untersuchenden Prozesse resultiert.

Zudem wurde eine Lösung gefunden, den Aufwand auch auf der Ebene einzelner Prozessvergleiche erheblich einzuschränken. Hierbei wird die Tatsache ausgenutzt, dass die Konditionen der Suchspezifikationen nicht immer kontextabhängig sind. Ein Beispiel hierfür ist die Suche von Elementen auf Basis des Namens wie etwa im Fall von `Activity notify_insurer: name = "Notify insurer";`. Hierbei sind keine Referenzen auf andere Elemente im Prozess enthalten, die die Existenz eines Trefferkandidaten und des dadurch definierten Namensraums voraussetzen.

Dementsprechend kann die Evaluation solcher Konditionen dazu verwendet werden, die Basismengen signifikant einzuschränken, bevor daraus die Permutationen der Trefferkandidaten gebildet werden. Das Resultat ist ein erheblich verbesserter Skalierungsgrad. Im Kontext der Untersuchungen, die im Rahmen der vorliegenden Arbeit durchgeführt wurden, zeigte sich hiermit eine durchweg praktikable Performanz.

5.2.2. Entwurf einer E-Service-Ausführungsplattform

Einer der wesentlichsten Aspekte des E-Service-Managements und der Mittelpunkt des E-Service-Entwicklungslebenszyklus besteht in der E-Service-Ausführung. Dies bewirkt die automatische Durchführung von Interaktionen zwischen VDPN-Teilnehmern und dient zur Koordination kooperativer Tätigkeiten der virtuellen Dienstleistungsproduktion. Zudem stellt die Automation des Dienstleistungsprozesses durch Einsatz technischer Hilfsmittel neben und nach dessen expliziter Gestaltung durch Entwurf von E-Service-Modellen ganz generell einen entscheidenden Faktor zur Produktivitätssteigerung von Dienstleistungsunternehmen dar.[13]

Die Ausführung von E-Services basiert auf einer Abbildung der Konzepte des E-Service-Metamodells auf Komponenten der IV-Infrastruktur von VDU. Diese Komponenten finden sich zum Teil in den fundamentalen Kommunikations- und Kooperationsmechanismen der horizontalen Basisinfrastruktur, die in VDU per se vorhanden sind. Darüber hinaus müssen aber einige Komponenten als spezifische Mechanismen eines ESMS entwickelt werden, die zusammen dessen *E-Service-Ausführungsplattform* bilden. In dieser Sektion werden im Folgenden Konzepte für derartige E-Service-Ausführungsplattformen diskutiert, die speziell auf Service-orient. E-Service-Modellen zugeschnitten sind.

Die Untersuchungen beginnen in Sektion 5.2.2.1 für den generellen Fall allg. Service-orient. E-Services. Hierbei werden zunächst Vorgänge und Mittel der Implementierung von E-Service-Entwürfen durch Service-orient. E-Service-Entwicklungsmethoden und Web Service-Basistechniken verglichen. Danach wird ein erweitertes Konzept erarbeitet, das Modelle für lokale Plattformen einzelner VDPN-Teilnehmer und eine globale ESMS-Plattform auf Basis Service-orient. Grid-Techniken verbindet.

Der zweite Teil der Untersuchung fokussiert in Sektion 5.2.2.2 den Entwurf einer Ausführungsplattform für den konkreten Fall von FRESCO-E-Services. Die FRESCO-Plattform realisiert das Grid-basierte Modell einer globalen ESMS-Plattform. Die E-Service-Ausführung beruht hier im Wesentlichen auf zwei Erweiterungen des BPEL-Kompositionsmodells: Zum einen wird das BPEL-Komponentenmodell zur Integration in das OGSA Grid erweitert. Zum anderen wird ein dynamisches Aggregationsmodell für die zusammengesetzten Interaktionsprozesse von E-Services ergänzt.

[13] Vgl. Sektion 2.2.2.2.

5.2.2.1. E-Service-Implementierung in Service-orient. Grids

Die Untersuchung Service-orient. E-Service-Ausführungsplattformen beginnt im Folgenden mit einer generellen Untersuchung der Implementierung Service-orient. E-Services im methodischen Kontext des Service-orient. E-Service-Entwicklungslebenszyklus und auf der Basis Service-orient. Techniken für VDU. Hierbei wird als erstes der Entwicklungsvorgang zwischen E-Service-Entwurf und -Ausführung in Bezug auf funktionale und nicht-funktionale Anforderungen an eine E-Service-Plattform untersucht. Diese Anforderungen werden dabei auf die grundlegenden Web Service-Techniken übertragen. Als nächstes wird die Charakteristik der horizontalen IV-Infrastruktur im VDU beleuchtet. Auf dieser Basis wird dann ein Service-orient. Plattformmodell für VDPN-Teilnehmer formuliert. Hierbei wird argumentiert, dass im VDU-Kontext insbesondere die Verwendung Service-orient. Grid-Techniken von Vorteil ist. Das resultierende Plattformmodell dient im Folgenden als Annahme für den Entwurf von E-Service-Plattformen. Der Schwerpunkt liegt dabei auf der Konzeption eines generischen E-Service Grid-Modells.

Strategien Service-orient. E-Service-Implementierung Die Ausführung eines E-Service bezieht sich auf die Erbringung einer Dienstleistung für einen spezifischen Kunden und mit der Beteiligung spezifischer Provider. Die hierzu notwendigen Interaktionsprozesse sind in Form eines logischen E-Service-Modells festgelegt. Dieses Modell bildet ein generelles Schema, in dem Teilnehmer durch Rollen abstrahiert werden und das im Folgenden als *E-Service-Schema* bezeichnet werden soll. Im Zuge der *E-Service-Aggregation* erfolgt die Zuweisung einer konkreten Teilnehmerkonstellation zu den Rollen des E-Service-Schemas. Das Ergebnis ist eine *E-Service-Instanz*. Für ein Schema können mehrfache Instanzen gebildet und ausgeführt werden. Ein Teilnehmer kann sich an mehrfachen Instanzen desselben E-Service-Schemas oder verschiedener Schemata beteiligen.

Im Service-orient. E-Service-Entwicklungslebenszyklus erfolgt die Bestimmung von E-Service-Instanzen im Zuge der Implementierung eines spezifischen VDPN (Stufe 3). Zur Ausführung von *E-Service-Instanzen* müssen diese dann implementiert werden. Die Implementierung von E-Service-Instanzen zum Zweck der Ausführung beginnt mit einem detaillierten physikalischen Entwurf. Dann ist der E-Service zunächst zu konstruieren und zu testen. Die Konstruktion betrifft dabei ein Service-orient. Anwendungssystem und bezieht sich auf eine konkrete Ausführungsplattform. Vor der Ausführung müssen ggf. komplementäre Maßnahmen zur Dienstleistungserbringung (Provision) festgelegt werden, was z. B. die Abrechnung und Zahlung betreffen kann. Zudem müssen die Komponenten des E-Service-AS auf den lokalen und globalen Anteilen der Ausführungsplattform installiert und konfiguriert werden (Deployment). Nach diesem Schritt kann die eigentliche Ausführung beginnen.

Unter Bezugnahme auf die vorangegangene Diskussion modellgetriebener E-Service-Entwicklung beginnt die E-Service-Implementierung mit dem Entwurf eines

plattformspezif. Modells. Auf dem Weg vom E-Service-Modell zum Service-orient. Anwendungssystem wird der Implementierungsvorgang dann im Wesentlichen von dem Software-Rahmenwerk der Ausführungsplattform bestimmt. Dieses Rahmenwerk stellt einen Großteil der generischen Funktionalität eines Service-orient. E-Service-Anwendungssystem zur Wiederverwendung bereit. Die E-Service-Implementierung besteht dann in der Spezialisierung des Rahmenwerks in Bezug auf die spezifischen Aspekte einer E-Service-Instanz. Diese Spezialisierung kann grundsätzlich in unterschiedlicher Art und Weise erfolgen. Zunächst kann die Spezialisierung des Rahmenwerks entweder durch dessen Konfiguration oder Erweiterung erfolgen. Konfiguration basiert auf Spezifikationen, die durch das Rahmenwerk zur Laufzeit interpretiert werden. Erweiterungen erfolgen durch Hinzufügen von Code, der vor der Ausführung kompiliert werden muss. Beide Vorgehensweisen haben Vor- und Nachteile, die hier nicht vertieft werden sollen. Es sei nur bemerkt, dass die Konfiguration tendenziell eine agilere und flexiblere Implementierung erlaubt, während die Erweiterung tief greifendere und weitergehende Spezialisierungen ermöglicht. Unabhängig von der Art der Spezialisierung soll diese bei der modellgetriebenen Entwicklung aus dem Modell hervorgehen. Dabei sollen Spezifikationen und Code zur Konfiguration und Erweiterung des Rahmenwerks in möglichst weitgehender, direkter und automatisierbarer Weise aus dem Modell erzeugt werden. Zu diesem Zweck dienen verschiedene Entwicklungswerkzeuge. Die wichtigsten sind Transformatoren und Generatoren, die Spezifikationen und Code aus dem Modell erzeugen und zur Verarbeitung durch Interpreter und Compiler bereitstellen. In den meisten Fällen ist daneben eine manuelle Programmierung spezifischer Aspekte notwendig. Hierzu existiert eine Vielzahl an Werkzeugen und Assistenten, die die Arbeit erleichtern und beschleunigen. U. a. stellen Ausführungsplattformen in der Regel Assistenten zur Installation und Konfiguration (Deployment) des spezialisierten Rahmenwerks zur Verfügung.

Der Begriff der Ausführungsplattform bezieht sich auf System- und Middleware-Plattformen, die einen homogenen Kontext zur Entwicklung und Ausführung von (verteilten) Anwendungssystemen bereitstellen. In der Regel beinhaltet dies ein Programmiermodell sowie Entwicklungs- und Laufzeitmechanismen zu dessen Unterstützung.[14] Im Falle Service-orient. E-Services wird eine Web Service Middleware vorausgesetzt. Diese muss zur Unterstützung des grundsätzlichen Service-orient. Realisierungskonzepts für E-Services[15] im Wesentlichen Modelle und Mechanismen zur Implementierung komplexer Web Service-Interaktionen bereitstellen. Dazu gehören Modelle für die Beschreibung, Publikation, Auffindung, Auswahl, Bindung und atomare Interaktion von/mit einzelnen Web Services, die Konversation zwischen multiplen Web Services und die Komposition von Web Services in Bezug auf Komponenten, Choreografie, Daten und Aggregation. Entsprechende Mechanismen beinhalten u. a.

[14] Eine ausführliche Einführung von Middleware-Plattformen findet sich in Sektion 3.3.2.
[15] Vgl. Sektion 4.3.1.2.

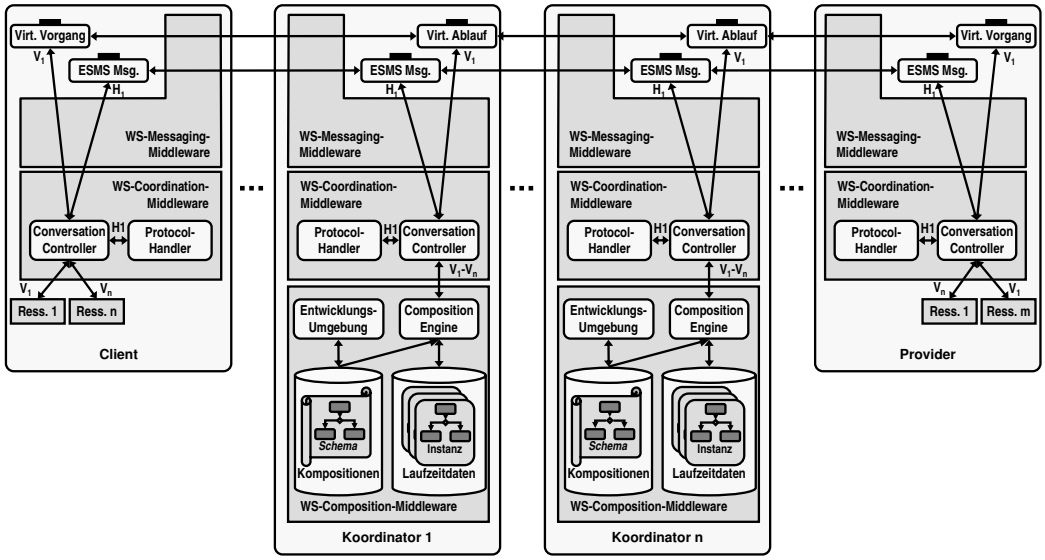

Abbildung 5.5.: Basisarchitektur Service-orient. E-Service-Implementierung

Software-Rahmenwerke und eine Server-Plattform (Hosting Environment) für die Implementierung und Bereitstellung atomarer Web Service-Komponenten auf Basis lokaler Plattformen und Ressourcen. Hinzu kommt eine Laufzeitumgebung zur Web Service-Komposition (Composition Engine).

Die Implementierung eines E-Service auf Basis von Web Service Middleware erfolgt im Wesentlichen als komplexe Form der Web Service-Komposition. Abbildung 5.5 zeigt deren Grundprinzip. VDPN-Teilnehmer, die als Client- oder Provider-Akteure auftreten, bringen virtuelle Vorgänge als Web Services ein. Diese basieren auf lokalen Ressourcen, die über Mechanismen lokaler System- oder Middleware-Plattformen an den Web Service gebunden werden. Diese Bindung erfordert in der Regel eine Erweiterung des Software-Rahmenwerks der WS-Plattform. Die Implementierung wird dabei meist durch Generatoren unterstützt, die den generischen Code der Bindung erzeugen (so genannte Stubs) und in Bezug auf die spezifischen Aspekte der lokalen Plattform und der Ressource angepasst werden müssen. VDPN-Koordinatoren realisieren die virtuellen Abläufe des virt. Dienstleistungsproduktionsnetzwerks ebenfalls als Web Services. Diese basieren jedoch nicht auf lokalen Ressourcen, sondern auf einer WS-Komposition. Dessen Schema geht mehr oder weniger direkt aus dem E-Service-Modell hervor und wird von den Laufzeitmechanismen der WS-Komposition interpretiert. Darüber hinaus ist meist eine Konfiguration der Laufzeitumgebung, z. B. in Bezug auf die Aggregation des zusammengesetzten Web Service, zu leisten. Eine derartige Implementierung stellt auf Grund der E-Service-Charakteristik hohe Anforderungen an die Service-orient. Middleware. Die wesentlichen Aspekte können hierbei wie folgt charakterisiert werden:

- *Web Service-Komponenten* repräsentieren individuelle Ressourcen verschiedener Organisationen. Interaktionen mit den Ressourcen müssen zuverlässig und sicher erfolgen. Zudem muss die Identität aller Ressourcen jeweils in multiplen Konversationen und über den gesamten Verlauf des virt. Dienstleistungsprozesses erhalten bleiben, um deren Zustände einzubeziehen.

- *Web Service-Kompositionen* repräsentieren abgeschlossene Interaktionsprozesse zwischen den Ressourcen. Der Verlauf eines virt. Dienstleistungsprozess ergibt sich aus der konstanten Kopplung multipler Instanzen verschiedener Interaktionsprozesse in unterschiedlichen Organisationen.

Diese Charakteristik der E-Service-Implementierung muss sich in den Modellen und Mechanismen von Web Services und WS-Komposition widerspiegeln, die der Ausführungsplattform zugrunde liegen. Das Web Service-Modell muss also nicht nur zuverlässige und sichere Interaktionen beinhalten, sondern vor allem zustandsbehaftet sein. Dies bildet eine grundlegende Anforderung an die Ausführungsplattformen aller VDPN-Teilnehmer. Das Web Service-Kompositionsmodell muss das zugrunde liegende erweiterte Web Service-Modell im Rahmen des Komponentenmodells aufgreifen. Daneben betreffen die Anforderungen vor allem das Aggregationsmodell. Hierbei reicht die isolierte Betrachtung der Aggregation einzelner Web Service-Kompositionen nicht aus. Die Aggregation muss für alle WS-Kompositionen, die einzelne Interaktionsprozesse eines virt. Dienstleistungsprozesses implementieren, in zusammenhängender und homogener Weise erfolgen. Dies bezieht auch die wechselseitige Aggregation von Kompositionen selbst ein. Derartige Anforderungen gehen über die fundamentalen Modelle und Mechanismen von Web Service-Plattformen und Web Service Composition Middleware hinaus. Daher sind Erweiterungen im Sinne einer E-Service Middleware notwendig. Dies betrifft zum einen die lokalen Web Service-Plattformen, die grundsätzlich als Basis bei allen VDPN-Teilnehmern vorausgesetzt werden. Zum anderen betrifft das eine globale übergeordnete Plattformebene, auf der Modelle und Mechanismen zur Implementierung des ganzheitlichen E-Service bereitgestellt werden, die von VDPN-Brokern und -Koordinatoren benötigt werden. Im Folgenden werden diese beiden Plattformtypen nacheinander hergeleitet.

Virtualisierung von Ressourcen im VDPN als Grid Services Die lokalen System- und Middleware-Plattformen von Netzwerk-Dienstleistungsunternehmen können prinzipiell in beliebiger Form auftreten. Grundsätzlich wird angenommen, dass die hierin ablaufenden Anwendungssysteme im Sinne der unternehmerischen Geschäftsprozesse einzelner Teilnehmer integriert sind und diese also abbilden und repräsentieren, um sie zu planen, zu steuern und zu kontrollieren. Insbesondere wird die Existenz operativer Systeme[16] angenommen, die zur Administration und Disposition von Ressourcen im Zuge der Produktion von Dienstleistungsinhalten dienen.

[16] Vgl. Sektion 3.2.1.1, S. 90.

Zur Teilnahme von Dienstleistungsunternehmen im strategischen Dienstleistungsnetzwerk an sich sowie insbesondere in virt. Unternehmen müssen sich die lokalen Plattformen grundsätzlich in eine globale horizontale IT-Infrastruktur eingliedern, die sie u. a. zur wechselseitigen Kommunikation und Kooperation befähigt.[17] Solche Basismechanismen werden auch für die lokalen Plattformen von VDPN-Teilnehmern vorausgesetzt. Die wichtigsten Anforderungen an die Kommunikationsinfrastruktur betreffen eine leistungsfähige Vernetzung sowie zuverlässige und sichere Interaktion zwischen Anwendungssystemen verschiedener Dienstleistungsunternehmen. Für die Kooperationsinfrastruktur werden Grundfunktionen für den gemeinsamen Zugriff auf Daten, die wechselseitige Koordination und die allgemeine Kollaboration gefordert.

Konkret wird eine Service-orient. VDU-Basisinfrastruktur angenommen. Hierbei liefern Web Service-Techniken die Basismechanismen zur Beschreibung, Publikation, Auffindung, Auswahl, Bindung und Interaktion von Anwendungssystemen/Ressourcen verschiedener Dienstleistungsunternehmen. Die Web Service-Basisarchitektur ist jedoch für die Anforderungen an die VDU-Infrastruktur zu generisch. Hierin wird vor allem der Aspekt interoperabler Kopplung abstrakter Anwendungssysteme hervorgehoben. Diese Kopplungstechnik muss zur Nutzung im virt. Unternehmen dazu angewendet werden, um die Ressourcen einzelner Netzwerkunternehmen über ihren Lebenszyklus zu verwalten und sie als virtuelle Ressource auf Netzwerkebene zur Verfügung zu stellen. Auf Netzwerkebene müssen diese virtuellen Ressourcen dann in koordinierter Weise und unter Vorgabe von Einsatzbedingungen zu einer Leistung des virt. Unternehmens mit vorhersehbaren Qualitätsmerkmalen kombiniert werden. Für eine derartige Anwendung der Web Service-Techniken fehlen in der Basisarchitektur jedoch die notwendigen Abstraktionen. Insbesondere sind hier Erweiterungen notwendig, die die Fähigkeiten lokaler System- und Middleware-Plattformen zur aktiven Verwaltung von Ressourcen und Kontrolle qualitativer Eigenschaften in vereinheitlichter Form auf der übergeordneten WS-Ebene bereitstellen.

Eine solche Erweiterung der Web Service-Basisarchitektur, die auf die Ressourcenverteilung und -verwaltung in virt. Unternehmen zugeschnitten ist, kann durch Integration von *Grid-Techniken* erreicht werden. Grid-Techniken wurden ursprünglich zur systemtechnischen Unterstützung verteilter Anwendungssysteme der groß angelegten wissenschaftlichen Forschung entwickelt [FK99]. Diese Unterstützung bezieht sich auf die Teilung und koordinierte Nutzung von Ressourcen in allgemeinen *virtuellen Organisationen* [FKT01]. Seitdem haben Grid-Techniken einen deutlichen Wandel vollzogen. Zum einen wurde die Anwendungsdomäne erweitert: Grid-Techniken werden mittlerweile in einer Vielzahl von Domänen eingesetzt, in denen Anwendungssysteme sich dynamisch an Änderungen von Organisationsstrukturen anpassen müssen. Dazu gehören insbesondere auch B2Bi-Anwendungssysteme. Zum anderen wurde die technische Basis weiterentwickelt. Eine wesentliche Tendenz geht hier zur Realisierung von Grid-Plattformen auf Basis Service-orient. Architektur.

[17] Vgl. Sektion 3.2.2.1 und Sektion 3.2.2.2, S. 96.

Diese Entwicklung wird vor allem durch die *Open Grid Service Architecture (OGSA)* demonstriert und bestimmt [FKN+02b]. OGSA beschreibt ein Architekturkonzept, das von dem *Open Grid Forum (OGF)* in einem offenen Community-basierten Prozess entwickelt wird.[18]

In OGSA werden Ressourcen einer virt. Organisation durch *Grid Services* mit explizit definierter Semantik repräsentiert. OGSA legt eine Reihe obligatorischer und optionaler Funktionen und Konventionen fest, die von einer Grid-Plattform in Form von Web Services zur Verfügung gestellt werden. Die Kombination und Erweiterung dieser Web Services unter Berücksichtigung der Konventionen führt zu individuellen Grid Services. Diese dienen als Bausteine zur Integration verteilter Ressourcen in ein Grid-basiertes Anwendungssystem und stellen dabei gleichzeitig die OGSA-Mechanismen zur verteilten Ressourcenverwaltung zur Verfügung. Letztere umfassen im Wesentlichen Erstellung, Benennen und Auffindung transienter zustandsbehafteter Grid Service-Instanzen. Dabei wird Ortstransparenz gewährleistet. Ferner besteht die Möglichkeit zur freien Wahl von Kommunikationsprotokollen und zur Einbindung von Funktionalitäten lokaler System- und Middleware-Plattformen. Über die Web Service-Schnittstellen eines Grid Service sind zudem Funktionen zur Verwaltung seiner Lebenszeit, Änderung und Benachrichtigung nutzbar. Durch die Bindung von Grid Services an geeignete Kommunikationsprotokolle kann insbesondere eine passende Semantik (z. B. „einmal oder gar nicht“) und Sicherung (z. B. durch Authentifizierung oder Verschlüsselung) von Aufrufen erreicht werden. Die Gesamtheit der OGSA-Mechanismen bildet eine Basisinfrastruktur, die auf verteilte Anwendungssysteme in virt. Unternehmen zugeschnitten ist.

OGSA liegt ein rein pragmatischer Begriff der virt. Organisation im institutionellen Sinne zugrunde, der diese als „... dynamic ensembles of resources, services and people ...“ [FKN+02a] im wissenschaftlichen oder geschäftlichen Bereich charakterisiert. Diese Strukturen können insbesondere hierarchisch strukturiert und überlappend sein. Aus diesem Grund betont OGSA einen virtuellen Ressorcenbegriff. Dabei wird die Repräsentation einer Ressource von ihrer Realisierung getrennt. Letztere kann dann auf ganz verschiedene und auch wechselnde Art je nach den aktuellen Begebenheiten durch eine oder mehrere andere virtuelle oder wirkliche physikalische Ressource erfolgen. Dies betrifft nicht nur die Ressourcen der Anwendungsdomäne, sondern auch diejenigen zur Realisierung der Grid-Plattform selbst, die z. B. oft auf die Mechanismen lokaler Plattformen abgebildet werden müssen. Bei der Virtualisierung hilft insbesondere die Service-Orientierung von OGSA. Die Nutzung von Web Services erlaubt eine homogene Deklaration von Ressourcen mit multiplen Möglichkeiten der Bindung an Protokolle zu deren Zugriff. Zur Nutzung einer Ressource erlauben Web Services nicht nur die Wahl passender Schnittstellen und Protokolle, sondern

[18] Die Entwicklung von OGSA wurde vom Global Grid Forum (GGF) (ein)geleitet. Dieses wurde mittlerweile mit der Enterprise Grid Alliance (EGA) zum Open Grid Forum (OGF) vereinigt, das diese Aufgabe weiterführt [GGF06].

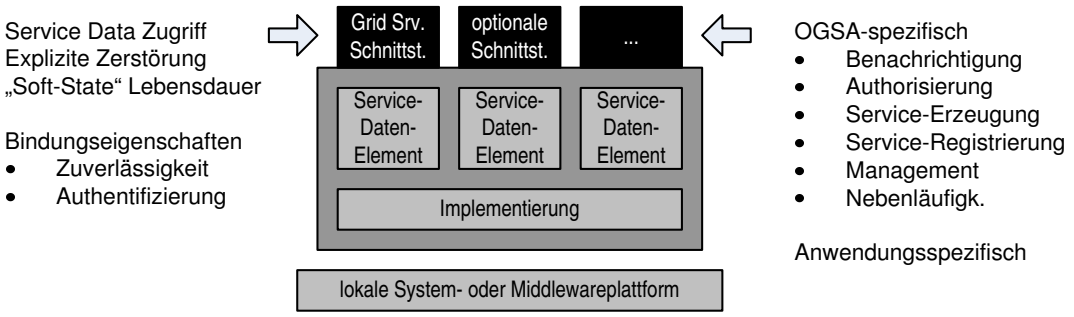

Abbildung 5.6.: OGSA Grid Service (nach [FKN+02a])

auch deren inhärent ortstransparente Implementierung in Bezug auf die akuten Anforderungen und Bedingungen.

OGSA spezifiziert die konkrete Art der Service-orient. Repräsentation von Ressourcen durch Definition von Web Service-Schnittstellen und -Konventionen. Die Schnittstellen beinhalten Auffindung *(discovery)*, dynamische Erzeugung *(creation)*, Lebenszeitverwaltung *(lifetime management)*, Benachrichtigung *(notification)* und Verwaltbarkeit *(manageability)* von Grid Services. Konventionen betreffen deren Benennung und Erweiterung. Hierbei müssen alle Grid Services eine obligatorische Schnittstelle (`GridService`) besitzen, die den Zugriff auf explizit spezifizierbare Laufzeitinformationen (Service Data) sowie Grundfunktionen des Lifetime Managements erbringt (vgl. Abb. 5.6).

Weitere Schnittstellen sind optional und müssen nur von Grid Services erbracht werden, die in einem Grid-basierten Anwendungssystem erweiterte Funktionen realisieren. Neben den OGSA-definierten Schnittstellen können weitere anwendungsspezifische Schnittstellen definiert werden, die zur Repräsentation der eigentlichen Ressource dienen. Die Implementierung eines Grid Service und dessen Ausführung kann durch beliebige Techniken und Plattformen erfolgen. Zudem können unterschiedliche Protokollbindungen mit zusätzlichen Eigenschaften (z. B. Zuverlässigkeit, Authentifizierung) eingebunden werden. Die Instanzen eines Grid Service besitzen dann eine individuelle Identität und sind zustandsbehaftet. Verschiedene Instanzen eines Grid Service werden durch eindeutige Namen referenziert, die als *Grid Service Handle (GSH)* bezeichnet werden. Sie besitzen nur eine begrenzte Lebensdauer, die bei Bedarf erneuert werden muss. Nach Ablauf der Lebensdauer oder durch explizite Anweisung werden Grid Service-Instanzen zerstört und die dahinterstehenden Ressourcen befreit. Konkret macht OGSA die Grid Service-Semantik u. a. durch folgende Verhaltensweisen und Schnittstellen explizit:

- *Grid Service-Auffindung* – um Grid Service-Instanzen finden zu können, werden sie durch einen *Registry Grid Service* erfasst, der die `Registry`-Schnittstelle implementiert. Um Eigenschaften einzelner Instanzen zu untersuchen, beinhaltet die obligatorische `GridService`-Schnittstelle die Operation `FindServiceData` zur

Abfrage von `Service Data`-Elementen. Vor dem Zugriff auf eine Instanz muss eine Referenz auf den Endpunkt von dessen Implementierung bei einem spezifischen *Mapper Grid Service* erfragt werden. Dieser muss die `HandleMap`-Schnittstelle mit der Operation `FindByHandle` bereitstellen, die einen GSH auf eine spezifische Referenz abbildet.

- *Dynamische Grid Service-Erzeugung* – um Grid Service-Instanzen bei Bedarf erzeugen zu können, implementieren spezielle *Grid Service Factories* die optionale OGSA `Factory`-Schnittstelle mit der Operation `CreateService`.

- *Grid Service-Lebenszeitverwaltung* – die Zerstörung von Grid Service-Instanzen erfolgt entweder explizit durch die Operation `Destroy` der `GridService`-Schnittstelle oder implizit nach Ablauf einer Lebensspanne. Die Lebensspanne kann bei Bedarf durch Aufruf der Operation `SetTerminationTime` der `GridService`-Schnittstelle verlängert werden. Dieses Vorgehen (genannt „Soft State-Protokoll“) sichert die letztendliche Freigabe aller Ressourcen auch im Fehlerfall des verteilten Anwendungssystems.

- *Grid Service-Benachrichtigung* – zur asynchronen Benachrichtigung von Zustandsänderungen zwischen Grid Service-Instanzen spezifiziert OGSA die optionalen Schnittstellen `Notification-Source` und `Notification-Sink`, die Operationen für Abonnements und Lieferungen von Benachrichtigungen enthalten.

Die durch OGSA vorgegebenen Funktionen und Eigenschaften von Grid Services müssen durch die lokalen System- oder Middleware-Plattformen der Teilnehmer virt. Organisationen auf Basis von deren Programmiermodell und Ausführungsumgebung realisiert werden. Solche Plattformen können etwa lokale Betriebssysteme sein, bei denen ein Grid Service z. B. als C-, Java- oder Fortran-Programm implementiert wird und Instanzen zu neuen Systemprozessen führen. Die Grid Service-Semantik muss dann innerhalb des Programms entweder manuell oder auf Basis von Bibliotheken implementiert werden.

In den meisten Fällen sind Plattformen jedoch durch eine Middleware gegeben, in der Anwendungssysteme auf Basis mächtiger Programmiermodelle flexibel, sicher und verwaltbar implementiert und ausgeführt werden können. Zudem stellen Middleware-Plattformen meist für viele Aspekte der OGSA-Funktionalität schon Mechanismen bereit. Hier kann durch eine Abbildung der geforderten Funktionalität auf die vorhandenen Mechanismen eine spezifische *OGSA-Plattform* realisiert werden, die dem Grid Service-Entwickler einige Aspekte der Implementierung abnimmt. Generell sollte eine solche Plattform Mechanismen bieten, um a) Grid Service Handles auf bindungsspezifische Referenzen abzubilden, b) Zugriffe auf Grid Service-Instanzen an lokale Ressourcen weiterzuleiten, c) Protokolle abzuwickeln und Daten für die Netzwerkübertragung zu konvertieren, d) die Lebensdauer zu verwalten und e) eine Authentifizierung durchzuführen.

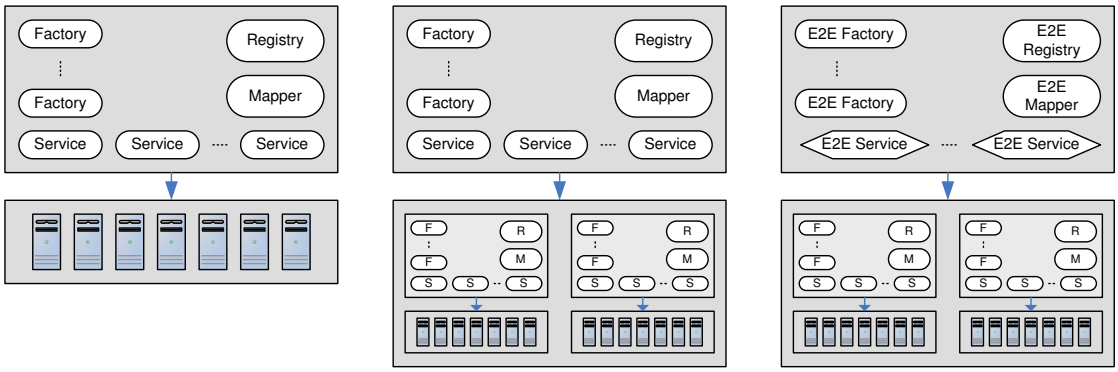

Abbildung 5.7.: VU-Strukturen mit OGSA Grids (nach [FKN+02b])

Durch das Prinzip der Virtualisierung von Ressourcen bzw. den sie repräsentierenden Grid Services sind auch für Grid-Plattformen verschiedene Realisierungsmöglichkeiten in konkreter oder virtueller Form möglich. Im einfachsten Fall ist eine lokale System- oder Middleware-Plattform vorhanden (vgl. Abb. 5.7 links). Diese beherbergt eine Reihe von Anwendungssystemen zum Zugriff auf lokale Ressourcen und bietet verschiedene Mechanismen zur Verwaltung von Instanzen. Auf Grid-Ebene werden typischerweise für alle Anwendungssysteme bzw. Ressourcen *Factory Grid Services* erzeugt und mit ihrem GSH bei einem *Registry Grid Service* registriert. Ein *Mapper Grid Service* übersetzt GSH in Referenzen entsprechend der spezifischen Protokollbindung. Wenn ein Teilnehmer über die Factory eine Ressource erzeugt, nutzt diese die Mechanismen der lokalen Plattform, um ein Anwendungssystem zu starten, das den Zugang zu einer lokalen Ressource erlaubt. Zum Zugriff auf das Anwendungssystem wird eine *Grid Service-Instanz* erstellt und in der Registry vermerkt. Der Teilnehmer erhält zudem eine Referenz, die ihm den Zugriff auf die Grid Service-Instanz auf Basis einer konkreten Protokollbindung erlaubt.

In Fällen, bei denen mehr als eine Organisation beteiligt ist und auf Grund räumlicher Trennung und unterschiedlicher administrativer Domänen mehrere lokale Plattformen existieren, kann eine einheitliche Sicht auf die Ressourcen einer virt. Organisation durch eine *virtuelle Grid-Plattform* realisiert werden (vgl. Abb. 5.7 Mitte). Hierin sind Factories für alle Grid Services der virt. Organisationen vorhanden. All diese sowie alle Grid Service-Instanzen sind in der virt. Registry vermerkt. Zugriffe auf die Grid Services der virt. Plattform werden jedoch stets an die entsprechenden realen Plattformen weitergeleitet. Eine noch komplexere Form der virt. Grid-Plattform ist in Abb. 5.7 rechts zu sehen. In diesem Fall werden auf virt. Ebene zusammengesetzte *End-to-End Grid Services* verwaltet. Deren Factories erzeugen eine Menge mit multiplen Grid Service-Instanzen verschiedener realer Plattformen. Die lokalen Instanzen werden durch die einzelne virt. Grid Service-Instanz zu einem zusammengesetzten Grid Service kombiniert.

Insgesamt lässt sich festhalten, dass OGSA eine spezifische Service-orient. Architektur darstellt, die die fundamentalen WS-Techniken um Abstraktionen und Mechanismen zur Repräsentation und Verwaltung zustandsbehafteter transienter Services erweitert, mit denen die Teilung von und der koordinierte Zugriff auf Ressourcen in virtuellen Organisationen möglich wird. Diese Erweiterung macht SOC-Techniken für den Einsatz als horizontale Basisinfrastruktur in virt. Unternehmen nutzbar. Aus diesem Grund wird OGSA-Konformität als Voraussetzung für die Ausführungsplattform von VDPN-Teilnehmern zur Partizipation im Rahmen von E-Service-Management-Systemen festgelegt.

Koordination virt. Ressourcen im E-Service Grid Auf Basis der genannten Voraussetzungen für die VDPN-Teilnehmer-Plattformen und der daraus resultierenden horizontalen VDPN-Basisinfrastruktur kann nun der Entwurf einer aufbauenden Plattformebene erfolgen, die die Implementierung und Ausführung von E-Services unterstützt. Hierbei wird die einleitende Diskussion über die Implementierung von E-Services auf Basis von WS-Techniken aufgegriffen und gezeigt, wie die identifizierten Anforderungen mithilfe einer Erweiterung der WS-Basisarchitektur um Grid-Mechanismen erfüllt werden können.

Wie bereits angesprochen, können bei der Service-orient. Implementierung von E-Services zwei Aspekte unmittelbar unterschieden werden. Zum einen gehen die virtuellen Aktivitäten eines E-Service auf die Ressourcen seiner Teilnehmer zur Erbringung einer Dienstleistung zurück und werden durch deren Integration implementiert. Zum anderen gehen die virtuellen Abläufe eines E-Service aus dessen Entwurf hervor und müssen explizit erschaffen werden. Entsprechend können auf der Grundlage virtueller Aktivitäten und Abläufe zwei Arten von Komponenten abgeleitet werden, aus denen sich ein Service-orient. E-Service-Anwendungssystem zusammensetzt. Zum einen beinhaltet ein E-Service-Anwendungssystem Komponenten, die den Zugriff und die Integration von Ressourcen realisieren. Diese Komponenten sollen im Folgenden als E-Service-Ressource bezeichnet werden:

Definition 20 (E-Service-Ressource) *Eine E-Service-Ressource ist eine Service-orient. Software-Komponente, die den internen Anteil eines Dienstleistungsprozesses eines VDPN-Clients oder -Providers* repräsentiert *und dessen Zustand über den gesamten Zeitraum einer individuellen Dienstleistungserbringung konstant widerspiegelt. Die E-Service-Ressource kapselt ein operationales Anwendungssystem, das den internen Geschäftsprozess realisiert und dessen automatischen Zugriff ermöglicht. Interaktionen mit VDPN-Koordinatoren werden gemäß den Vorgaben eines gemeinsamen E-Service-Modells durch den zuverlässigen und sicheren Austausch von Web Service-Nachrichten realisiert.*

Zum anderen beinhaltet ein E-Service-Anwendungssystem Komponenten, die die Interaktion zwischen Ressourcen im Sinne eines virt. Dienstleistungsprozesses steuern

und kontrollieren. Diese Komponenten sollen im Folgenden als E-Service Engine bezeichnet werden:

Definition 21 (E-Service Engine) *Eine E-Service Engine ist eine Service-orient. Software-Komponente, die einen vorgegebenen Anteil des durch ein E-Service-Modell spezifizierten Interaktionsprozesses eines VDPN-Koordinators* realisiert *und dessen Zustand über den Zeitraum einer individuellen Dienstleistungserbringung widerspiegelt. Die E-Service Engine setzt die Logik des Interaktionsprozesses programmatisch um und erlaubt deren automatische Ausführung. Dabei initiiert sie in pro-aktiver Weise Interaktionen mit anderen VDPN-Teilnehmer gemäß den Vorgaben des von ihr realisierten Interaktionsprozesses mittels zuverlässigem und sicheren Austausch von Web Service-Nachrichten.*

Einzelne Komponenten sind in einem E-Service-Anwendungssystem durch wechselseitige Interaktion miteinander integriert. Engines bilden hierbei die zentralen Bestandteile: Zum einen steuern sie partielle Interaktionen mit einer Teilmenge von Ressourcen. Zum anderen interagieren sie untereinander zur Komposition partieller Interaktionen zu einem ganzheitlichen virt. Dienstleistungsprozess. Die proaktive Steuerung von Interaktionen durch Engines bildet die Ausführung des E-Service-Anwendungssystems. Diese Ausführung geschieht potenziell massiv parallel durch multiple Engines. Die ganzheitliche Anwendungslogik ergibt sich in deren Zusammenspiel.

Diese Anwendungslogik geht auf den Entwurf des virt. Dienstleistungsprozesses in Form eines plattformunabhäng. Modell zurück. Das PIM spezifiziert die Clients, Provider und Koordinatoren eines VDLP sowie deren partielle Interaktionsprozesse. Dieses Modell wird schon in einer frühen Phase des E-Service-Entwicklungslebenszyklus entworfen und in Hinblick auf eine ESMS-Plattform zu einem Service-orient. plattformspezif. Modell konkretisiert. Im PSM werden die Akteure als Web Services und die Interaktionsprozesse als Kompositions- und Koordinationsprozesse abgebildet. PIM und PSM bilden ein E-Service-Schema, das für verschiedene Gruppierungen von Teilnehmern wiederholt zum Einsatz kommen kann. Bei der Implementierung einer E-Service-Instanz werden aus dem plattformspezif. Modell die Engine- und Ressourcen-Komponenten des E-Service-Anwendungssystems abgeleitet. Die hierzu angestrebte Strategie besteht darin, das PSM in Konfigurationen für eine Web Service Composition Engine zu transformieren. Das heißt, dass für jeden Koordinator und jeden seiner partiellen Kompositionsprozesse ein Kompositionsschema erstellt wird. Jedes Kompositionsschema dient dann zur Konfiguration einer E-Service Engine. Ferner müssen für alle im PSM spezifizierten Clients und Provider E-Service-Ressourcen bereitgestellt werden. Diese müssen die vorgegebenen Koordinationsprozesse umsetzen, indem sie die darin für sie festgelegten WS-Schnittstellen bereitstellen und auf diejenigen ihrer Interaktionspartner in der vorgegebenen Abfolge zugreifen.

Die Komponenten eines E-Service-Anwendungssystems existieren voneinander getrennt auf den unterschiedlichen lokalen Plattformen der VDPN-Teilnehmer. Jeder

Teilnehmer, der in einer E-Service-Instanz eine oder mehrere Rollen als Client, Provider oder Koordinator übernimmt, realisiert dabei auf seiner Plattform die entsprechenden E-Service-Ressourcen und/oder -Engines. Zur Ausführung einer E-Service-Instanz müssen diese Komponenten dauerhaft zueinander in Beziehung gesetzt werden. Dazu ist es notwendig, dass für jede Rolle im E-Service-Schema bei deren Teilnehmer eine Instanz der zur Rolle gehörigen Komponente erstellt wird und diese einen individuellen Web Service-Endpunkt bereitstellt. Daraufhin müssen sich alle Komponenten an diejenigen Web Service-Endpunkte binden, die die Rollen ihrer im E-Service-Schema festgelegten Interaktionspartner darstellen. Damit die Komponenten Identität und Zustand über den gesamten Ablauf der E-Service-Instanz bewahren, müssen sie ihren Web Service-Endpunkt exklusiv beibehalten. Falls ein Teilnehmer an der Ausführung mehrerer E-Service-Instanzen beteiligt ist, muss er für jede E-Service-Instanz auch eine individuelle Instanz seiner Komponenten (sowie der dahinterstehenden physikalischen Ressourcen) erstellen und mit individuellen Web Service-Endpunkten versehen. Web Services sind jedoch ursprünglich als statische Einheiten ohne Zustand ausgelegt worden. Weder die Erzeugung und dauerhafte Referenzierung individueller Instanzen noch die plattformübergreifende Verwaltung solcher Referenzen sind in den Standards der WS-Basisarchitektur vorgesehen oder werden durch COTS Web Service Middleware unterstützt. Zur Ausführung von E-Service-Instanzen sind daher erweiterte Abstraktionen und Mechanismen erforderlich, wie sie in der vorherigen Sektion durch Grid-Techniken eingeführt wurden.

Auf Grund der geschilderten Charakteristik von E-Service-AS soll eine entsprechende Ausführungsplattform nicht auf Basis der Web Service-Basisarchitektur, sondern auf Basis der von OGSA realisierten Erweiterung in Hinblick auf Grid-Techniken entworfen werden. Das Konzept einer OGSA-basierten E-Service-Plattform basiert auf der Realisierung der Komponenten eines E-Service-Anwendungssystemen als Grid Services. Dadurch können die Mechanismen einer Grid-Plattform verwendet werden, um Instanzen von Komponenten zu erzeugen und über ihre Lebenszeit zu verwalten. Während dieser Zeit können die Instanzen nicht nur eindeutig referenziert werden, sondern es kann auch bei Bedarf deren Implementierung oder deren Bindung gewechselt werden, ohne dass die Referenz ihre Gültigkeit verliert. Dies ist im E-Service-Kontext besonders nützlich, um ggf. sichere oder zuverlässige Kommunikationsprotokolle für die Kopplung einzelner Komponenten des E-Service-Anwendungssystems verwenden zu können. Des Weiteren ermöglichen Grid-basierte Komponenten die Bereitstellung von Laufzeitinformationen durch Service Data-Elemente und Benachrichtigungsmechanismen, was ganz allgemein die Verwaltung von Komponenten erleichtert und speziell einen Ansatzpunkt für Provision-Mechanismen, z. B. zur Protokollierung, Leistungserfassung oder Abrechnung von E-Services bietet.

Abbildung 5.8 zeigt die Architektur eines Grid-basierten E-Service-Anwendungssystems. Hierin unterhalten alle VDPN-Teilnehmer eine eigene Grid-Plattform auf Basis ihrer lokalen System- oder Middleware-Plattformen. Auf dieser Basis stellen

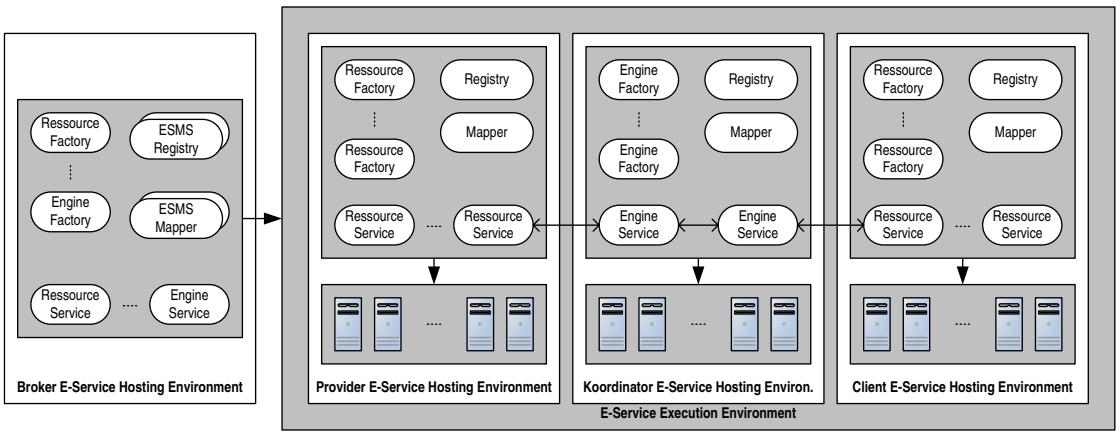

Abbildung 5.8.: Ressourcen und Engines im E-Service Grid

Teilnehmer für die ihren Rollen im E-Service entsprechenden Komponenten Grid Service Factories zur Verfügung. Alle Factories und Instanzen der lokalen Grid Service-Komponenten sind in einem Repository vermerkt. Mapper bilden die GSH von Instanzen auf Implementierungen ab, die lokale Ressourcen oder Engines kapseln. Konkret realisieren Clients und Provider Grid Services für E-Service-Ressourcen. Koordinatoren realisieren E-Service Engine Grid Services.

Ein wesentlicher Aspekt des hier entworfenen E-Service Grid ist die Aggregation von E-Service-Instanzen. Die offensichtliche Lösung besteht in einer virtuellen Grid-Plattform, die typischerweise von einem Netzwerk-Broker betrieben wird. Im Kontext der virtuellen Plattform ergibt sich eine ganzheitliche Perspektive auf E-Services mit allen dazugehörigen Komponenten. Für jede E-Service-Komponente jeder Rolle existiert hier eine Grid Service Factory. Factories, die inhärent eine Kennzeichnung der von einer Komponente realisierten WS-Schnittstelle beinhalten (obligatorisches Service Data Element der OGSA `GridService`-Schnittstelle) müssen hierbei zusätzlich mit einer Rolle im E-Service-Schema in Beziehung gesetzt werden, da die gleiche Schnittstelle für verschiedene Rollen verwendet werden kann, wobei sich die entsprechenden Komponenten dann in ihrer Identität unterscheiden müssen. Die Kennzeichnung von Rollen kann in der virtuellen Plattform z. B. durch spezifische Repositories oder Service Data Elemente der Factories erreicht werden. Am Anfang der Aggregation werden die Factories genutzt, um Grid Service-Instanzen der Komponenten zu erzeugen. Hierbei wird die Erzeugung an die lokale Plattform desjenigen Teilnehmers weitergeleitet, der die spezifische Rolle in der spezifischen Instanz des E-Service einnehmen soll. Dazu muss zuvor die Zuteilung von Rollen zu Teilnehmern durch den Broker erfolgen, der daraufhin die entsprechenden Factories erstellt. Ist eine Komponente einmal durch die Factory erzeugt worden, behält sie ihre Identität in Form des GSH für die gesamte Dauer der E-Service-Erbringung bei. Sie wird dabei im Kontext der virtuellen Grid-Plattform des Brokers registriert.

Anfragen von Komponenten, die eine Kopplung mit der Komponente im Rahmen derselben E-Service-Instanz eingehen wollen, werden dann an die entsprechende Komponenteninstanz verwiesen. Dazu muss jede registrierte Komponenteninstanz durch ihre Rolle, Schnittstelle und E-Service-Instanz eindeutig gekennzeichnet sein. Dies kann wiederum durch Definition spezieller Repositories und/oder Service Data-Elemente erreicht werden. Z. B. könnten spezifische Repositories für jede E-Service-Instanz verwendet werden. Insgesamt kann nach dem Muster des dargestellten Entwurfs eine verteilte Ausführungsplattform für Service-orient. E-Services auf sehr natürliche Weise realisiert werden.

5.2.2.2. FRESCO-Plattform

Im Folgenden soll der Entwurf einer ESMS-Ausführungsumgebung am konkreten Beispiel der FRESCO-Plattform betrachtet werden, die zur Implementierung und Ausführung von FRESCO E-Service-Instanzen dient. Den Ausgangspunkt der Betrachtung bildet die in Sektion 5.2.1.2 beschriebene FRESCO-Methodik zur modellgetriebenen Entwicklung von E-Services und die dabei eingeführte Hierarchie der E-Service-Modelle (vgl. Abb. 5.3). Plattformunabhängige und -abhängige E-Service=Modelle werden hierin durch das FRESCO E-Service-Metamodell bestimmt und bilden E-Service-Schemata. Hierin werden entscheidende Aspekte der E-Service-Aggregations- und -Ausführungssemantik festgelegt, die durch eine E-Service-Plattform umgesetzt werden müssen. Zudem wird durch die Entwicklungsmethodik auch die Strategie zur Entwicklung von Komponenten eines E-Service-Anwendungssystems vorgegeben. Im Einklang mit der Realisierung des physikalischen E-Service-Entwurfs auf Basis eines BPEL-Generators erfolgt der Entwurf von E-Service Engines auf Basis einer BPEL-Laufzeitumgebung. Gleichzeitig soll das in der vorangegangenen Sektion entworfene generelle Konzept eines E-Service Grid realisiert werden. Dazu sind die diskutierten Optionen zur Grid-basierten Realisierung der E-Service-Aggregation mit den Vorgaben des FSM abzugleichen und als konkreter Mechanismus umzusetzen.

Im Folgenden wird zunächst der ganzheitliche Entwurf der FRESCO E-Service-Plattformarchitektur betrachtet. Anschließend wird ein Konzept zur E-Service-Aggregation im Einklang mit dem FSM entwickelt. Schließlich wird die Umsetzung der Komponenten von FRESCO E-Service-Anwendungssystemen fokussiert. Dies betrifft vor allem ein Konzept zur Integration von BPEL Engines mit der OGSA-basierten FRESCO-Plattform und deren Aggregationsmechanismen.

Integrierte E-Service-Implementierung und Ausführung Die FRESCO E-Service-Plattform soll eine integrierte Umgebung zur Implementierung und Ausführung von E-Service-Instanzen realisieren. Dabei soll als Grundlage eine horizontalen Basisinfrastruktur von VDPN-Teilnehmern in Form von OGSA-Mechanismen vorausgesetzt werden. Auf dieser Basis erfolgt der Entwurf der Plattformarchitektur durch Konzeption eines *generischen Komponentenmodells* sowie jeweils eines *Architekturmodells*,

Entwicklungsmodells und *Ausführungsmodells* für E-Service-Anwendungssysteme [ZBL03, ZLP04, ZL04a].

Das generische Komponentenmodell dient als Grundlage für den aufbauenden Entwurf spezifischer Komponenten. Dies sind zum einen *Anwendungskomponenten* als Bestandteile von E-Service-Anwendungssystemen und *Plattformkomponenten*, die Mechanismen der FRESCO-Plattform zur Entwicklung und Ausführung von E-Service-Anwendungssystemen verfügbar machen. Die wichtigste Festlegung ist diejenige, dass alle Komponenten Spezialisierungen von OGSA Grid Services darstellen sollen, d. h. dass sie die OGSA `GridService` Schnittstelle implementieren müssen. Komponenten können dann mit den Mechanismen einer OGSA-Plattform erzeugt und verwaltet werden. Ihre organisationsübergreifende Interoperabilität ist durch die zugrunde liegenden Web Service-Techniken gewährleistet. Anwendungs- oder plattformspezifische Schnittstellen können als Erweiterung der Basisschnittstelle realisiert werden. Zur Interaktion mit Plattformkomponenten werden zudem die Formate von E-Service-Metadaten als XML-Schema-Typen definiert.[19]

Auf dem generischen Komponentenmodell setzt das Architekturmodell für E-Service-Anwendungssysteme auf. Dieses hat die Aufgabe, die Software-Komponenten eines FRESCO E-Service-Anwendungssystems und deren Beziehungen zueinander festzulegen. Art und Beziehung von Anwendungskomponenten leiten sich hier aus dem FSM ab.[20] Anwendungskomponenten spiegeln die Rechte und Pflichten wider, die durch eine Rolle im E-Service-Schema repräsentiert werden. Hierbei werden Client/Provider- und Koordinator-Rollen unterschieden, die, wie in der vorangegangenen Sektion erläutert, durch E-Service-Ressourcen und -Engines ausgefüllt werden.

Eine *FRESCO E-Service-Ressource* stellt eine Anwendungskomponente dar, die sich auf einen `eService Client` oder `eService Provider` innerhalb des `eService Interaction Context` einer `eService Shell` bezieht. Die Rolle ist an verschiedenen Interaktionen (`eService Client Feedback` oder `eService Resource Directive`) im Zuge von Interaktionsprozessen von Capabilities beteiligt. Dabei ist es ihre Pflicht, die Menge der Kontaktpunkte (`eService Demand Contact` oder `eService Asset Access`) bereitzustellen, auf die hierbei zugegriffen wird. Die E-Service-Ressource bezieht sich auf die Konkretisierung dieser Elemente im Kontext eines Koordinationsmodells (`Orchestrated Multi-Party Conversation`). Hierin bildet eine Client/Provider-Rolle den Partner einer WS-Konversation (`Conversation Party`), der für Interaktionen (`Orchestrator Directive`) von Protokollen (`Web Service Composition Protocol`) Kontaktpunkte (`Conversation Party Listener`) bereitstellt. Die Kontaktpunkte werden dabei durch die Operationen

[19] Es sei bemerkt, dass die Festlegung des gleichen Komponentenmodells für E-Service-Anwendungssysteme und Plattformmechanismen dazu führt, dass diese miteinander in Beziehung treten können und dadurch grundsätzlich das technische Potenzial zur Realisierung reflektiver und selbstmodifizierender Anwendungssysteme entsteht, d. h. dass Letztere durch Zugriff auf Plattformmechanismen Einfluss auf ihre eigene Entwicklung und Ausführung nehmen können.

[20] Vgl. FSM-UML-Modell, S. 264.

eines Web Service `PortType` spezifiziert, den die E-Service-Ressource implementieren muss. Zudem muss die E-Service-Ressource die spezifizierten Protokolle implementieren. Das bedeutet zum einen, dass sie Nachrichten an dem von ihr bereitgestellten Endpunkt in der spezifizierten Abfolge annehmen muss. Zum anderen ist jedes Protokoll mit genau einem Interaktionspartner verbunden (`Conversation Orchestrator`), mit dessen Kontaktpunkten (`Conversation Orchestrator Receptor`) die E-Service-Ressource in der spezifizierten Abfolge Interaktionen (`Party Statement`) durchführen muss. Die Kontaktpunkte des Orchestrators gliedern sich in diejenigen einer Capability ein (`eService Capability Service`). Eine Capability fasst nämlich die Kontaktpunkte aller Orchestrators ihrer Interaktionsprozesse zusammen. Sie werden in ihrer Gesamtheit von einem Provider (`eService Capability Provider`) erbracht, der die Rollen aller an Interaktionsprozessen der Capability beteiligten Orchestrators in sich vereint. Die zusammengefassten Kontaktpunkte einer Capability werden als Operationen eines einzelnen Web Service `PortType` spezifiziert. Die E-Service-Ressource muss sich zur Laufzeit mithilfe des in der nächsten Sektion beschriebenen Aggregationsmechanismus an einen entsprechenden Endpunkt binden. Abbildung 5.9 zeigt einige E-Service-Ressourcen und veranschaulicht verschiedene Formen ihrer Interaktion im Zuge von VDLP.

Während die Client- und Provider-Rollen eines Foundational Research on Service Composition E-Service sich in Bezug auf ihre Eingliederung in den VDLP entsprechen und beide auf E-Service-Ressourcen abgebildet werden können, unterscheidet sich die Koordinatorrolle erheblich. Client und Provider stellen eine passive Schnittstelle zur Interaktion mit ihren vorhandenen internen Prozessen bereit, die einem Konversationsprotokoll des E-Service genügt. Ein FRESCO-Koordinator stellt ebenfalls eine Schnittstelle zur Interaktion bereit, diese ist jedoch aktiv und muss auf Basis multipler Kompositionsschemata des E-Service erst realisiert werden. Grundsätzlich muss ein FRESCO-Koordinator eine Capability realisieren, die verschiedene Interaktionsprozesse zur aktiven Steuerung des VDLP beinhaltet. Er vereinigt dabei die Rolle eines `Capability Providers`, der die Capability als Ganzes realisiert mit den verschiedenen `Capability Interaction Coordinator`-Rollen, die jeweils einen der Interaktionsprozesse der Capability realisieren. Die Realisierung dieser kombinierten Aufgabe erfolgt durch eine spezielle Art von Anwendungskomponente: der E-Service Engine. Konkret ist eine *FRESCO E-Service Engine* eine Anwendungskomponente, die sich auf die verschiedenen `eService Interaction Coordinators` von `eService Capability Interaction Flows` einer `eService Capability` und deren `eService Provider` bezieht. `eService Interaction Coordinators` sind grundsätzlich an allen Interaktionen eines `Capability Interaction Flow` beteiligt. Sie greifen auf die Kontaktpunkte (`eService Interaction Contact`) verschiedener Clients, Provider oder anderer Koordinatoren zu. Sie selbst stellen zum Zugriff auf den Interaktionsprozess wiederum Kontaktpunkte (`eService Interaction Service`) bereit. Deren Zugriff durch die Teilnehmer erfolgt jedoch über die Kontaktpunkte der Capability `eService Capability Service`, die durch den `eService Capability Provider` für alle Interaktionsprozesse zusammengefasst und

bereitgestellt werden. Die Interaktion zwischen Koordinatoren bildet einen Spezialfall, da dies entscheidend den Verlauf des VDLP bestimmt. Ansonsten ist die Interaktion mit einem anderen Koordinator bzw. `eService Capability Provider` identisch zu der Interaktion mit einem Client/Provider.

Die E-Service Engine bezieht sich auf die Konkretisierung einzelner Interaktionsprozesse im Kontext eines Web Service-Kompositionsmodells. Hierin bilden die `eService Interaction Coordinators` WS-Aggregatoren (`Web Service Aggregators`) eines zusammengesetzten Web Service (Web Service Composition), die jeweils den Orchestrierungsprozess (`Web Service Orchestration Process`) mit verschiedenen WS-Providern (`Web Service Provider`) steuern und dabei auf deren WS-Operationen (`Web Service Component Operation`) zugreifen und auch selbst WS-Operationen (`Web Service Composition Operation`) bereitstellen. E-Service Clients, -Provider sowie -Koordinatoren assoziierter Capabilities bilden auf dieser Ebene allesamt Web Service-Provider. Die Operationen der verschiedenen Provider und des Aggregators sind jeweils als Web Service `PortType` spezifiziert. Eine E-Service Engine muss die WSDL `PortTypes` aller Aggregatoren ihrer Interaktionsprozesse gemeinsam implementieren und einen entsprechenden Endpunkt bereitstellen. Zudem muss sie die `PortTypes` der Provider einbeziehen und entsprechende Endpunkte unter Zuhilfenahme eines Aggregationsmechanismus zur Laufzeit binden. Schließlich muss eine E-Service Engine die Orchestrierungsprozesse implementieren. Wie dies geschieht, ist aus allgemeiner Perspektive belanglos. Im Sinne des FSMV wird jedoch die Implementierung als BPEL-basierte WS-Kompositionen vorgeschlagen. Der Entwurf einer entsprechenden E-Service Engine auf Basis einer BPEL-Laufzeitumgebung wird weiter unten beschrieben. Abbildung 5.9 zeigt Beispiele einiger E-Service Engines in ihrer zentralen Position zwischen den E-Service-Ressourcen und deutet ihre Implementierung auf Basis von Kompositionsschemata an.

Das Architekturmoldell beschreibt, wie die Komponenten eines FRESCO E-Service-Anwendungssystems und deren Beziehungen aus dem FRESCO E-Service-Metamodell hergeleitet werden. Aufbauend legt das Entwicklungsmodell fest, wie sich die Implementierung von E-Service-Instanzen als Überführung von E-Service-Schemata in Anwendungskomponenten vollzieht, wie sich dieser Vorgang in den modellgetriebenen Entwicklungsprozess eingliedert und wie dessen Mechanismen als Plattformkomponenten in die FRESCO-Plattform einbezogen werden sollen. Eine wesentliche Anforderung an das Entwicklungsmodell ist Agilität bei der Implementierung von E-Service-Instanzen. Aus diesem Grund wird angestrebt, die Mechanismen der modellgetriebenen Entwicklung auf der Plattformebene einzubeziehen. Das Ziel besteht darin, einen möglichst großen Anteil der Entwicklung von E-Service-Anwendungssystemen auf Basis von automatisch generierten oder transformierter Spezifikationen bzw. Code-Fragmenten zu vollziehen. Um dies zu erreichen wird in FRESCO der Mechanismus zur Generierung von BPEL-Kompositionsschemata[21] als Komponente in die FRESCO-

[21] Siehe Sektion 5.2.1.2, S. 326.

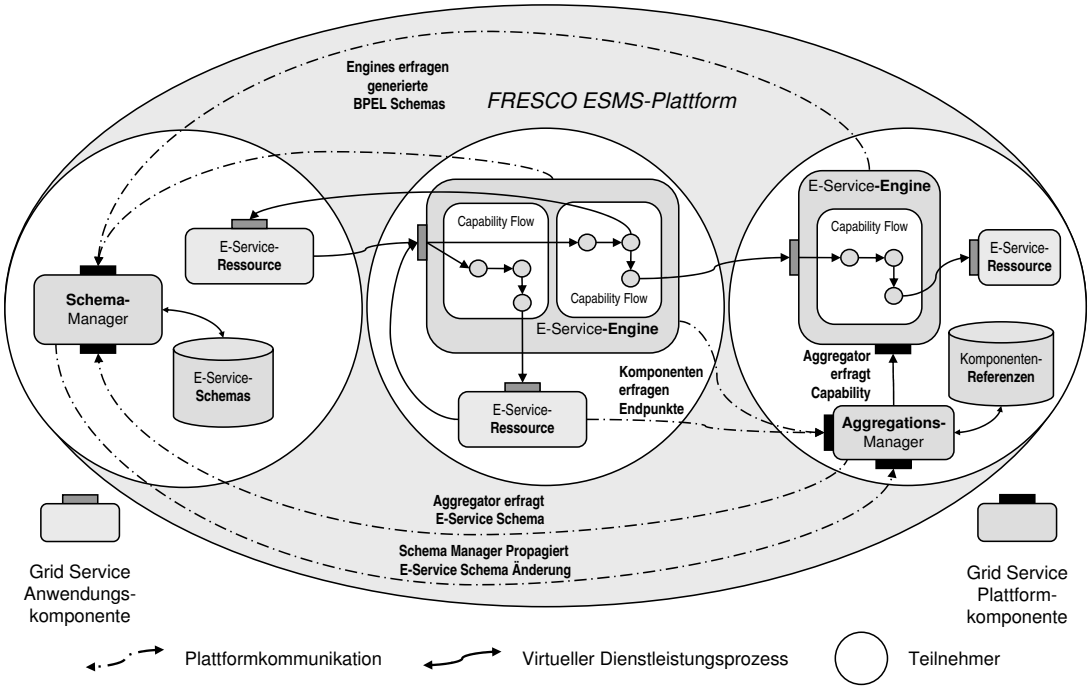

Abbildung 5.9.: Komponenten der FRESCO E-Service-Plattform

Plattform integriert. Diese als *Schema-Manager* bezeichnete Plattformkomponente stellt eine Schnittstelle zur Verfügung, die auf Anweisung das BPEL-Kompositionsschema eines `eService Capability Interaction Flow` generiert. Die Grundlage der Transformation bilden E-Service-PSM in XPDL-Repräsentation, die in einem Repository vorgehalten werden. Eine weitere Funktion des Schema-Managers ist die Änderung von E-Service-PSM. Auf diese Weise können auf der Plattformebene ad hoc-Änderungen am E-Service-Schema vorgenommen und sofort als E-Service-Instanz implementiert werden. Diese Implementierung geht derart vonstatten, dass BPEL-Kompositionsschemata zur Konfiguration einer BPEL-Laufzeitumgebung dienen. Diese bildet den Kern einer E-Service Engine, die in der übernächsten Sektion beschrieben wird. Die Implementierung von E-Service Engines basiert neben der Konfiguration der Laufzeitumgebung auf deren Kapselung als Anwendungskomponente der FRESCO-Plattform. Beides wird von einem zur E-Service Engine gehörigen Assistenten automatisch durchgeführt. Abbildung 5.9 deutet diesen Vorgang im Zusammenspiel von E-Service Engines und Schema-Manager an. Die Kapselung der BPEL-Laufzeitumgebung sowie auch die Entwicklung von E-Service-Ressourcen als Anwendungskomponente im Sinne des generischen Komponentenmodells basiert auf einem Software-Rahmenwerk. Die Entwicklung von E-Service-Ressourcen erfolgt damit in manueller Form auf den lokalen System- und Middleware-Plattformen von Clients und Providern. Um die Entwicklungszeit hier weiter zu reduzieren, sollte

versucht werden, im Rahmen des strategischen Netzwerks Standards für die Schnittstellen von Ressourcen zu entwickeln und diese beim Entwurf von E-Service-PSM zu verwenden. Im günstigsten Fall sind die entsprechenden Anwendungskomponenten dann zur Zeit der Implementierung von E-Service-Instanzen schon vorhanden.

Der verbleibende Aspekt der Ausführung von E-Service-Instanzen auf der FRESCO-Plattform wird durch das Ausführungsmodell für E-Service-Anwendungssysteme beschrieben. Den Ausgangspunkt bildet hier eine Menge von Anwendungskomponenten, die aus der Implementierung einer E-Service-Instanz hervorgegangen sind. Diese müssen grundsätzlich von den konkreten VDPN-Teilnehmern auf Basis ihrer lokalen System- oder Middleware-Plattformen zur Verfügung gestellt werden. Dazu müssen die Komponenten dort erst installiert und konfiguriert werden (Deployment), bevor dann Instanzen erzeugt werden können und dadurch eine Allokation lokaler Ressourcen stattfindet. Beides sollte bei Bedarf während der Laufzeit geschehen, um nicht unnötig Ressourcen zu binden, bevor diese benötigt werden. Für die Verwaltung der Komponenteninstanzen von E-Services stellt die Plattform einen Aggregationsmechanismus zur Verfügung, der als Plattformkomponente nutzbar ist. Die Funktionsweise dieses *Aggregation-Managers* wird in der folgenden Sektion noch genauer beschrieben. Im Gesamtzusammenhang der FRESCO-Plattform beginnt die Ausführung einer E-Service-Instanz mit der Analyse des E-Service-Schemas durch den Aggregation-Manager, der dazu das XPDL-basierte E-Service-PSM vom Schema-Manager anfragt. Dem Aggregation-Manager muss dann mitgeteilt werden, welchem VDPN-Teilnehmer welche Rollen zugewiesen werden und dieser veranlasst dann das rechtzeitige Deployment zugehöriger Anwendungskomponenten und die Allokation von Ressourcen durch Erzeugung von Komponenteninstanzen. Der eigentliche Ablauf beginnt mit dem Deployment und der Erzeugung einer Instanz der ersten E-Service Engine, die dann das von ihr implementierte `eService Initial Capability Interaction Pattern` ausführt. Diese steuert dann die Interaktionen assoziierter E-Service-Ressourcen und stößt ggf. weitere E-Service Engines an, die die Ausführung weiterführen. Zur Laufzeit des E-Service-AS besteht die wesentliche Aufgabe des Aggregation-Managers in der Bereitstellung von Bindungsinformationen für die Instanzen von Anwendungskomponenten. Letztere stellen vor der Durchführung einer Interaktion eine Anfrage an den Aggregation-Manager, um den Endpunkt der Komponente des Interaktionspartners zu erfragen und sich dynamisch an diesen zu binden. Die Ausführung endet, wenn alle E-Service Engines ihre Interaktionsprozesse beendet haben. Dabei zerstört der Aggregation-Manager die Komponenten-Instanzen und gibt die lokalen Ressourcen wieder frei.

Dynamische Aggregation von E-Service-Ressourcen Die Aggregation von E-Services dient dem Zweck, die Anwendungskomponenten eines E-Service-Anwendungssystems mit den Teilnehmern der hierdurch implementierten E-Service-Instanz zu assoziieren, im Verlauf der Ausführung die Erzeugung von Instanzen der Anwen-

dungskomponenten zu steuern und die dynamische Bindung zwischen Komponenteninstanzen zu unterstützen.

Aus der Perspektive des konzeptionellen VDL-Modells plant ein Teilnehmer des Dienstleistungsnetzwerks in der Rolle eines FRESCO-Brokers die virtuelle Produktion einer Dienstleistung durch Abstraktionen von (a) dem virt. Dienstleistungsprozess als Interaktionsmuster zwischen lokalen Anteilen des physikalischen Dienstleistungsprozesses (Demands, Assets und Capabilities) und (b) den für die Ausführung lokaler Prozesse zuständigen Teilnehmern (Client, Provider und Koordinator-Rollen). Während der Dienstleistungsproduktion werden die geplanten Rollen durch konkrete Teilnehmer übernommen. Diese müssen die für ihre Rolle(n) geplante(n) lokalen Prozesse durchführen und ihren Kooperationspartnern durch die geplanten Interaktionen zugänglich machen. Zudem müssen sie die für ihre(n) lokalen Prozess(e) geplanten Interaktionen mit Kooperationspartnern vollziehen. Bevor sie dies jedoch tun können, müssen sie wissen, welche Teilnehmer die Prozesse durchführen und deren lokale Prozesse müssen zum Zeitpunkt der Interaktion gestartet sein. Der Broker muss daher dafür Sorge tragen, dass die VDPN-Teilnehmer ihre lokalen Prozesse rechtzeitig starten. Zudem muss er den Teilnehmern ihre Interaktionspartner mitteilen. E-Service-Aggregation bildet diese Konzepte auf die Plattformebene ab.

Die Aggregation auf Plattformebene basiert auf der Abbildung von Demands, Assets und Capabilities und deren Clients, Provider und Koordinatoren im E-Service-Modell auf Ressourcen und Teilnehmer der ESMS-Ausführungsumgebung. Demands und Assets basieren auf internen Geschäftsprozessen von Clients und Providern. Diese werden in der Ausführungsumgebung von Teilnehmern des VDPN in Form von E-Service-Ressourcen-Komponenten repräsentiert und zugänglich gemacht. Capabilities spezifizieren externe Interaktionsprozesse, die durch einen Koordinator zu erbringen sind. Sie werden in der Ausführungsumgebung von Teilnehmern des VDPN in Form von E-Service Engine-Komponenten realisiert und ebenfalls zugänglich gemacht. Unabhängig von der Art der lokalen Prozesse bieten alle Anwendungskomponenten einen Zugang über Prozessschnittstellen nach Vorgaben des generischen Komponentenmodells der Plattform. Dies ist die Basis für die wechselseitige Kopplung von Anwendungskomponenten zu einem E-Service-Anwendungssystem. Unter diesem Gesichtspunkt ist es ausreichend, Demands, Assets und Capabilities allesamt als einheitliche Komponenten zu betrachten, die auf Basis abstrakter *Ressourcen* realisiert werden. Entsprechend können Client, Provider und Koordinator allesamt als einheitliche Teilnehmer betrachtet werden, die Ressourcen für die Erzeugung von Komponenteninstanzen bereitstellen. Der Aggregationsmechanismus analysiert daher das plattformspezif. Modell eines E-Service auf die darin spezifizierten `eService Clients`, `eService Provider` und `eService Capability Provider` sowie deren in WSDL `PortTypes` zusammengefasste Kontaktpunkte. Er registriert alle diese E-Service-Rollen als *Teilnehmerrollen* und deren Kontaktpunkte als *Ressourcenklassen* der Plattform. Jede Teilnehmerrolle geht mit der Verpflichtung einher, während der E-Service-Ausführung auf Anfrage Ressourcen bereitzustellen und als Komponenteninstanz mit

dem WSDL `PortType` der Ressourcenklasse verfügbar zu machen. Es ist die Aufgabe des Aggregationsmechanismus dafür Sorge zu tragen, dass die Teilnehmer eines ESMS die mit ihren Rollen verbundenen Komponenteninstanzen rechtzeitig bereitstellen und dass diese Komponenteninstanzen im Kontext verschiedener E-Service-Schemata und -Instanzen richtig eingeordnet werden.

Der Vorgang der Aggregation beginnt, sobald das Schema einer zu implementierenden E-Service-Instanz feststeht. Dies ist entweder schon während der Identifikation potenzieller Teilnehmer auf Stufe 3 des FSMV der Fall, oder aber im Anschluss an die Verhandlungen der potenziellen Teilnehmer und der daraus ggf. resultierenden Änderung des Schemas zu Beginn von Stufe 4. An diesem Punkt erfolgt eine Analyse des Schemas in Bezug auf die zur Implementierung einer E-Service-Instanz benötigten Ressourcenklassen und Teilnehmerrollen. Ferner wird für den E-Service eine Instanz deklariert und mit einem eindeutigen Bezeichner gekennzeichnet. Hierdurch werden multiple E-Service-Instanzen unterschieden, die dann existieren können, wenn das E-Service-Schema nicht verändert wurde und schon für die Implementierung anderer VDPN ohne Veränderung zum Einsatz gekommen ist. Die Verwaltung der neuen E-Service-Instanz beinhaltet drei wesentliche Aspekte:

- *Assoziation* – Zuordnung von Teilnehmerrollen zu VDPN-Teilnehmern und Registrierung von deren lokalen Ausführungsumgebungen als Teil der organisationsübergreifenden ESMS-Plattform.
- *Deployment* – Installation und Konfiguration von Anwendungskomponenten in der lokalen Ausführungsumgebung von VDPN-Teilnehmern entsprechend den Ressourcenklassen ihrer Teilnehmerrollen.
- *Allokation* – Anweisung zur Bereitstellung lokaler Ressourcen und Erzeugung von Komponenteninstanzen, deren Endpunkte registriert und anderen Komponenten zur dynamischen Bindung mitgeteilt werden.

Die Behandlung dieser Aspekte bei der Ausführung von E-Service-Instanzen kann auf verschiedenartige Weise erfolgen. Grundlegend sind verschiedene *Assoziations-, Deployment- und Allokationsmodelle* möglich. Aufbauend können *Aggregationsstrategien* definiert werden, die spezifische Modelle kombinieren und zu verschiedenen Zeitpunkten der Ausführung zum Einsatz bringen.

Zur Konfiguration einer E-Service-Instanz müssen die Teilnehmerrollen mit VDPN-Teilnehmern assoziiert werden, die während der VDPN-Implementierung identifiziert wurden. Assoziationsmodelle können hierzu eine aktive oder passive Vorgehensweise verfolgen. Bei der aktiven Assoziation nutzt der Aggregationsmechanismus Repositories des strategischen Netzwerks (z. B. interne Partner-DBs), in denen Informationen zu den Netzwerkunternehmen vorgehalten werden. Im Service-orient. Kontext könnten hier z. B. UDDI-Broker-Mechanismen genutzt werden. Bei passiver Assoziation

wird die Zuordnung nicht vom Aggregationsmechanismus selbst, sondern einer externen Instanz gesteuert. Dies kann wiederum ein automatischer Mechanismus sein, der eine spezifische Schnittstelle des Aggregationsmechanismus nutzt. Dies kann aber auch manuell über ein interaktives GUI erfolgen. In beiden Fällen wird die externe Instanz benachrichtigt, falls die Aggregationsstrategie die Assoziation einer spezifischen Rolle erfordert. Der entsprechende Aggregationsmechanismus wartet dann solange ab, bis die Assoziation durch die externe Instanz erfolgt ist und setzt die Strategie dann fort.

Die Assoziation von Teilnehmern mit Teilnehmerrollen setzt eine statische Definition von Rollen auf Basis von E-Service-Schemata voraus, d. h. alle Rollen eines E-Service müssen zum Zeitpunkt des Entwurfs explizit festgelegt und benannt werden. In manchen Fällen sind die Rollen aber vom dynamischen Verlauf des DLP abhängig und zur Zeit des Entwurfs noch nicht bekannt. Z. B. könnte der DLP einer Transportdienstleistung verschiedene Teilstrecken beinhalten, die zum Zeitpunkt der Produktionsplanung noch nicht festgelegt werden sollen. Die Teilstrecken werden stattdessen am Anfang der Dienstleistungsproduktion auf Basis von aktuellen Bedingungen ermittelt. Erst dann ist die Anzahl unterschiedlicher Rollen von Frachtführern der einzelnen Abschnitte bekannt. In derartigen Situationen wird eine dynamische Definition und Assoziation von Rollen benötigt. In FRESCO wird die Assoziation von Rollen daher auf Basis von Korrelation mit spezifischen Nachrichteninhalten vorgenommen. Hierbei wird bei der Definition von Rollen im E-Service-Schema ein `fresco.CorrelationSet` definiert, das die Datentypen von Nachrichteninhalten enthält, die zur Korrelation verwendet werden sollen. Generell wird immer der Bezeichner der E-Service-Instanz zur Korrelation verwendet. Daneben können andere Information hinzugefügt werden. Z. B. könnte im Transportbeispiel eine Rolle definiert werden, die mit der laufenden Nummer der Teilstrecke korreliert. Falls dann im Verlauf der Transportdienstleistung a eine Teilstrecke x ermittelt wird, erfolgt die Zuordnung eines Teilnehmers als Frachtführer der Transportdienstleistung a auf Teilstrecke x.

Bevor ein Teilnehmer Komponenteninstanzen einer Ressourcenklasse zur Interaktion mit seinen lokalen Prozessen erzeugen und bereitstellen kann, müssen die Implementierungen von Anwendungskomponenten der E-Service-Instanz zunächst in dessen lokaler Ausführungsumgebung installiert und konfiguriert werden. Dieser Vorgang muss in einer Aggregationsstrategie berücksichtigt werden. Entsprechende Deploymentmodelle können wiederum nach aktiver oder passiver und automatischer oder manueller Ausführung unterschieden werden. Aktive Varianten sind zwangsläufig auch automatisch. Solche Modelle sind auf Grund der Komplexität des Vorgangs nur schwer zu realisieren. In FRESCO wird jedoch eine passive semi-automatische Variante des Deployment für E-Service Engine-Komponenten unterstützt, deren nähere Betrachtung in der nächsten Sektion erfolgt. Für E-Service-Ressourcen-Komponenten ist das Deployment generell eine individuelle Aufgabe, auf die nur passiv und manuell Einfluss genommen werden kann.

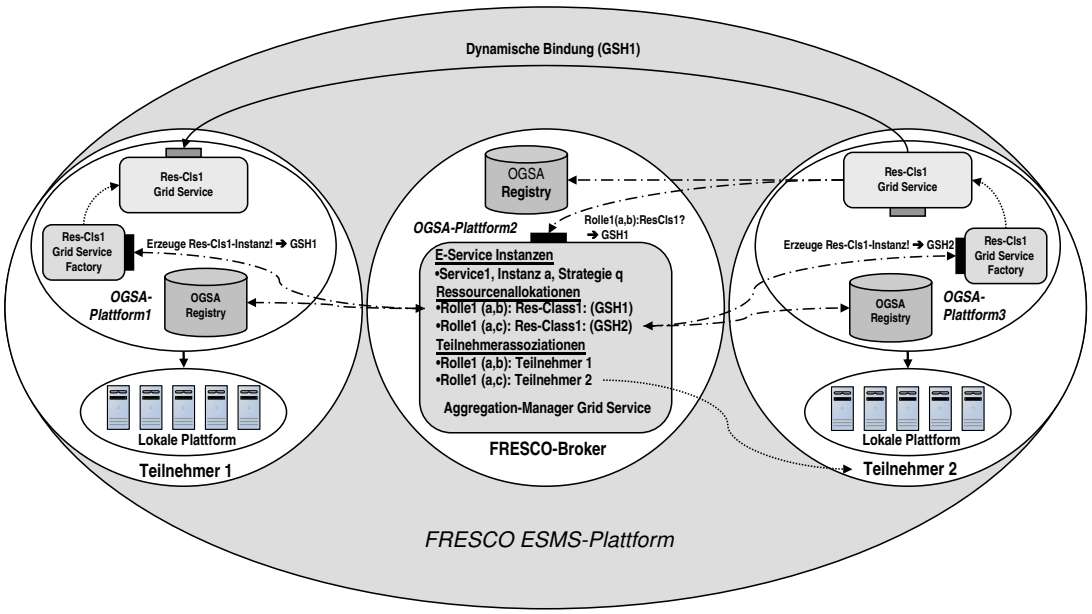

Abbildung 5.10.: FRESCO Aggregation-Manager und Grid-Architektur

Der letzte und entscheidende Schritt zur Ausführung einer E-Service-Instanz besteht darin, Ressourcen von Teilnehmern anzufordern und zu deren Zugriff Komponenteninstanzen zu erzeugen. Für diesen Aspekt ist ein aktives, automatisches Allokationsmodell am besten geeignet, da hierdurch Unterbrechungen vermieden werden, die bei einer dynamischen Bindung von Komponenten den Interaktionsprozess verzögern würden. Eine manuelle, passive Allokation kann trotzdem in manchen Fällen notwendig werden, wenn die Allokation von Ressourcen mit zu komplexen oder schlecht automatisierbaren lokalen Prozessen einhergeht. Für ein automatisches Allokationsmodell können im Kontext des Grid-basierten Komponentenmodells OGSA-Mechanismen herangezogen werden. Hierzu werden in FRESCO für Teilnehmer die Referenzen auf spezifische Grid Repositories in Form von GSH registriert. In diesen Repositories müssen die Teilnehmer Factories für die mit ihren Rollen zusammenhängenden Ressourcenklassen registrieren. Wenn nun die Aggregationsstrategie bei einem Teilnehmer die Allokation von Ressourcen einer Ressourcenklasse erfordert, so wird im automatisierten Allokationsmodell die entsprechende Factory genutzt. Diese ist vom Teilnehmer derart zu implementieren, dass die notwendigen Schritte zur Allokation der lokalen Ressourcen eingeleitet werden. Ferner muss eine Komponenteninstanz erzeugt und deren GSH registriert werden. Diese Registrierung erfolgt jedoch auf globaler Ebene nicht in einem einfachen OGSA Repository, sondern bei der spezifischen Grid Service-Komponente des Aggregation-Managers. Diese Komponente bietet eine erweiterte Schnittstelle an, die spezifische Operationen zur Anfrage der GSH für korrelierte Rollen bereitstellt. Die Funktion könnte zwar auch durch einfache

OGSA Repositories mit erweiterten Service Data-Elementen erreicht werden, dies würde aber die Anfrage nach GSH für Komponenten verkomplizieren. Abb. 5.10 zeigt die resultierende Architektur des FRESCO E-Service Grid.

Die Wahl verschiedener Assoziations-, Deployment- und Allokationsmodelle zusammen mit spezifischen Zeitpunkten zu deren Anwendung im Kontext der Ausführung von E-Service-Instanzen bildet eine Aggregationsstrategie. Der Grund für die Existenz verschiedener Strategien besteht in verschiedenen Möglichkeiten zur Optimierung eines E-Service-Anwendungssystems in Bezug auf konkurrierende Qualitätsmerkmale. Z. B. könnten noch vor dem Start der E-Service-Instanz Assoziation, Deployment und Allokation so weit wie zu diesem Zeitpunkt bekannt komplett durchgeführt werden. Auf diese Weise muss zur Ausführung im besten Fall gar keine Aggregation mehr durchgeführt werden und die damit zusammenhängenden potenziellen Verzögerungen entfallen. Der Nachteil ist hier, dass lokale Ressourcen unnötig und zum Teil untragbar lange blockiert werden. Das andere Extrem besteht in der Durchführung aller Aggregationsaufgaben zum spätesten möglichen Zeitpunkt, d. h. Assoziation, Deployment und Allokation finden alle erst dann statt, wenn der GSH einer korrelierten Rolle das erste Mal von einer Komponente angefragt wird. Hierbei kann es jedoch zu Verzögerungen kommen. Zum Teil finden diese extremen Aggregationsstrategien Anwendung. Es sind jedoch auch beliebige Zwischenstufen denkbar. Im Bedarfsfall sollte eine Anpassung an den spezifischen E-Service möglich sein.

Agile Konstruktion BPEL-basierter E-Service Engines Die Funktion eines E-Service-Anwendungssystems besteht in der Steuerung von Interaktionen zwischen Teilnehmern eines virt. Dienstleistungsproduktionsnetzwerks im Zuge eines Dienstleistungsprozesses, um durch diesen VDLP ganz generell die Koordination kooperativer Tätigkeiten bei der Dienstleistungsproduktion durchzusetzen und zudem durch Automatisierung die Produktivität zu steigern. In einem E-Service-Anwendungssystem werden Teilnehmer und Steuereinheiten durch Grid-basierte Software-Komponenten (kurz: Anwendungskomponenten) repräsentiert. E-Service-Ressourcen realisieren Schnittstellen zu internen Geschäftsprozessen von VDPN-Teilnehmern und erlauben die Interaktion mit diesen im Rahmen des E-Service-Anwendungssystems. E-Service Engines realisieren Orchestrierungsprozesse zur Steuerung von Interaktionen zwischen Anwendungskomponenten. Primär sind dies Interaktionen zwischen E-Service-Ressourcen im Rahmen einer Capability des E-Service-Schemas. Daneben wird durch Interaktionen zwischen E-Service Engines die Verknüpfung von Capabilities zur E-Service-Shell des E-Service-Schemas realisiert.

Für die Funktion eines E-Service-Anwendungssystems ist die Art der Implementierung ihrer E-Service Engines grundsätzlich unerheblich. Die Komponenten müssen sich nur an den durch das Komponentenmodell gesetzten Standard Grid-basierter Interaktion halten und die im E-Service-Schema spezifizierten Vorgaben für Interaktionsinhalte, -reihenfolge und -logik befolgen. D. h. E-Service Engines könnten

prinzipiell mittels einer beliebigen Programmiersprache implementiert werden. Hierbei ist jedoch die Anforderung agiler Entwicklung zu beachten. Die Konstruktion der Komponenten sollte daher möglichst mit der modellgetriebenen Methodik der FRESCO E-Service-Entwicklung gekoppelt werden und auf generierten Artefakten beruhen. Dies ist auf Basis des gegebenen Workflow-basierten E-Service-Schemas einfacher durch Generierung von Workflow-Schemata als von Code einer allg. Programmiersprache zu realisieren. Insbesondere bieten sich hier Workflow-basierte Web Service-Kompositionsmodelle an, die explizit zur Implementierung von Interaktionsprozessen auf Basis von Web Services dienen. Konkret wird hierzu in FRESCO das Web Service-Kompositionsmodell von BPEL verwendet. Hierbei werden mit dem weiter vorne beschriebenen Mechanismus[22] für alle `eService Capability Interaction Flows` einer `eService Capability` entsprechende BPEL-Schemata generiert. Diese Schemata dienen dann zur Konfiguration von und Interpretation durch eine BPEL-Laufzeitumgebung. Hierdurch wird der Kern einer E-Service Engine implementiert.

Die beschriebene Implementierungsstrategie erfordert eine Architektur zur Integration BPEL-basierter Web Service-Kompositionen in die OGSA-basierte FRESCO-Plattform. Dies erfordert zum einen die Erweiterung der durch BPEL-Laufzeitumgebungen realisierten Web Services um Schnittstellen und Eigenschaften des FRESCO-Komponentenmodells, die deren Verwaltung und Zugriff durch Anwendungs- und Plattformkomponenten ermöglichen. Hierbei müssen vor allem Unterschiede bei der Referenzierung von Komponenteninstanzen überbrückt werden. Zum anderen muss die BPEL-Laufzeitumgebung die Verhaltensmuster zum Zugriff auf Anwendungs- und Plattformkomponenten befolgen. Dies erfordert insbesondere eine Harmonisierung der unterschiedlichen Aggregationsmodelle.

Die Referenzierung von Web Service-Kompositionen erfolgt in BPEL standardmäßig über einen statischen Endpunkt. Hierüber wird ein spezifischer Conversation Controller der BPEL-Laufzeitumgebung angesprochen. Dieser leitet Anfragen von Clients anhand einer explizit spezifizierten Korrelation mit Teilen der eingehenden Nachricht an die dazu passende Instanz des Kompositionsschemas weiter. Falls diese Instanz nicht existiert, wird sie implizit erzeugt. Auch die Zerstörung von Instanzen erfolgt implizit. In FRESCO erfolgt die Referenzierung von Komponenteninstanzen auf Basis von OGSA durch Auflösung eines GSH, der für jede Komponenteninstanz eindeutig und dauerhaft ist. Alle Komponenteninstanzen und deren GSH werden durch einen Aggregationsmechanismus verwaltet, der diese mit E-Service-Instanzen in Beziehung setzt und in diesem Zusammenhang mittels Factories explizit erzeugt und je nach Aggregationsstrategie explizit oder implizit wieder zerstört. Für eine E-Service Engine, die auf Basis einer Menge von Web Service-Kompositionen implementiert ist, kann ein entsprechendes Verhalten durch Kapselung der BPEL-Laufzeitumgebung mittels eines Adapter (E-Service Engine Adapter) erreicht werden. Dieser Adapter implementiert die Operationen aller WSDL `PortTypes` von Interaktionsprozessen der

[22] Siehe Sektion 5.2.1.2,S. 326.

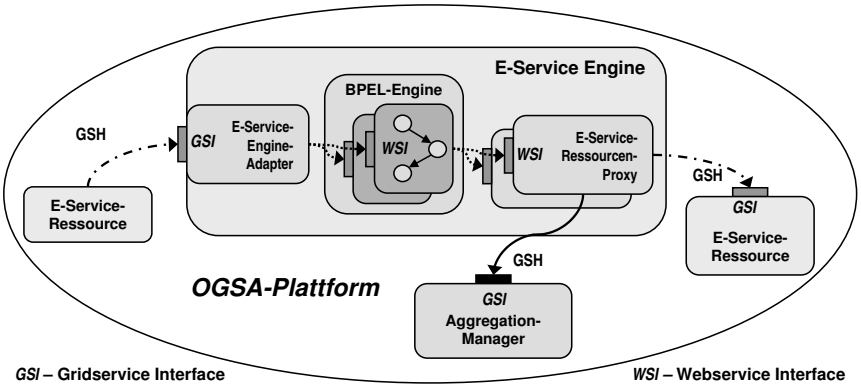

Abbildung 5.11.: Architektur von FRESCO E-Service Engines

Capability als einzelnen Grid Service. Eine Operation o eines Interaktionsprozesses i wird dadurch implementiert, dass Zugriffe auf o an den statischen Endpunkt des Web Service weitergeleitet werden, der durch das zu i korrespondierende BPEL-Schema implementiert wird. Hierbei wird jeder Operation der Bezeichner der E-Service-Instanz als Parameter hinzugefügt und im Kompositionsschema als zusätzliche Korrelation spezifiziert. Wenn der Aggregationsmechanismus eine Komponenteninstanz der E-Service Engine (d. h. des Adapter) erzeugt, teilt sie dieser den Bezeichner der zugehörigen E-Service-Instanz mit. Hierdurch wird sichergestellt, dass die BPEL-Laufzeitumgebung beim ersten Zugriff auf die Komponenteninstanz eine neue Instanz des BPEL-Schemas erzeugt. Abb. 5.11 zeigt eine vereinfachte Darstellung der entsprechenden Architektur von FRESCO E-Service Engines.

Die Aggregation von Web Service-Komponenten erfolgt in vielen BPEL-Laufzeitumgebungen mittels statischer Bindung von Web Service-Providern bei der Erzeugung einer Instanz der Web Service-Komposition. Daneben sieht das Aggregationsmodell die Möglichkeit einer dynamischen Bindung vor. Hierbei werden Endpunkte von Web Service-Providern zur Laufzeit extern z. B. bei einem UDDI-Broker erfragt und dann den Rollen im Kompositionsschema explizit zugeteilt. In FRESCO wird die Aggregation von Komponenteninstanzen eines E-Service-Anwendungssystems grundsätzlich dynamisch im ganzheitlichen Kontext einer E-Service-Instanz durchgeführt. Hierzu wird ein Aggregationsmechanismus verwendet, der als externe Plattformkomponente (Aggregation-Manager) realisiert ist. Obwohl dieses Vorgehen prinzipiell mit der dynamischen Aggregation in BPEL harmoniert, kann von dem Mechanismus kein Gebrauch gemacht werden. Dies liegt daran, dass Web Service-Kompositionen auf Basis von BPEL keine Grid Service-Komponenten mittels GSH adressieren können. Aus diesem Grund werden in E-Service Engines alle Grid Service-Komponenten über einen Stellvertreter (E-Service-Ressourcen-Proxy) angesprochen. Der Proxy erbringt einen Web Service, an dessen Endpunkt sich eine Web Service-Komposition statisch bindet. Hierbei wird für jeden `Client`, `Provider` oder `Capability Provider` des `eService`

`Capability Interaction Flow` einer Capability bzw. für alle von dieser verwendeten Ressourcenklassen ein Proxy vorgehalten. Die Operationen im WSDL `PortType` jeder Ressourcenklasse werden dabei um Parameter erweitert, mit denen aggregationsrelevante Metadaten vom BPEL-Prozess an den Proxy übermittelt werden. Hierzu gehören die Bezeichner der E-Service-Instanz, Ressourcenklasse, Teilnehmerrolle und aller relevanten Korrelationsinformationen. Wenn nun ein BPEL-Prozess eine Interaktion durchführt, so sendet er die entsprechende Nachricht zusammen mit den Metadaten an den zugehörigen Proxy. Dieser führt dann mittels der Metadaten eine Anfrage beim Aggregation-Manager durch und erfragt dort den GSH der entsprechenden Ressource. Nachdem er diesen erhalten hat, leitet er die Nachricht ohne die Metadaten an die Ressource weiter.

Durch die Verwendung von E-Service Engine Adaptern und -Ressourcen-Proxies kann die Realisierung von E-Service Engines der FRESCO-Plattform auf Basis von BPEL-Laufzeitumgebungen erfolgen. Die Implementierung einer spezifischen E-Service Engine umfasst dann (a) die Spezifikation je eines BPEL-Schemas für jeden Interaktionsprozess der zugrunde liegenden Capability, (b) die Konstruktion eines Proxy für jede an einem der Interaktionsprozesse beteiligte Ressourcenklasse sowie (c) die Konstruktion eines Adapter für die durch die Capability implementierte Ressourcenklasse. Die Spezifikation der BPEL-Schemata bildet ohne Zweifel den wichtigsten Anteil der Implementierung. Diese Aufgabe wird von dem weiter vorne entworfenen Generatormechanismus automatisch erledigt. Daneben verursacht jedoch auch die Konstruktion der unter Umständen zahlreichen Proxies und des Adapter einen Aufwand, der der gewünschten Agilität des Entwicklungsprozesses entgegensteht. Zur Beschleunigung der Implementierung beinhaltet die FRESCO-Plattform einen Mechanismus zur automatischen Durchführung von Codegenerierung, Compilierung und Deployment für diese Bestandteile [FW04]. Der Mechanismus ist als Plattformkomponente realisiert (Engine-Manager) und lässt sich in den Aggregationsvorgang eingliedern. Dabei wird er vom Aggregationsmechanismus angesprochen, wenn die Aggregationsstrategie das Deployment der Ressourcenklasse einer Capability erfordert. Dann wird die komplette Konstruktion sowie die Installation und Konfiguration einer E-Service Engine binnen weniger Minuten vollzogen.

5.3. Prototypische ESMS-Implementierung im FRESCO-TK

Nachdem in der letzte Sektion der Entwurf Service-orient. E-Service-Management-Systeme unter generellen Gesichtspunkten und für den spezifischen Fall von FRESCO-E-Services untersucht wurde, sollen die Betrachtungen nun mit der Implementierung von ESMS-Entwürfen fortgesetzt werden. Bei dieser Betrachtung kann und soll nicht auf die Implementierung aller generellen Entwurfsoptionen eingegangen werden. Stattdessen wird die Konstruktion des FRESCO Toolkit (FRESCO-TK) als technisches Rahmenwerk zur Implementierung der Entwürfe für ein FRESCO-ESMS fokussiert.

Dies erlaubt zum einen die detaillierte Betrachtung einer konkreten Lösung und soll zum anderen als Proof-of-Concept einer Service-orient. E-Service-Technologie dienen.

In Sek. 5.3.1 werden als Erstes die Anforderungen an die Bestandteile einer Implementierung zusammengefasst. Wesentliche Aspekte umfassen hier (a) die Umsetzung der geplanten Grid-basierten Basisinfrasruktur als Grundlage für (b) eine E-Service-Plattform mit vertikalen Fachfunktionen der E-Service-Entwicklung und Ausführung unter (c) Berücksichtigung der Rahmenbedingungen virtueller Dienstleistungsproduktion. Das Ergebnis ist ein Katalog funktionaler Anforderungen, die in Form verschiedenartiger Artefakte zu implementieren sind.

Anschließend wird in Sek. 5.3.2 die notwendige Infrastruktur zur Implementierung der Artefakte identifiziert. Dazu werden die einzelnen Artefakte in Bezug auf die für ihre Realisierung notwendigen Infrastrukturbestandteile analysiert und letztere in eine Gesamtarchitektur des FRESCO Toolkit eingeordnet. Für die verschiedenen Kategorien notwendiger Infrastruktur werden dann Basistechniken ausgewählt, die die technische Grundlage des FRESCO-TK bilden sollen. Eine fundamentale Rolle übernimmt dabei die Grid Service Middleware. Aus diesem Grund erfolgt hier eine kurze Einführung der konkreten Konzepte und Techniken.

Auf Basis der Infrastruktur gewählter Basistechniken erfolgt in Sek. 5.3.3 die Implementierung der darauf aufsetzenden E-Service-Plattform-Architektur als horizontaler Infrastrukturbestandteil des FRESCO-ESMS. Dies betrifft zum einen die Spezifikation von Komponenten der Anwendungsarchitektur als Grid Services und deren Implementierung auf Grundlage der Basistechniken. Zum anderen werden die Komponenten der Plattformarchitektur ebenfalls als Grid Services umgesetzt und Datenformate für deren Kommunikation festgelegt.

Aufbauend auf die ESMS-Plattform-Infrastruktur folgt in Sek. 5.3.4 die Implementierung von Plattformkomponenten als ESMS-Fachfunktionen. Dies beginnt mit einem interaktiven grafischen Entwurfswerkzeugs für E-Service-Schemata. Dann werden Plattformkomponenten für die Verwaltung und Transformation von E-Service-Schemata (Schema-Manager), die automatische Konstruktion BPEL-basierter E-Service Engines (Engine-Manager) und die dynamische Aggregation von E-Service-Instanzen (Aggregation-Manager) behandelt.

5.3.1. Implementierungsspezifische Anforderungen

Die Implementierung des FRESCO ESMS-Entwurfs soll in erster Linie die grundsätzliche Realisierbarkeit einer E-Service-Technologie zeigen. Eine solche Technologie gliedert sich in den komplexen technischen Kontext der IT-Infrastruktur virt. Unternehmen ein, deren Aufgaben und Funktionen in Sektion 3.2.2 eingeführt wurden. Dieser technische Kontext ist selbst zum großen Teil Subjekt aktueller Forschungsbemühungen und kann keinesfalls als einheitliche, homogene Vorgabe in die Untersuchung eingehen. Auf der anderen Seite können und sollen bei der Implementierung aber auch nicht alle Aspekte einer VDU-Infrastruktur abgedeckt werden, die gar nicht

im Fokus der Untersuchung liegen. Statt dessen soll die Implementierung sich auf die primären Aspekte der E-Service-Technologie konzentrieren. Darüber hinaus sollen in möglichst generischer Weise Wege antizipiert werden, um die Kernfunktionen der E-Service-Technologie in einen breiteren technischen Kontext zu integrieren.

Die primären Aspekte der E-Service-Technologie im Kontext einer breiteren VDU-Infrastruktur beziehen sich auf die Planung und Steuerung virt. Dienstleistungsprozesse auf Basis Service-orient. Koordination und Komposition. Diese Funktionen gliedern sich, wie in Sektion 4.3.1.2 gezeigt wurde, sowohl in Form von horizontalen Basisfunktionen der Kommunikation und Koordination als auch in Form von vertikalen Fachfunktionen in die VDU-Infrastruktur ein. Eine Implementierung muss nun zeigen, dass die konzipierte Art der Planung und Steuerung von Interaktionen zwischen Teilnehmern eines VDPN sich auf Basis konkreter Service-orient. Techniken wirklich durchführen lässt. Dabei müssen auch Schnittstellen zu orthogonalen Funktionsbereichen, z. B. in Bezug auf Sicherheit, aufgezeigt werden. Ferner sind die nicht-funktionalen Anforderungen der VDU-Infrastruktur in Bezug auf Flexibilität, Agilität und Standardkonformität einzubeziehen, wobei insbesondere der letzte Aspekt maßgeblich von der Implementierung beeinflusst wird.

Konkrete funktionale Anforderungen für die Implementierung sind durch die in Sektion 5.2.1.2 und Sektion 5.2.2.2 beschriebenen Entwürfe eines FRESCO-ESMS gegeben. Sie werden in Tabelle 5.4 noch einmal zusammengefasst. Die Anforderungen sind hier nummeriert und in Bezug auf verschiedene Aspekte klassifiziert. Dazu gehört die Herkunft aus dem Methodik- oder Plattform-Entwurf (Quelle) und die Einordnung als horizontale Basisinfrastruktur oder vertikale Fachfunktion der IT-Infrastruktur im VDU (Art). Die Inhalte der Anforderungen ergeben sich direkt aus den einzelnen Modellen und Methoden des FRESCO-ESMS-Entwurfs. Die meisten Inhalte stehen in direktem Bezug zur entworfenen E-Service-Plattform-Architektur und lassen sich dort als Artefakte verschiedener Typen einordnen. Die fundamentalsten Artefakte sind hier Infrastrukturmechanismen in Form von Middleware, die für die Plattform vorausgesetzt werden. Die Standard-Middleware ist in Bezug auf die Komponenten und Beziehungen der Plattform zu erweitern. Die diesbezüglich im Entwurf geforderten Inhalte lassen sich durch Artefakte in Form erweiterter Middleware-Schnittstellen (IDL) und Software-Rahmenwerken (Framework) umsetzen. Für die Plattform werden des Weiteren eine Reihe vertikaler Fachfunktionen zur Entwicklung und Ausführung von E-Services gefordert. Entsprechende Artefakte umfassen verschiedene Plattformkomponenten, mit denen Fachfunktionen in Form von integrierten Mechanismen der Plattform (Services) umgesetzt werden. Zudem gehört hierzu eine Komponente zum E-Service-Schema-Entwurf (Designer) in Form einer autonomen interaktiven Anwendung (Application). In der Tabelle werden zudem die Abhängigkeiten der Anforderungen herausgestellt, aus denen sich Hinweise für den Implementierungsvorgang der Entwürfe ableiten lassen.

In den verbleibenden Sektionen dieses Kapitels wird beschrieben, wie diese Artefakte durch das FRESCO-TK implementiert werden. Hierzu soll noch hervorgehoben

Nr.	Quelle	Art	Inhalt	Artefakt	Typ	Bezug
P1	Plattform	Infrastruktur	Teilnehmer-Plattform Infrastruktur (OGSA)	Infrastruktur	Middleware	
P2	Plattform	Infrastruktur	ESMS-Komponentenmodell	Plattform	IDL	P1
P3	Plattform	Infrastruktur	E-Service-Anwendungsarchitektur	Plattform	IDL	P2
P4	Plattform	Infrastruktur	E-Service-Plattformarchitektur	Plattform	IDL	P2
P5	Plattform	Infrastruktur	E-Service-Ressourcen-Architektur	Plattform	Framework	P3
P6	Plattform	Infrastruktur	E-Service-Ressourcen Konstruktion	Plattform	Framework	P5
P7	Plattform	Infrastruktur	E-Service-Engine Infrastruktur (BPEL)	Infrastruktur	Middleware	
P8	Plattform	Infrastruktur	E-Service-Engine Architektur	Plattform	Framework	P3, P7
M1	Methodik	Fachfunktion	E-Service-Schema UML-Entwurf	Designer	Application	
M2	Methodik	Fachfunktion	E-Service-Instanz BPEL-Entwurf	Schema-Manager	Service	M1, P4
M3	Methodik	Fachfunktion	E-Service-Schema Transformation	Schema-Manager	Service	M1, P4
P9	Plattform	Fachfunktion	E-Service-Engine Konstruktion	Engine-Manager	Service	P4, P7, P8
P10	Plattform	Fachfunktion	E-Service-Aggregation	Aggregation-Manager	Service	M1, P3, P4

Tabelle 5.4.: Funktionale Anforderungen an die FRESCO-ESMS-Implementierung

werden, dass die Implementierung in prototypischer Form erfolgen soll. D. h. vor allem, dass die Anforderungen nicht solche einschließen, die gemeinhin für die Implementierung missionskritischer Anwendungssysteme[23] gefordert werden. Beispiele hierfür sind Anforderungen an die Robustheit, Zuverlässigkeit und Performanz eines Anwendungssystems, die über die Implementierung der Anwendungsfunktion hinaus zusätzliche Maßnahmen erfordern. Des Weiteren werden für die prototypische Implementierung keine expliziten Funktionen gefordert, die in besonderer Weise der Wiederverwendbarkeit, Erweiterbarkeit oder Wartbarkeit eines Anwendungssystems dienen.

5.3.2. Gesamtarchitektur und Techniken

Die Implementierung des FRESCO-ESMS-Entwurfs durch das FRESCO-TK beinhaltet eine Reihe von Artefakten, die der horizontalen IT-Basisinfrastruktur von VDU zugeordnet werden können. Hierunter fallen generell alle generischen Funktionen, die eine allg. Form der Kommunikation oder Kooperation im VDU unterstützen. Entsprechende Artefakte einer FRESCO-ESMS-Plattform wurden in der letzten Sektion identifiziert (P1-P8). Diese beinhalten sowohl Anteile einer spezifischen horizontalen Infrastruktur für E-Service-Plattformen (P2-P6,P8) als auch generische Infrastrukturbestandteile, die in Form von grundlegenden Basistechniken vorausgesetzt werden (P1,P7). In dieser Sektion werden nun zunächst die Infrastrukturbestandteile betrachtet, die für die Realisierung der geforderten Artefakte vorauszusetzen sind. In der nächsten Sektion folgen dann die spezifischen Infrastrukturbestandteile der FRESCO-ESMS-Plattform.

[23] Ein Anwendungssystem wird hier als *missionskritisch* bezeichnet, falls dessen Fehlfunktion zu erheblichem Schaden für die Anwendung führen würde.

Im Folgenden werden in Sektion 5.3.2.1 zunächst die notwendigen Infrastrukturbestandteile identifiziert. Hierzu werden die infrastrukturellen Anforderungen der Artefakte einer ESMS-Plattform untersucht und in eine konkrete Toolkit-Architektur eingeordnet. Im Anschluss werden in Sektion 5.3.2.2 für die Infrastrukturbestandteile konkrete Basistechniken ausgewählt, die als Grundlage für die Konstruktion des Toolkit verwendet werden sollen. Dies beinhaltet vor allem OGSI- und BPEL-konforme Laufzeitumgebungen. Neben Middleware-Techniken werden aber auch solche betrachtet, die dem Entwicklungsvorgang an sich und der Implementierung einzelner Artefakte dienen. Eine Schlüsselrolle für die Betrachtungen der folgenden Abschnitte nimmt hier die GT3 Core Infrastructure als OGSI-Laufzeitumgebung ein. In Sektion 5.3.2.3 werden daher zunächst die wichtigsten OGSI-Konzepte eingeführt. Des Weiteren wird dort der GT3 Core vorgestellt und dessen Erweiterung zu einer FRESCO-ESMS-Plattform erläutert.

5.3.2.1. Toolkit Architektur

In der FRESCO Toolkit Architektur werden die primären Infrastrukturbestandteile mit den ESMS-spezifischen Artefakte des FRESCO ESMS-Entwurfs in Beziehung gesetzt und durch sekundäre Infrastrukturbestandteile einer konkreten Implementierung ergänzt. Das Ergebnis ist als Übersicht in Abbildung 5.12 zu sehen.

Generell ist zunächst anzumerken, dass als fundamentale Infrastruktur natürlich Systemplattformen vorausgesetzt werden. Auf Basis dieser Systemplattformen werden Laufzeitumgebungen für verschiedenartige höherwertige Komponenten eines ESMS bereitgestellt.[24] Es liegt in der Natur der virtuellen Organisationsstruktur, dass die Systemplattformen im VDU als heterogen angenommen werden müssen. Dies stellt kein grundsätzliches Problem dar, denn es könnte für jede Systemplattform eine individuelle Implementierung der ESMS-Plattform vorgenommen werden. Um jedoch den Aufwand bei der Einführung von ESMS zu verringern, sollte eine Implementierung auf Basis von Laufzeitumgebungen vorliegen, die für eine möglichst große Bandbreite an Systemplattformen verfügbar sind. Auf diese Weise kann die ESMS-Plattform auf den unterstützten Systemplattformen ohne zusätzlichen Entwicklungsaufwand realisiert werden.[25]

Auf Basis einer möglichst verbreiteten Laufzeitumgebung ist die wichtigste Infrastrukturkomponente durch eine Grid Middleware-Plattform gegeben. Diese Middleware-Schicht beinhaltet ein Programmiermodell für Grid Services sowie Mechanismen zu deren Ausführung. In der Regel liegt so eine Middleware in Form von API und

[24] Der Begriff der Systemplattform soll hier Hardware und Betriebssystem umfassen. Eine Laufzeitumgebung wird entweder durch die Systemplattform selbst (Betriebssystem) oder eine darauf aufsetzende Middleware-Plattform bereitgestellt (siehe Sektion 3.3.2.1). Allgemein ist hierfür auch der Begriff *Hosting Environment* gebräuchlich.

[25] Beispiele für Laufzeitumgebungen sind Java VM [SDN06] bzw. J2EE [Bod02] oder auch das .Net-Framework [Mic05].

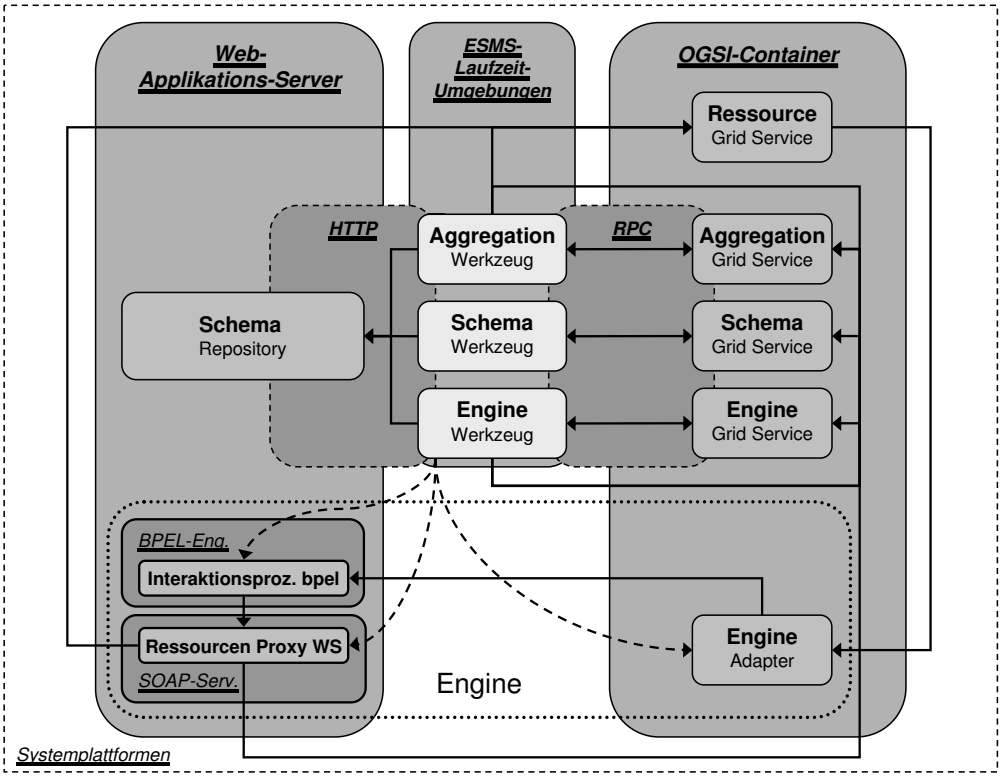

Abbildung 5.12.: FRESCO-TK Architektur

Entwicklungswerkzeugen für die zugrunde liegende Laufzeitumgebung sowie einem Container für Grid Service-Komponenten vor.[26] Konkret fordert der FRESCO ESMS-Entwurf als Basis verschiedener höherwertiger Artefakte eine OGSA-Infrastruktur. Diese wird im Detail durch Spezifikation der Open Grid Service Infrastructure (OGSI) [TCF+03] definiert. Entsprechend kommt zur implementierung des FRESCO-ESMS ein OGSI-konformer Container zum Einsatz. Im Kontext der OGSI Middleware werden verschiedene Artefakte Implementiert, die einer Erweiterung der generischen Grid-Plattform zu einer E-Service-Plattform dienen. Dazu gehören die Definitionen von Grid Service-Schnittstellen für allgemeine und spezifische Komponenten der ESMS-Plattform im Sinne der Anwendungs- und Plattformarchitektur. Spezifische Container-Mechanismen im Sinne einer E-Service-Ressourcen-Architektur und ein Programmiermodell für die Konstruktion von E-Service-Ressourcen können durch Erweiterung der Grid Middleware-API realisiert werden.

Die Fachfunktionen des FRESCO ESMS, die als Grid Service-Komponenten Services der ESMS-Plattform bilden, sollten grundsätzlich nicht komplett auf Basis des OGSI Containers implementiert werden. Der Container ist nicht unbedingt als Laufzeitum-

[26] Unter *Container* wird hier eine Laufzeitumgebung mit Mechanismen und Services für die Ausführung von Middleware-Komponenten verstanden.

gebung komplexer Anwendungssysteme ausgelegt und in der Regel fehlen hier die umfangreichen Mechanismen moderner Middleware-Plattformen, die eine Implementierung komplexer Fachfunktionen erleichtern können. Aus diesem Grund soll die Implementierung der Fachfunktionen stattdessen auf Basis separater Laufzeitumgebungen erfolgen, deren Auswahl in Bezug auf die spezifischen Anforderungen bei deren Implementierung erfolgen kann. In jedem Fall ist dann als zusätzliche Infrastruktur eine Kommunikationsmöglichkeit zwischen den Komponenten zu schaffen, die die Fachfunktionen in einer spezifischen Laufzeitumgebung realisieren und denen, die die Fachfunktionen im OGSI Container repräsentieren und als Grid Services zugreifbar machen. Im FRESCO-TK wird hierfür die klassische Variante eines RPC-Mechanismus gewählt. Im Fall der Schema-Manager-Komponente wird ein Zugriff auf E-Service-Schemata über das Web mittels HTTP realisiert. Hierzu ist die Infrastruktur eines Web-Applikationsservers[27] notwendig.

Die neben dem OGSI Container zweite essenzielle Infrastrukturkomponente wird durch die geforderte BPEL-Laufzeitumgebung gebildet. Eine solche BPEL Engine erlaubt die Ausführung von BPEL-Schemata durch Steuerung von Web Service-Interaktionen und stellt das Herz einer E-Service Engine dar. Die E-Service Engine-Architektur spezifiziert daneben noch verschiedene interne Adapter- und Proxy-Komponenten zur Vermittlung zwischen Web und Grid Service-Interaktionen. Die Realisierung dieser Architektur erfordert eine zusätzliche Infrastruktur in Form von WS Middleware. Dies beinhaltet Werkzeuge, API und Laufzeitmechanismen zur Verarbeitung von WSDL-Spezifikationen und zum entsprechenden Austausch von SOAP-Nachrichten über variable Kommunikationsprotokolle. Während der E-Service Engine Adapter im Kontext des OGSI Containers realisiert wird um die E-Service Engine als Grid Service-Anwendungskomponente der ESMS-Plattform zugreifbar zu machen, ist für den E-Service-Ressourcen-Proxy ein SOAP-Server notwendig. In dessen Laufzeitumgebung kann der Proxy von der BPEL Engine als Web Service angesprochen werden, um den Aufruf dann an den Grid Service einer Anwendungskomponente weiterzuleiten. Sowohl für die BPEL Engine als auch für den separaten SOAP-Server werden im FRESCO-TK Varianten gewählt, die auf einem Applikationsserver basieren.

5.3.2.2. Auswahl von Basistechniken

Für die in der letzten Sektion beschriebenen Infrastrukturbestandteile werden im FRESCO Toolkit konkrete Basistechniken zusammengestellt. Die Auswahl dieser Basistechniken erfolgt auf Grundlage verschiedener Kriterien, die vor allem aus den Rahmenbedingungen des virt. Dienstleistungsunternehmens hervorgehen. Ein fundamentales Kriterium geht auf die schon angesprochene Überlegung zur Voraussetzung einer für möglichst viele Systemplattformen verfügbaren Ausführungsplattform zurück. Ein weiteres Kriterium ist die Entwicklungsweise und Lizenzierung der Tech-

[27] Siehe Sektion 3.3.2.2, S. 126.

Hersteller	Paket	Version	Funktion	Quelle
Middleware-Techniken				
IBM	BPWS4J	2.0	BPEL-Laufzeitumgebung	http://www.alphaworks.ibm.com/tech/bpws4j
Apache Jakarta	Tomcat	4.1.24	Web Application Server	http://tomcat.apache.org/
Globus	GT3 Core	3.0.2	OGSI-Middleware	http://www.globus.org/toolkit/
Apache Web Srv.	Axis	1.1	Web Service Middleware	http://ws.apache.org/axis/
Sun	RMI	1.4.1	Verteiltes OO-System	http://java.sun.com/javase/
Entwicklungstechniken				
Sun	J2SE	1.4.1	OO-Entwicklungs- und Laufzeitumgebung	http://java.sun.com/javase/
Gentleware	Poseidon	2.5.1	UML Werkzeug	http://www.gentleware.com/
Eclipse	Eclipse	3.2	Java IDE	http://www.eclipse.org/
Apache Ant	Ant	1.5	Entwicklungswerkzeug	http://ant.apache.org/
E. Gamma et al	JUnit	3.8.1	Unit Tests	http://www.junit.org/
Hilfstechniken				
Castor Project	Castor	0.9.4.3	XML binding API	http://www.castor.org/
T. Parr et al	ANTLR	2.7.2	Parser Generator	http://www.antlr.org/
Sun	JAX-RPC	1.0	Java API für XML RPC	http://java.sun.com/webservices/jaxrpc/
Sun	Swing	1.4.1	GUI Toolkit	http://java.sun.com/javase/
E.R. Harold et al	XOM	1.0d21	Java XML Objektmodell	http://www.xom.nu/
M. Duftler et al	WSDL4J	1.4	Java WSDL API	http://wsdl4j.sourceforge.net/
Apache Xerces	Xerces-J	2.6.2	XML Parser	http://xerces.apache.org/
Apache Jakarta	Velocity	1.2	Template Engine	http://jakarta.apache.org/
Apache Jakarta	Commons Logging	1.0.3	Java Log API	http://jakarta.apache.org/
Apache Jakarta	Commons Collections	2.1	Collection Bibliothek	http://jakarta.apache.org/
Apache Jakarta	Commons beanutils	1.6.1	JavaBeans Zugriff	http://jakarta.apache.org/
J. Añez	JRCS	0.1.7	Diff Algorithmus	http://www.suigeneris.org/jrcs/
Apache Jakarta	Commons fileUpload	1.0	HTTP Upload	http://jakarta.apache.org/
Apache Jakarta	Commons httpClient	2.0	HTTP Anfragen	http://jakarta.apache.org/
Apache Jakarta	Commons IO	1.0	Dateiverwaltung	http://jakarta.apache.org/

Tabelle 5.5.: Primäre FRESCO-TK-Techniken

niken. Hier soll so genannte *freie Software* bevorzugt werden, die in einem offenen Community-Prozess entwickelt wurde. Gründe hierfür sind u. a. die meist große Basis von Entwicklern und Testern, die weite Verbreitung, die Möglichkeit individueller Anpassungen und die geringen Kosten. Vor allem auch unter diesem Gesichtspunkt soll als einheitliche Ausführungsumgebung die Java-Umgebung (bzw. aus technischer Sicht java VM) gewählt werden. Im Gegensatz zu der aus technischer Sicht plausiblen Alternative des .Net-Framework wird nicht nur Java selbst in einem offenen Prozess entwickelt. Es existiert darüber hinaus auch ein immenses Umfeld offener Software-Entwicklung, in dem freie Implementierungen für nahezu alle notwendigen Infrastrukturbestandteile verfügbar sind, die in vielen Fällen sogar die offiziellen Referenzimplementierungen der zugrundeliegenden Standards bilden. Die entsprechende Auswahl verwendeter Basistechniken wird in Tabelle 5.5 zusammengefasst und im Folgenden erläutert.

Die zentrale Basistechnik des FRESCO-TK ist durch die *GT3 Core Infrastructure* gegeben. Der GT3 Core bildet die offizielle Referenzimplementierung des OGSI-Standards (Version 1.0) und beinhaltet ein Programmiermodell, verschiedene Entwicklungswerkzeuge sowie einen OGSI Container auf Basis von Java. Das Software-Paket ist in zahlreichen wissenschaftlichen und geschäftlichen Umgebungen im produktiven Einsatz und seine Eignung im Kontext des FRESCO-TK steht außer Zweifel.[28] Die Wahl einer BPEL-Laufzeitumgebung gestaltete sich schwieriger, denn zur Zeit der

[28]Eine genauere Betrachtung des GT3 Core erfolgt in der nächsten Sektion.

Entwicklung des FRESCO-TK waren nur sehr wenige Alternativen vorhanden, von denen keine als freie Software verfügbar war. Die möglichen Optionen umfassten hier lediglich den IBM Prototypen *IBM BPEL Java Run Time (BPWS4J)* sowie den kommerziellen *BPEL Orchestration Server* der Firma Collaxa.[29] Da die Collaxa-Engine erhebliche funktionale Mängel aufwies, fiel die Wahl auf BPWS4J.[30] Diese Wahl ist jedoch als exemplarisch aufzufassen, denn das FRESCO-TK ist nur lose an die BPEL Engine gebunden und deren Ersetzung ist konzeptionell einfach. Die Wahl des Applikationsservers fiel auf die etablierte *Tomcat* Software des Apache-Jakarta-Projekts, die sich in zahlreichen professionellen Umgebungen im Einsatz befindet. Ferner kann Tomcat als leistungsfähige Laufzeitumgebung sowohl für den GT3 Core als auch für BPWS4J eingesetzt werden. Dies gilt auch für die verwendete Web Service Middleware, bei der die Wahl auf die *Axis* Software des Apache Web Service-Projekts fiel. Axis bildet die Referenzimplementierung des SOAP-Standards und gilt allg. als stabil und leistungsfähig. In diesem Zusammenhang ist noch die *Java API for XML-Based RPC (JAX-RPC)* zu erwähnen, die als Standard zur Web Service-Programmierung in Java zum Einsatz kommt. Als letzte Basistechnik aus dem Bereich der Middleware ist ein RPC-Mechanismus zu wählen. Hierbei fiel die Wahl auf die *Remote Method Invocation (RMI)* Technik von Sun. Deren Werkzeuge, Mechanismen und API sind standardmäßig in J2SE integriert.

Die Konstruktion der verschiedenen Artefakte des FRESCO-TK auf Basis der Middleware-Techniken erfordert verschiedene Basistechniken zur Software-Entwicklung. Die Basis bildet die *Java 2 Platform, Standard Edition (J2SE)* als grundlegende Sammlung von Werkzeugen und API zur Entwicklung von Software für die Java-Laufzeitumgebung. In Verbindung dazu wird das Poseidon-Modellierungswerkzeug für den grafischen Entwurf von UML-Diagrammen eingesetzt. Poseidon bildet auch die Basis für die Implementierung des FRESCO-TK Entwurfswerkzeugs für E-Service-Modelle als Plugin. Die Wahl von Poseidon gründet sich auf dessen Erweiterungsmechanismus durch einen Plugin-Mechanismus und ein leistungsfähiges Software-Rahmenwerk zur Codegenerierung auf Basis der *Velocity* Software des Apache-Jakarta-Projekts. Zur interaktiven Software-Entwicklung wurde die Eclipse IDE der gleichnamigen Open-Source-Community gewählt. Zur Handhabung der komplexen Konstruktions- und Deployment-Vorgänge im Zusammenhang mit den verschiedenen Middleware-Techniken wurde die *Ant* Software des Apache-Ant-Projekts eingesetzt, die heute als Java-basierter Nachfolger des klassischen „make“-Werkzeugs der Unix-Welt weit verbreitet ist. Weitere Techniken zur Unterstützung des Entwicklungsprozesses umfassen einen in Eclipse integrierten CVS-Mechanismus für die Synchronisation von Entwickler-Teams und die *JUnit* Software von Gamma et al. für das Testen wesentlicher Aspekte der Plattformkomponenten.

[29] Der Collaxa BPEL Orchestration Server ist mittlerweile von der Firma Oracle übernommen worden und in den *Oracle BPEL Process Manager* übergegangen.

[30] BPWS4J ist auf die Interim-März-2003-Version von BPEL 1.1 [ACG+03] ausgelegt. Das FRESCO-TK bezieht sich daher auf diese BPEL-Version.

Für die Implementierung spezifischer Aspekte der Plattform sind verschiedene weitere Basistechniken von Nutzen, die als Hilfstechniken klassifiziert werden können. Ein solcher Aspekt ist die Verarbeitung von E-Service-Schemata in Form von XPDL-Repräsentationen. Da für XPDL keine geeignete Java-API verfügbar ist, muss die Verarbeitung auf Basis generischer XML-API entwickelt werden. Diese Aufgabe erfolgt durch Generierung einer JavaBean-basierten Binding-API für das XML-Schema von *XPDL* mittels der *Castor* Software des gleichnamigen Open-Source-Projekts. Weitere Hilfstechniken zur XPDL-Verarbeitung umfassen die erweiterte XML-API *XOM* von E. R. Harold et al., den XML-Parser *Xerces* des gleichnamigen Apache-Projekts und die JavaBeans-Hilfs-API *beanutils* des Apache-Jakarta-Commons-Projekts. Diese Hilfstechniken dienen auch zur Konstruktion von BPEL-Schemata. Hierzu wird zusätzlich die WSDL-API *WSDL4J* von M. Duftler et al. verwendet. Ein weiterer Aspekt ist die Implementierung der Regelsprache zur E-Service-Schema-Änderung. Hierzu wird der Parser-Generator *ANTLR* von T. Parr et al. verwendet. Die Implementierung eines Verzeichnisses für E-Service-Schema-Spezifikationen erfolgt auf Basis von zwei Mechanismen des Apache-Jakarta-Commons-Projekts zur Unterstützung der Konstruktion von Web-Anwendungssystemen: *fileUpload* und *httpClient*. Weitere verwendete Hilfstechniken in Bezug auf GUI-Entwicklung, Logging, Dateioperationen, Textvergleiche und Datenstrukturen können der Tabelle entnommen werden.

5.3.2.3. Grid-Infrastruktur

Der Entwurf der FRESCO ESMS-Plattform basiert fundamental auf einer Grid-Infrastruktur. Diese Infrastruktur erweitert die grundlegende Web Service-Architektur durch essenzielle Konzepte und Verhaltensmuster zur Realisierung von komplexen verteilten AS. Dies bezieht sich insbesondere darauf, dass Ressourcen hier über den Zeitraum eines kontrollierten Lebenszyklus miteinander in dauerhafter Beziehung stehen und miteinander in geordneter Weise interagieren können. Das Konzept zur Kombination von WS- und Grid-Techniken wird durch die Open Grid Service Architecture (OGSA) [FKN+02b, FKS+05] beschrieben, deren technische Spezifikation durch die Open Grid Service Infrastructure (OGSI) erfolgt und deren Version 1.0 [TCF+03] durch die GT3 Core Infrastructure (GT3 Core) implementiert wird [TCF+03].

Nachdem die Konzepte der OGSA im Zuge des Entwurfs als Grundlage der FRESCO ESMS-Plattformarchitektur verwendet wurden, erfolgt nun die Anwendung und Erweiterung von Primitiven und Protokollen der OGSI sowie deren Implementierung durch den GT3 Core zur Implementierung einer prototypischen ESMS-Plattform. Zum Verständnis dieser Vorgehensweise werden nun zunächst einige wichtige OGSI-Techniken beschrieben, die die schon eingführten OGSA-Konzepte konkretisieren. Im Anschluss erfolgt eine kurze Übersicht des GT3 Core und es wird dessen Erweiterung um die geforderten Artefakte der FRESCO ESMS-Implementierung diskutiert.

OGSI-Basiskonzepte Das wesentliche Prinzip der OGSA besteht darin, virt. Ressourcen in virt. Organisationen durch Grid Services zu repräsentieren. Aus technischer Sicht sind Grid Services dabei grundsätzlich Web Services, deren spezifische Semantik durch erweiterte Verhaltensmuster und Schnittstellen festgelegt wird. In OGSI werden dazu Erweiterungen von WSDL und XML-Schema in Bezug auf zustandsbehaftete Web Services, erweiterbare Web Service-Schnittstellen, asynchrone Benachrichtigung über Zustandsänderungen, Referenzierung und Gruppierung von Service-Instanzen sowie Spezifikation von Service-Attributen mit XML-Schema spezifiziert. Gleich wird zunächst die Erweiterung der Ausdrucksmittel von WSDL betrachtet. Dann werden die verschiedenen OGSI-Prinzipien zum Verhalten von Web Services in Bezug auf Instanziierung, Referenzierung, Erweiterung, Fehlerbehandlung und Lebenszyklus erläutert. Schließlich erfolgt ein Überblick von Schnittstellenspezifikationen, in denen sich die Prinzipien konkret manifestieren.

OGSI 1.0 nutzt die Web Service Interface Definition Language WSDL in der Version 1.1. Diese Sprache wird dabei in ihrer Ausdrucksmächtigkeit zur Beschreibung von Grid Services erweitert und als Grid WSDL (GWSDL) bezeichnet. GWSDL beinhaltet gegenüber der WSDL im Wesentlichen die zusätzlichen Konzepte der Erweiterung einer Web Service-Schnittstelle durch Vererbung und die Repräsentation des Web Service-Zustands durch Attribute. Beide Konzepte werden durch eine Erweiterung des `portType`-Elements wie folgt realisiert:

- *PortType Extension* – GWSDL beinhaltet ein Konzept zur Erweiterung bzw. Vererbung von Schnittstellen, das sich an entsprechende Ansätze von WSDL 1.2 anlehnt. Hierbei erhält das `portType` Element ein zusätzliches optionales Attribut `extends`. Damit kann bei der Spezifikation eines `PortType` Elements P_1 eine Liste beliebiger PortType Elemente $P_2, ..., P_n$ per Qualified Name (QName) referenziert werden. P_1 erbt dann alle `operation` und `serviceData` Elemente von $P_2, ..., P_n$.

- *Service Data* – GWSDL erlaubt die Definition von Attributen, die den öffentlich wahrnehmbaren Zustand eines Grid Service repräsentieren. Diese werden innerhalb jeweils eines komplexen Service Data Elements (SDE) `serviceData` als Unterelement des `portType`-Elements jeweils mitsamt Name, Eigenschaften und XML-Schema Typ deklariert. Varianten statischer, dynamischer, optionaler und erweiterbarer Attribute verschiedener Kardinalität und ggf. mit Standardwerten sind möglich. Der Zugriff erfolgt mittels der Operationen `findServiceData()` und `setServiceData()` des obligatorischen `GridService portType`. Bei der Erweiterung eines `portType` werden auch dessen Attribute vererbt.

Auf Basis von GWSDL erfolgt die Spezifikation von *Grid Service-Beschreibungen*. Letzteres beinhaltet in erster Linie die Syntax der Schnittstelle eines Grid Service sowie in beschränktem Umfang auch seine Semantik. Die Beschreibung dient zum einen als Vorlage zur Implementierung des Grid Service selbst sowie seiner Clients.

Zum anderen dient sie der Suche nach *Grid Service-Instanzen*. Zu jeder Beschreibung können beliebig viele unabhängige und zustandsbehaftete Instanzen existieren.

Eine Grid Service-Instanz ist eindeutig und für alle Zeit durch mindestens einen oder mehrere Grid Service Handles (GSH) gekennzeichnet. OGSI definiert GSH durch einen XML-Schema-Typ `HandleType` als URI. Diese dienen alleine zur Identifizierung von Grid Service-Instanzen und enthalten in der Regel nicht genug Informationen zur Bindung mittels eines spezifischen Protokolls. Hierzu wird eine Grid Service-Instanz stattdessen zusätzlich durch eine oder mehrere Grid Service References (GSR) referenziert. GSR sind in OGSI durch einen abstrakten komplexen XML-Schema-Typ `ReferenceType` definiert. Dieser ist in Bezug auf das spezifische Protokoll zu konkretisieren, für das die Grid Service-Instanz einen Endpunkt bereitstellt. Solche Endpunkte können sich im Laufe des Lebenszyklus einer Grid Service-Instanz ändern und daher können auch GSR ihre Gültigkeit verlieren und durch andere ersetzt werden. Clients leiten gültige GSR entweder eigenständig aus GSH ab oder nutzen die Standardoperation `GridService::HandleResolver::findByHandle()` eines Utility Grid Service. Die wichtigsten Metadaten zur Nutzung einer Grid Service-Instanz können als *Service Locator* zusammengefasst werden. Dessen `LocatorType` kombiniert GSH, GSR und QNames implementierter Schnittstellen.

Verschiedene Aspekte einer Grid Service-Instanz stehen in Zusammenhang zu Zeitpunkten. Dies betrifft z. B. die Gültigkeitsdauer von SDE und GSR sowie die Lebensdauer. Zur Repräsentation von Zeitpunkten übernimmt OGSI das globale GMT-Zeitmodell, klammert jedoch die Synchronisation von Uhren explizit aus. Zeitpunkte können zur Spezifikation der zeitlichen Gültigkeit eines XML-Elements mittels der Attribute `goodFrom`, `goodUntil` und `availableUntil` verwendet werden. Diese Attribute sind z. B. für den `ReferenceType` definiert, können aber auch für alle XML-Elemente mit erweiterbaren Attributen verwendet werden. Hierzu gehören z. B. `serviceData` Elemente.

Der Lebenszyklus einer Grid Service-Instanz betrifft in erster Linie den Lebenszyklus ihrer Implementierung im Kontext einer beliebigen Laufzeitumgebung. Im Kontext von OGSI wird hiervon jedoch abstrahiert. Stattdessen werden grundsätzliche Prinzipien zur Kontrolle des Lebenszyklus mittels Erzeugung und Zerstörung von Grid Service-Instanzen in abstrakter Form definiert. Die Erzeugung von Grid Service-Instanzen erfolgt in der Regel mithilfe eines Factory Grid Service durch die Standardoperation `GridService::Factory::createService()`. Die Factory abstrahiert die Erzeugung der Grid Service-Implementierung. Die resultierende Grid Service-Instanz hat eine begrenzte Lebensdauer, die sich in ihrem Attribut `GridService::terminationTime` ausdrückt und mittels der Operationen `GridService::requestTerminationBefore()` und `GridService::requestTerminationAfter()` verlängert werden kann, solange die Instanz noch benötigt wird (*Soft-State*-Ansatz). In Ergänzung hierzu kann die Instanz aber auch jederzeit mittels `GridService::destroy` explizit zerstört werden.

Zusätzliche OGSI-Verhaltensmuster, die hier nur am Rande erwähnt werden sollen, betreffen Konventionen zur Benennung und Änderung von Schnittstellen, standardi-

sierte Fehlerbehandlung sowie Mechanismen zur Spezialisierung von Operationen der OGSI-Standardschnittstellen. In Bezug auf die Evolution der Beschreibung oder Implementierung eines Grid Service schreibt OGSI vor, dass jede Änderung, die nicht rückwärts kompatibel ist, grundsätzlich in einem neuen `portType` mit eigenem Namen erfolgen muss. Änderungen sind auch oft in Bezug auf die von einem individuellen Grid Service geerbten OGSI-Standardschnittstellen nötig, wenn Parameter von Standardoperationen sich auf Aspekte des individuellen Grid Service beziehen. Ein Beispiel sind spezifische Parameter zur Erzeugung von Instanzen individueller Grid Services mittels `Factory::createService()`. Entsprechende Parameter werden mittels OGSI `ExtensibilityType` als Sequenz aus XML-Schema-`any`-Typen definiert. Zusätzlich existiert dann ein zu der entsprechenden Operation gehöriges `serviceData`-Element vom Typ `OperationExtensibilityType`, das eine statische Liste möglicher konkreter Typen zum korrekten Aufruf der Operation deklariert. Im betrachteten Beispiel ist das `Factory::createServiceExtensibility`. Schließlich sei noch erwähnt, dass OGSI einen `FaultType` als Basis einheitlicher Fehlerbehandlung definiert.

Die beschriebenen Konventionen zur Erweiterung allgemeiner Web Services zu Grid Services besitzen zum einen eine „ideelle“ Dimension als Verhaltensmuster bei der Anwendung von Web Service-Techniken. Zum anderen besitzen sie aber auch eine „physische“ Dimension in Form von standardisierten GWSDL-Schnittstellenbeschreibungen. Die hierin definierten XML-Schema-Typen und GWSDL `portType`-Elemente müssen bei der Beschreibung individueller Grid Services zugrunde gelegt werden. Dabei muss grundsätzlich jede Grid Service-Beschreibung in der Vererbungshierarchie auf den `portType GridService` zurückgehen. Grid Services, die OGSA-spezifische Verhaltensmuster wie die Erzeugung, Dereferenzierung, Benachrichtigung oder Gruppierung unterstützen sollen, müssen andere Standardschnittstellen erweitern.

Die wichtigsten GWSDL-Schnittstellen der OGSI und ihre Rolle im Kontext der OGSA wurden bereits in Sektion 5.2.2.1 eingeführt. Eine komplette Übersicht aller in OGSI spezifizierten `portType-`, `serviceData-` und `operation`-Elemente wird der Vollständigkeit halber in Tabelle 5.6 aufgeführt. An dieser Stelle sollen lediglich noch die Schnittstellen zur Verwaltung von Grid Service-Gruppen kurz erläutert werden, da sie im Kontext der E-Service-Plattform eine besondere Relevanz besitzen.

Die Verwaltung von Grid Service-Gruppen wird durch die `portType`-Elemente `ServiceGroup`, `ServiceGroupEntry` und `ServiceGroupRegistration` geregelt. Eine `ServiceGroup` ist eine Grid Service-Instanz, die Informationen über andere Mitglieder-Grid Service-Instanzen verwaltet. Ihr `portType` definiert zwei SDE. Die `membershipContentRule` definiert Kombinationen von Schnittstellen und Inhalten, von denen für alle Mitglieder der Gruppe mindestens eine zutreffen muss. Die Mitglieder der Gruppe sind in einer `Entry`-Liste jeweils als `EntryType` vermerkt, der den Inhalt sowie je einen Locator des Mitglieds und seiner `ServiceGroupEntry`-Instanz beinhaltet. Der Zugriff auf die Informationen sowie die Lebenszyklusverwaltung der Gruppe erfolgt über die Standardmechanismen des `GridService portType`. Einzelne Mitgliedschaften werden

GWSDL Element	Typ	Inhalt
GridService	pt	Grid Service Basisverhalten
interface	*sd*	*Liste aller portTypes*
serviceDataName	*sd*	*Liste aller serviceData Elem.*
factoryLocator	*sd*	*Service Locator der Factory*
gridServiceHandle	*sd*	*Liste aller GSH*
gridServiceReference	*sd*	*Liste aller GSR*
findServiceDataExtensibility	*sd*	*::findServiceData Erweiterungen*
setServiceDataExtensibility	*sd*	*::setServiceData Erweiterungen*
terminationTime	*sd*	*Terminierungszeitpunkt*
::findServiceData	op	serviceData suchen
::setServiceData	op	serviceData setzen
::requestTerminationAfter	op	Terminierungszeitpunkt setzen
::requestTerminationBefore	op	Terminierungszeitpunkt setzen
::destroy	op	Grid Service zerstören
GridService::HandleResolver	pt	Abstr. Dereferenzierungskonzept
handleResolverScheme	*sd*	*Mögliche URI Schemes von GSHs*
::findByHandle	op	GSH in GSR auflösen
GridService::Factory	pt	Abstr. Instanziierungskonzept
createServiceExtensibility	*sd*	*::createService Erweiterungen*
::createService	op	Grid Service Instanz erzeugen

GWSDL Element	Typ	Inhalt
GridService::NotificationSource	pt	Basis für Abo Anbieter
notifiableServiceDataName	*sd*	*Liste möglicher serviceData Elem.*
subscribeExtensibility	*sd*	*::subscribe Erweiterungen*
::subscribe	op	Abo registrieren
GridService::NotificationSubscription	pt	Repräsentation von Abos
subscriptionExpression	*sd*	*Abo Beschreibung*
sinkLocator	*sd*	*Service Locator des Empfängers*
NotificationSink	pt	Basis für Abo Empfänger
::deliverNotification	op	Abo liefern
GridService::ServiceGroup	pt	Repräsentation von Gruppen
membershipContentRule	*sd*	*Bedingung für Mitgliedschaft*
entry	*sd*	*Liste von Mitgliedern*
GridService::ServiceGroupEntry	pt	Repräsentation von Mitgliedern
memberServiceLocator	*sd*	*Service Locator des Mitglieds*
content	*sd*	*Mitgliedschaftsbasis*
ServiceGroup::ServiceGroupRegistration	pt	Basis für Gruppenverwaltung
addExtensibility	*sd*	*::add Erweiterungen*
removeExtensibility	*sd*	*::remove Erweiterungen*
::add	op	Hinzufügen von Mitgliedern
::remove	op	Entfernen von Mitgliedern

Tabelle 5.6.: OGSI Grid Service-Standardschnittstellen

als `ServiceGroupEntry`-Instanz repräsentiert. Diese beinhaltet ebenfalls zwei SDE: einen Locator `memberServiceLocator` der Mitglied-Grid Service-Instanz und deren Inhalt `content`. Die Dauer der Mitgliedschaft korreliert mit dem Lebenszyklus der `ServiceGroupEntry`-Instanz und wird über die Standardoperationen des `GridService portType` geregelt. Schließlich wird durch die `ServiceGroupRegistration`-Schnittstelle eine Erweiterung der `ServiceGroup` durch Operationen zur Verwaltung von Mitgliedschaften definiert. Dies beinhaltet das Zufügen von Mitgliedern durch `add()` und das Entfernen von Mitgliedern durch `remove()`. Die Operationen enthalten jeweils erweiterbare Parameter für Inhalte von Gruppenmitgliedern. Deren konkrete Typen müssen in entsprechenden SDE und in Konformität zur `membershipContentRule` der `ServiceGroup` spezifiziert werden.

GT3 Core-Integration Die OGSI-Spezifikation bildet die Grundlage zur Spezifikation erweiterter Mechanismen der E-Service-Plattform-Infrastruktur. Die Implementierung der E-Service-Plattform erfolgt entsprechend als Erweiterung einer Implementierung von OGSI. Im FRESCO-TK liegt diese OGSI-Implementierung in Form der GT3 Core Infrastructure vor. Der GT3 Core liefert eine Laufzeitumgebung, die Grid Services aufnehmen und ausführen kann. Die Laufzeitumgebung vermittelt dabei zwischen einer Anwendung, die den Inhalt des Grid Service realisiert und der darunter liegenden WS-Middleware (im Wesentlichen SOAP-Engine), die den Inhalt als Grid Service zugreifbar macht. Des Weiteren unterstützt der GT3 Core die Entwicklung von Grid Services durch Programmiermodelle für die Erbringung und Nutzung von Grid Services durch AS sowie durch Werkzeuge.

Abb. 5.13 gibt einen Überblick des GT3 Core, der hierin die weißen Elemente umfasst. Die *OGSI-Referenzimplementierung* beinhaltet Implementierungen aller OGSI-

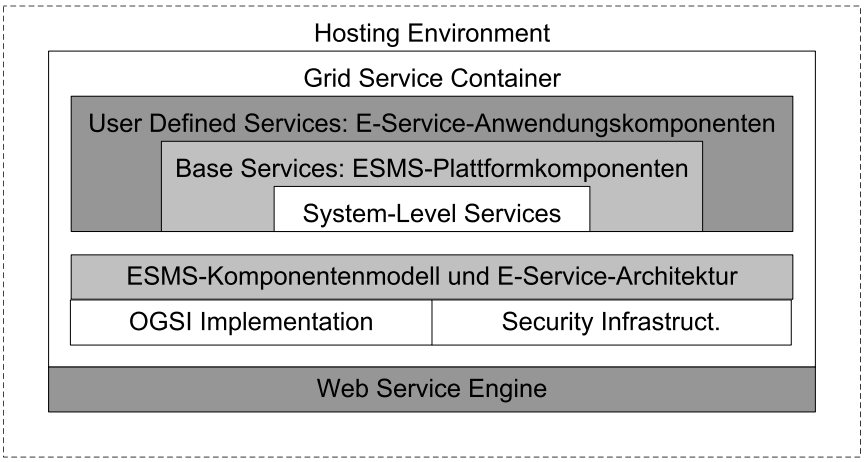

Abbildung 5.13.: FRESCO GT3 Core-Architektur (nach [SG03])

Schnittstellen sowie API und Werkzeuge zur Entwicklung OGSI-konformer Services. Die *Sicherheitsinfrastruktur* beinhaltet die Sicherung von Nachrichten auf SOAP- sowie Transportebene, wechselseitige Punkt-zu-Punkt-Authentifizierung und Authorisation sowie „single-sign-on“ Funktionalität. Neben diesen Grundfunktionen für Grid Services beinhaltet der GT3 Core auch einige aufbauende systemtechnische *Laufzeitmechanismen* (System-Level Services) auf Infrastrukturebene. Hierzu gehören z. B. Mechanismen zur Administration der Laufzeitumgebung oder zum Logging. Im Wesentlichen auf Basis der OGSI- und Sicherheitsfunktionen werden auch so genannte *Basisdienste* (Base Services) realisiert. Hier liefert das erweiterte GT3 verschiedene Grid Services z. B. für Programmausführung, Datenmanagement und Informationsdienste, die jedoch nicht zum GT3 Core gehören und im Kontext des FRESCO-TK keine Rolle spielen. Noch eine Ebene höher werden dann am Ende individuelle *benutzerdefinierte Services* (user-defined Services) realisiert. Alle Grid Services interagieren mit der Laufzeitumgebung in der abstrakten Form eines *Grid Service Containers*. Der Container kapselt Implementierungsdetails der Laufzeitumgebung. Er beinhaltet zudem Mechanismen, um den Lebenszyklus von Grid Service-Instanzen zu verwalten und externe Anfragen an diese zu leiten. Grid Services können in verschiedenen Implementierungen des Containers ablaufen. Dieser kapselt und erweitert dabei jeweils eine Web Service Engine, bei der es sich in der konkreten Container-Implementierung um Apache Axis handelt. Schließlich laufen Container und Web Service Engine in einer übergeordneten Laufzeitumgebung ab, für die der GT3 Core einige Alternativen anbietet.

Die Implementierung der FRESCO ESMS-Plattform gliedert sich als Erweiterung von OGSA-Konzepten und OGSI-Konstrukten in die Architektur des GT3 Core ein. Dies wird in Abb. 5.13 durch hellgraue Elemente verdeutlicht. Parallel zur Erweiterung von OGSI Grid Services als ESMS-Komponenten setzt das FRESCO-TK auf der OGSI-

Implementierung des GT3 Core auf und erweitert sie um Implementierungen von FRESCO ESMS-Komponenten. Des Weiteren werden hier die Architekturmodelle für E-Service-Anwendungssysteme und für die E-Service-Plattform als OGSI-Erweiterungen definiert und Mechanismen zur Entwicklung von E-Service-Komponenten als Grid Services im GT3 Core Container bereitgestellt. Alles zusammen ergibt eine spezifische ESMS-Infrastruktur. Die vertikalen Fachfunktionen der ESMS-Plattform werden hingegen als aufbauende Basisdienste implementiert. Auf diesen Grundlagen können dann konkrete Anwendungskomponenten für E-Services als benutzerdefinierte Services realisiert werden. Diese werden auf Basis der ESMS-Infrastruktur entwickelt und interagieren zur Laufzeit mit den Fachfunktionen der Plattformkomponenten. In den nachfolgenden Sektionen wird nun zunächst die Implementierung der ESMS-Infrastruktur erläutert. Im Anschluss folgt dann die Implementierung der ESMS-Fachfunktionen.

5.3.3. E-Service-Plattform-Infrastruktur

Nach der Festlegung einer fundamentalen Infrastruktur vorausgesetzter Basistechniken für das FRESCO-TK werden nun aufbauende Infrastrukturbestandteile der FRESCO-ESMS-Plattform als Grundlage für die Realisierung von E-Service-Anwendungs- und -Plattformkomponenten betrachtet.

In diesem Sinne fokussiert Sektion 5.3.3.1 die Spezifikation der E-Service-Anwendungsarchitektur auf Basis von Grid Services und die Implementierung von Anwendungskomponenten mit den Basistechniken. Erst werden erweiterte Grid Service-Schnittstellen für E-Service-Ressourcen und Engines definiert und deren Interaktionen bei der E-Service-Ausführung betrachtet. Dann werden die internen Architekturen von E-Service-Ressourcen und Engines in der Laufzeitumgebung erläutert und ein Programmiermodell für E-Service-Ressourcen vorgestellt.

Am Ende diese Abschnitts erfolgt in Sektion 5.3.3.2 die Spezifikation der E-Service-Plattformarchitektur auf Basis der Grid-Infrastruktur. Hier wird als Grundlage zunächst die technische Repräsentation von E-Service-Schemata als persistente Archive und als transiente Datenstrukturen zur Kommunikation von Meta-Informationen zwischen Plattformkomponenten beschrieben. Aufbauend werden die einzelnen Plattformkomponenten dann als Grid Services spezifiziert und deren Interaktionen bei der E-Service-Entwicklung und -Ausführung erläutert.

5.3.3.1. E-Service-Anwendungssystem-Architektur

Die FRESCO-TK E-Service-Anwendungssystem-Architektur beinhaltet eine Erweiterung des OGSI-Komponentenmodells sowie Implementierungen der erweiterten Komponententypen mittels verschiedener Basistechniken der FRESCO-TK-Ausführungsplattform. Gleich werden zunächst erweiterte Schnittstellen und Konventionen für E-Service-Ressourcen und Engines auf Basis der OGSI-Spezifikation formuliert.

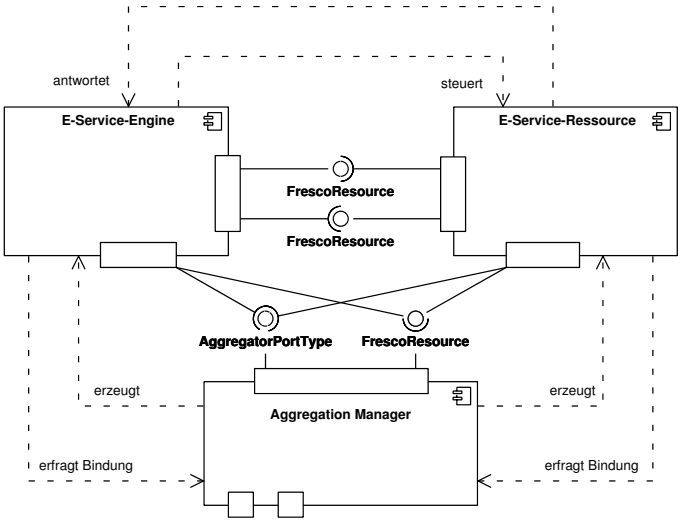

Abbildung 5.14.: FRESCO-TK E-Service-Anwendungssystem-Architektur

Im Anschluss wird die Implementierung von E-Service-Ressourcen mittels eines Programmiermodells auf Basis der OGSI-Implementierung des GT3 Core betrachtet. Schließlich erfolgt ein Überblick der E-Service Engine-Architektur. Die Architektur setzt anteilig auf dem GT3 Core, der Axis SOAP-Engine sowie der BPWS4J BPEL Engine auf und bildet die Basis zur automatischen E-Service-Konstruktion durch den FRESCO-TK Engine-Manager.

Anwendungskomponenten und -Interaktion Die Architektur eines FRESCO E-Service-Anwendungssystems basiert auf zwei grundlegenden Typen von Anwendungskomponenten. E-Service-Ressourcen repräsentieren die Assets von FRESCO-Providern und erlauben deren Interaktion im Kontext eines virt. Dienstleistungsprozesses während der E-Service-Ausführung. E-Service Engines repräsentieren Capabilities von FRESCO-Koordinatoren und erlauben die aktive Steuerung spezifischer VDLP-Anteile während der E-Service-Ausführung. Abb. 5.14 zeigt die Komponenten der FRESCO E-Service-Anwendungsarchitektur in ihrer Umsetzung im FRESCO-TK als UML Deployment-Diagramm.

Die Implementierung von Anwendungskomponenten als erweiterte OGSI Grid Services basiert auf einer Abstraktion der internen Prozesse von VDPN-Teilnehmern. Unabhängig davon, ob die Anwendungskomponente eines Teilnehmers auf einem internen Asset oder einer Capability basiert, wird nur deren fundamentale Eigenschaft als Ressource im globalen VDLP betrachtet. In diesem Sinne wird zur Spezifikation von Anwendungskomponenten als erweiterte OGSI Grid Services nur die eine Standardschnittstelle des `FrescoResource portType` definiert, die alle E-Service-Ressourcen und Engines implementieren müssen.

Der `FrescoResource portType` erfüllt im Wesentlichen zwei Aufgaben. Zum einen erweitert er indirekt den OGSI `GridService portType` und definiert dadurch alle Anwendungskomponenten als OGSI-konforme Grid Services. Genauer wird als direkte Erweiterung des OGSI `GridService portType` zunächst ein abstrakter `FrescoPortType` definiert. Die `FrescoPortType`-Schnittstelle selbst enthält keinerlei Service Data Elements oder Operationen, sondern dient lediglich zur Markierung der Semantik eines OGSI Grid Services als generische Komponente eines FRESCO E-Service-Management-Systems. Dies spiegelt sich in den `interface` SDE aller ESMS-Komponenten wider. Der wesentlichere Effekt besteht jedoch darin, dass alle ESMS-Komponenten und insbesondere die Anwendungskomponenten das OGSI-Komponentenmodell erben. D. h., dass alle Anwendungskomponenten als zustandsbehaftete Web Services mit Soft-State Lebenszyklus und SDE-Attributen vorliegen. Auf dieser Basis werden die konzeptionellen Kontaktpunkte zur Interaktion zwischen Assets und Capabilities eines E-Service-Modells als Grid Service-Operationen individueller Schnittstellen implementiert, die von `FrescoResource` abgeleitet werden.

Die zweite Aufgabe des `FrescoResource portType` besteht darin, verschiedene Elemente im Zusammenhang mit einheitlichen Verhaltenskonventionen zu definieren, die für alle Anwendungskomponenten vorgeschrieben werden.[31] Hierbei werden zunächst Konventionen übernommen, die durch den übergeordneten `FrescoPortType` definiert werden.[32] Dies betrifft im Wesentlichen die Festlegung von Datenformaten für die Repräsentation und Kommunikation von Meta-Informationen des ESMS.[33] Des Weiteren wird ein spezifischer Fehlertyp `fresco:FrescoFaultType` eingeführt, der für alle Operationen von ESMS-Komponenten zur einheitlichen Fehlerbehandlung verwendet werden sollte. Aufbauend wird im `FrescoResource portType` ein `frescoRessourceState` SDE vom XML-Schema-Typ `fcr:ResourceStateType` definiert (Listing 5.5).

```
<xsd:complexType name="ResourceStateType">
  <xsd:complexContent>
    <xsd:extension base="fresco:ResourceInstanceType">
      <xsd:sequence>
        <xsd:element name=" status" type=" string"/>
        <xsd:element name="timestamp" type="xsd:dateTime"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Listing 5.5: `frescoRessourceState` SDE

[31] GWSDL- und XML-Schema-Elemente für FRESCO-Anwendungskomponenten werden in einem XML-Namensraum zusammengefasst, der im Folgenden mit dem Präfix `fcr` abgekürzt wird.

[32] GWSDL- und XML-Schema-Elemente für generische FRESCO ESMS-Komponenten werden in einem XML-Namensraum zusammengefasst, der im Folgenden mit dem Präfix `fresco` abgekürzt wird.

[33] Eine genauere Betrachtung der ESMS-Datenformate erfolgt in Sektion 5.3.3.2.

GWSDL Element	Typ	Inhalt
GridService:: FrescoPortType	pt	Definition von Nachrichtenformaten und Fehlertypen für alle Komponenten
FrescoPortType:: FrescoRessource	pt	Management von Meta-Informationen für alle E-Service-Ressourcen
frescoRessourceState	*sd*	*Metainformationen ueber den Zustands einer E-Service-Ressource*
::setFrescoResourceState	op	Zuweisung von Meta-Informationen

Tabelle 5.7.: FRESCO-TK-Basiskomponentenschnittstellen

Im SDE werden für alle Anwendungskomponenten diverse Meta-Informationen bereitgestellt, die auf der Erweiterung des `fresco:ResourceInstanceType` basieren. Hierin wird im Wesentlichen die Beziehung einer Anwendungskomponente zu ihrer E-Service-Instanz beschrieben. Dies beinhaltet Informationen über Klassen und die Kennungen des E-Service, der E-Service-Ressource und des Teilnehmers, der die Ressource bereitstellt. Diese Informationen müssen bei der Erzeugung einer Ressource durch die Plattformkomponente des Aggregation-Managers zugewiesen werden.[34] Hierzu dient die Operation `FrescoResource::setFrescoResourceState()`. Falls Instanzen von Anwendungskomponenten außerhalb des Kontextes einer E-Service-Instanz erzeugt werden, sollte beim Versuch der Interaktion mit diesen ein spezifischer Fehler vom Typ `fcr:resourceNotConfiguredFault` resultieren. Tabelle 5.7 fasst die Erweiterungen der OGSI Grid Service-Schnittstelle für Grid Service-Komponenten der FRESCO ESMS-Plattform zusammen.

Zur kompletten Beschreibung der E-Service-Anwendungssystem-Architektur muss noch ein Verhaltensmuster in Bezug auf die Vorgehensweise bei der aktiven Durchführung von Interaktionen zwischen Anwendungskomponenten spezifiziert werden. Die aktive Interaktion einer Anwendungskomponente mit einer anderen erfolgt z. B. im Zuge der Steuerung des VDLP durch eine E-Service Engine oder bei einer asynchronen Antwort durch eine E-Service-Ressource. Hierzu muss die aktive Komponente eine Bindung des Interaktionspartners vornehmen. Diese Bindung erfolgt grundsätzlich dynamisch im Kontext einer E-Service-Instanz. Sie ist von der Ressourcenklasse und der Teilnehmerrolle des Interaktionspartners sowie den aktuellen Inhalten der Interaktion abhängig. Die aktive Komponente muss den GSH des Interaktionspartners mittels dieser Informationen von einer Aggregation-Manager-Plattformkomponente ermitteln, die den Interaktionspartner daraufhin ggf. erst erzeugt. Hierzu dient die Operation `correlatedAssociation()` des `AggregatorPortType` einer Aggregation-Manager-Plattformkomponente. Anschließend kann die Anwendungskomponente den GSH in geeigneter Weise auflösen und die Interaktion durchführen.

[34] Die genaue Spezifikation der Aggregation-Manager-Komponente erfolgt in Sektion 5.3.3.2.

E-Service-Ressourcen-Architektur Aufbauend auf der Spezifikation einer E-Service-Ressource als erweiterter OGSI Grid Service soll nun die interne Software-Architektur entsprechender Komponenten und deren Implementierung im Kontext des GT3 Core erläutert werden. Die Betrachtung beschränkt sich dabei auf die Architektur und Implementierung eines E-Service-Ressource Grid Service im GT3-Container. Dessen Integration mit operativen Anwendungssystemen von VDPN-Teilnehmern ist ein individuelles Problem. Hier sind spezifische A2Ai-Lösungen auf Basis organisationsinterner integrativer Middleware-Techniken zu finden, wie sie im Grundlagenteil der vorliegenden Arbeit in allgemeiner Form beschrieben wurden.[35]

Zur Implementierung von Grid Services beinhaltet der GT3 Core Client- und Serverseitige Programmiermodelle auf Basis eines Software-Rahmenwerks für Java. Die Implementierung von E-Service-Ressource Grid Services beinhaltet eine Kombination dieser Anteile und erfordert darüber hinaus den Umgang mit ESMS-Datenformaten und der Plattformkomponente des Aggregation-Managers. Entsprechend basiert die interne Architektur von E-Service-Ressource Grid Services auf den Super- und Utility-Klassen aller drei Bereiche. Im Folgenden werden die einzelnen Bereiche an einem Beispiel erläutert.

Das betrachtete Beispiel fokussiert die Architektur und Implementierung der E-Service-Ressource *TransportControl*. Diese steht im Zusammenhang mit der E-Service Engine *TransportCoordination*, die als gegeben angenommen wird. Die Schnittstellen beider Anwendungskomponenten sind als GWSDL `portType` spezifiziert. Der `TransportControlPortType` beinhaltet hierbei eine Operation `upcomingTransport()` und der `TransportCoordinationPortType` eine weitere Operation `awaitStatusReport()`. Im Laufe des VDLP ruft TransportCoordination die Operation `upcomingTransport()` bei TransportControl auf. Später ruft TransportControl dann seinerseits `awaitStatusReport()` bei TransportCoordination auf. Bei der Realisierung von TransportControl muss also zunächst der `TransportControlPortType` implementiert werden. Im weiteren Verlauf wird dann die Aggregation-Manager-Plattformkomponente zwecks GSH für den `TransportCoordinationPortType` kontaktiert. Anschließend erfolgt dann der Aufruf von `awaitStatusReport()`. Die Implementierung von TransportControl umfasst also (a) einen Server-Anteil, der die Inhalte von `TransportControlPortType` erbringt und als Grid Service verfügbar macht, (b) einen Client-Anteil, der sich an einen `AggregatorPortType` bindet und die Operation `correlatedAssociation()` aufruft und (c) einen Client-Anteil, der sich an `TransportCoordinationPortType` bindet und `awaitStatusReport()` aufruft. Abbildung 5.15 zeigt die resultierende Architektur von TransportControl und dessen Implementierung auf Basis des FRESCO-TK Software-Rahmenwerks im Überblick. In der Abbildung gehören Elemente mit ≪`globus`≫ Stereotyp entweder zum GT3 Core Software-Rahmenwerk oder werden davon generiert. Elemente mit ≪`fresco`≫ Stereotyp werden vom FRESCO-TK Software-Rahmenwerk ergänzt. Elemente mit ≪`ressource`≫ Stereotyp stellen die individuelle Implementierung dar.

[35] Siehe Sektion 3.3.2.3.

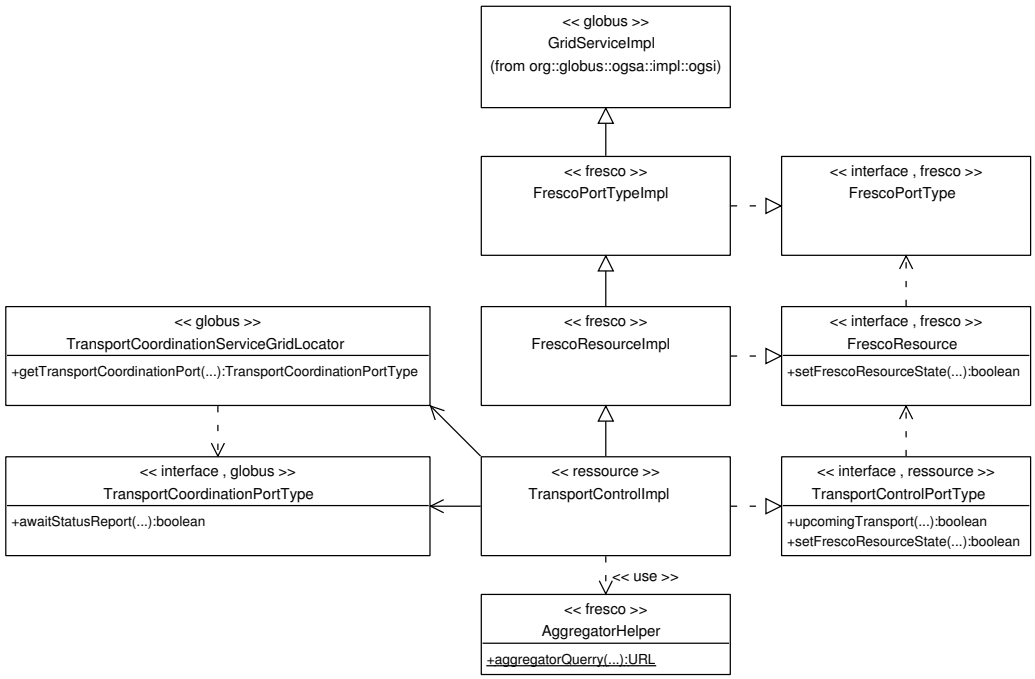

Abbildung 5.15.: FRESCO-TK E-Service-Ressourcen-Programmiermodell

Die Server-seitige Programmierung von Grid Services basiert auf einer Implementierung des OGSI `GridService portType` durch die `GridServiceBase`-Schnittstelle des GT3 Core und ihre beiden alternativen Implementierungen für entweder transiente oder persistente Grid Services. Im FRESCO-TK dient die transiente Implementierung `GridServiceImpl` als Basis zur Ableitung von abstrakten Implementierungen der beiden grundlegenden Schnittstellen `FrescoPortType` und `FrescoResource`. Zur Implementierung eines E-Service-Ressource Grid Service muss nun zunächst dessen Schnittstellenbeschreibung in korrespondierende Java-Klassen übersetzt werden. Hierzu beinhaltet der GT3 Core die üblichen Werkzeuge einer Web Service Middleware, um zu GWSDL-Definitionen Java-Schnittstellen, -Stub- und -Skeleton-Klassen sowie -Datentypen zu generieren, die den Abbildungsvorschriften der JAX-RPC-Spezifikation entsprechen. Im Beispiel resultiert die Compilierung des `TransportCoordinationPortType` u. a. in einer gleichnamigen Java-Schnittstelle, die implementiert werden muss. Neben den individuellen Operationen beinhaltet die Schnittstelle alle Operationen von `portType` Elementen der Vererbungshierarchie. Deren Implementierung erfolgt durch Erweiterung von `FrescoResourceImpl`. Es verbleibt die Implementierung der individuellen Operationen. Dies erfordert in der Regel die Integration von operativen Anwendungssystemen. Ein Beispiel zur Anbindung autonomer Anwendungssysteme mittels klassischer RPC-Middleware ist im FRESCO-TK durch die Implementierungen der Plattformkomponenten gegeben.

Durch die dynamischen Bindungen zwischen Anwendungskomponenten muss die Interaktion mit dem Aggregationsmechanismus als Teil jeder Anwendungskomponente implementiert werden, die aktive Interaktionen betreibt. Aus diesem Grund erfolgt die Implementierung der hierzu notwendigen Vorgänge durch die Hilfsklasse `AggregatorHelper` des FRESCO-TK Software-Rahmenwerks. Die Klasse beinhaltet die statische Methode `aggregatorQuery()`, deren Parameter die notwendigen Meta-Informationen zur Aggregation einer Ressourcen-Instanz umfassen. Die Klasse wandelt diese Informationen in das Datenformat der ESMS-Plattform, bindet sich an eine Grid Service-Instanz der Aggregation-Manager-Plattformkomponente und stößt die Aggregation der nachgefragten Anwendungskomponente an. Im Beispiel ist das eine Instanz der E-Service Engine TransportCoordination. Das Ergebnis ist ein GSH, den die Hilfsklasse aus dem Datenformat der ESMS-Plattform in die Java-Repräsentation einer URI wandelt.

Der verbleibende Client-seitige Bereich der Architektur dient zum Aufruf individueller Operationen einer anderen Anwendungskomponente. Hierzu muss zunächst deren Schnittstellenbeschreibung vorliegen. Hieraus wird wiederum mithilfe eines Werkzeugs Java Code für einen Client Stub gemäß JAX-RPC generiert. Im Beispiel wird für TransportCoordination eine Java-Schnittstelle des `TransportCoordinationPortType` erzeugt. Daneben erlaubt ein zugehöriger `TransportCoordinationServiceGridLocator` die Auflösung der GSH URI in eine GSR und liefert mit der Operationen `getTransportCoordinationPort()` einen konfigurierten Stub, der zum Zugriff auf `awaitStatusReport()` genutzt werden kann.

E-Service Engine-Architektur E-Service Engines sind spezielle Anwendungskomponenten, deren Anteil am globalen VDLP nicht auf den lokalen DLP eines realen Assets zurückgeht, sondern von einer BPEL-Laufzeitumgebung als Steuerprozesses einer Capability synthetisiert wird. Davon abgesehen kann die E-Service Engine jedoch ebenfalls als E-Service-Ressource betrachtet werden, die sich mittels `FrescoResource portType` in die E-Service-AS-Architektur fügt. Daher basiert die interne E-Service Engine-Architektur auch auf der E-Service-Ressourcen-Architektur. Dies verteilt sich hier aber über mehrere Komponenten, die gemeinsam der Kapselung einer BPEL Engine dienen. Abbildung 5.16 zeigt die Architektur im Überblick.

Der Server-seitige Bestandteil der E-Service-Ressourcen-Architektur befindet sich im E-Service Engine Adapter. Die Implementierung der individuellen Operationen erfolgt durch einen Web Service Client Stub, der die Operationen an die BPEL Engine leitet. Aktive Interaktionen werden durch die BPEL Engine angestoßen und zunächst vom Web Service Server Stub eines E-Service-Ressourcen-Proxy aufgefangen. Dieser beinhaltet die Interaktion mit dem Aggregation-Manager und den Client-seitigen Bestandteil der E-Service-Ressourcen-Architektur. Eine genauere Betrachtung der zusätzlichen Bestandteile der E-Service Engine-Architektur und deren Konstruktion erfolgt in Sektion 5.3.4.3.

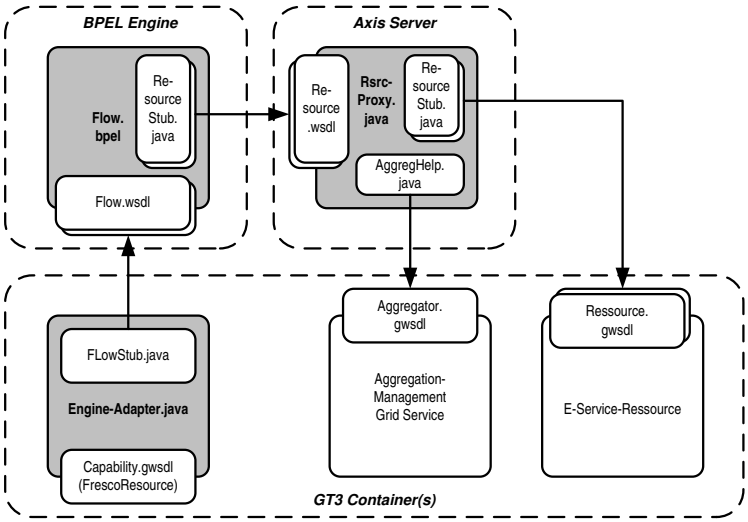

Abbildung 5.16.: FRESCO-TK E-Service Engine-Architektur

5.3.3.2. E-Service-Plattform-Architektur

Die FRESCO E-Service-Plattform bildet die Laufzeitumgebung eines ESMS, in deren Kontext sowohl die Entwicklung als auch die Ausführung von E-Services als integrierter Prozess im Sinne des FSMV stattfindet. Die Plattform umfasst zum einen Anteile einer horizontalen Basisinfrastruktur auf Basis von Grid Service Middleware und zum anderen Anteile vertikaler Fachfunktionen, die Entwicklungs- und Ausführungsmethoden umsetzen. In der letzten Sek. wurden die Grundlagen der horizontalen Basisinfrastruktur als Erweiterung des OGSI-Komponentenmodells und einer entsprechenden Erweiterung von dessen Implementierung im GT3 Core realisiert. Des Weiteren wurde die darauf aufsetzende Architektur von E-Service-Anwendungssystemen und die Implementierung von deren Anwendungskomponenten gezeigt.

In dieser Sektion wird nun die Architektur der Plattformebene betrachtet. Dies umfasst im Wesentlichen Plattformkomponenten für vertikale Fachfunktionen und deren Funktionsweise bei der Entwicklung von E-Service-Schemata, der Implementierung von E-Service-Instanzen und der Ausführung von E-Service-Anwendungssystemen. Als einleitender Schritt wird hierzu die Repräsentation von E-Service-Schemata und -Instanzen in Form von XML-Nachrichtenformaten zum Austausch von E-Service-Meta-Informationen zwischen Plattformkomponenten betrachtet. Im Anschluss werden dann die Plattformkomponenten als erweiterte ESMS Grid Services spezifiziert und deren Interaktion untereinander sowie mit den Komponenten laufender E-Service-Anwendungssysteme erläutert. Am Ende wird eine generische Architektur sowie fundamentale Bestandteile zur Implementierung einzelner Plattformkomponenten dargestellt.

Repräsentation von E-Service-Schemata und -Instanzen Die Plattformkomponenten des ESMS erbringen zum einen Fachfunktionen für die E-Service-Entwicklung, bei denen E-Service-Schemata formuliert, registriert, analysiert, manipuliert, transformiert und archiviert werden. Zum anderen erfüllen sie Fachfunktionen für die E-Service-Ausführung, bei denen E-Service-Instanzen aggregiert werden, was die Assoziation von Rollen, die Installation von Anwendungskomponenten und die Allokation von Ressourcen beinhaltet. Die Realisierung dieser Funktionen erfordert eine Repräsentation von E-Service-Schemata und -Instanzen auf Ebene der Plattform und einzelner Komponenten. Letztere sind weitgehend frei, die interne Repräsentationsform auf ihre individuelle Funktion abzustimmen. Auf Plattformebene sind jedoch einheitliche Formate zu definieren, die es den ESMS-Komponenten ermöglichen, gemeinsam auf E-Service-Schemata zuzugreifen und miteinander E-Service-Meta-Informationen auszutauschen. In diesem Sinne legt das FRESCO-TK Formate für *persistente* und *transiente* Repräsentationen von E-Service-Schemata und -Instanzen fest.

Persistente Repräsentation erfasst E-Service-Schemata als Menge von XML-Dokumenten, die aus dem E-Service-Entwurfsvorgang hervorgehen und im ESMS archiviert werden. Solche so genannten *E-Service-Schema-Spezifikationen* bilden die Basis der E-Service-Repräsentation im ESMS. Sie beinhalten verschiedene Teilspezifikationen in Form von XPDL-, WSDL- und GWSDL-Definitionen.[36] Diese Anteile werden auf standardisierte Art und Weise in XML-Namensräume unterteilt, als Paket von XML-Dokumenten strukturiert und im Web archiviert. Die Orte der Archivierung müssen sich per URI adressieren und zugreifen lassen, wobei der Zugriff im FRESCO-TK per HTTP erfolgt. Die transiente Repräsentation erfasst spezifische, zum Teil temporäre Aspekte von E-Service-Schemata und -Instanzen als XML-Nachrichten, die zwischen ESMS-Komponenten ausgetauscht werden. Der Austausch dieser so genannten *E-Service-Meta-Informationen* bildet die Grundlage zur Kooperation von ESMS-Komponenten im Zuge einer integrierten Vorgehensweise im Sinne des FSMV. *E-Service-Meta-Informationen* müssen zur Interaktion im Rahmen des zugrunde liegenden OGSI-basierten Komponentenmodells in Form von XML-Nachrichten erfasst werden. Aus diesem Grund werden XML-Schema-Typen definiert, die zur Datenmodellierung von E-Service-Meta-Informationen und als Vorgabe für Nachrichtenformate dienen. Als Grundlage für die nachfolgende Spezifikation von Plattformkomponenten wird im Folgenden das resultierende Datenmodell erläutert.

Das Datenmodell für transiente E-Service-Meta-Informationen wird in Abb. 5.17 in einer Repräsentation als UML-Klassendiagramm gezeigt.[37] Ein Grundprinzip ist hierin die Verknüpfung von E-Service-Meta-Informationen mit der E-Service-Schema-Spezifikation. Hierzu werden Basistypen definiert, die einzelne Elemente der E-

[36] Eine genauere Beschreibung der E-Service-Schema-Spezifikation und -Archivierung erfolgt in den Sektionen 5.3.4.1 (E-Service-Entwurfswerkzeug) und 5.3.4.2 (E-Service-Schema-Manager).

[37] Als Grundlage für die Repräsentation von XML-Schema-Typen als UML-Klassen wurde deren offizielle Java-Abbildung gewählt.

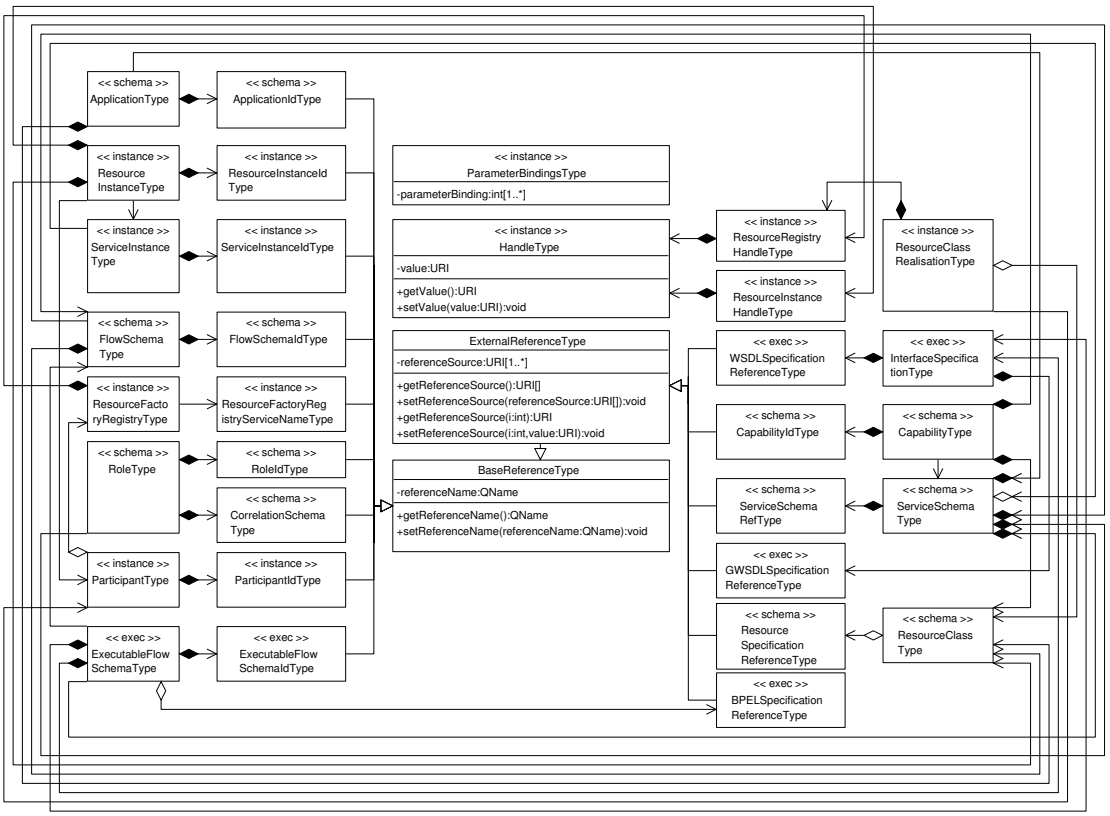

Abbildung 5.17.: Datenmodell für transiente E-Service-Meta-Informationen

Service-Schema-Spezifikation referenzieren. Den Ausgangspunkt bilden abstrakte Supertypen für die Referenzierung von Elementen mittels ihrem eindeutigen QName (`BaseReferenceType`) und im Zusammenhang mit ihrem Ort (`ExternalReferenceType`). Von `ExternalReferenceType` abgeleitete Typen repräsentieren die einzelnen XML-Dokumente der E-Service-Schema-Spezifikation. Spezialisierungen des `BaseReferenceType` repräsentieren hingegen Basiselemente, die entweder aus dem E-Service-PSM stammen oder zur Charakterisierung einer E-Service-Instanz dienen. Aufbauend auf den Repräsentationen von Dokumenten und Basiselementen werden Typen definiert, die Konzepte von E-Service-Schemata und -Instanzen repräsentieren und zueinander in Beziehung setzen. Diese Konzepte gehen entweder aus den Basiselementen selbst oder deren Auswertung hervor.

Bezüglich der E-Service-Schema-Repräsentation werden hier u. a. die Kontaktpunkte ≪`eService Interaction Contacts`≫ (`ApplicationType`) einer Rolle ≪`eService Role`≫ (`RoleType`) im E-Service-Modell (`ServiceSchemaType`) als *Ressourcenklasse* (`ResourceClassType`) zusammengefasst und mit einer Schnittstellenbeschreibung (`ResourceSpecificationReferenceType`) in Beziehung gesetzt. Auch ≪`eService Capability`≫-Elemente und deren Interaktionsmuster ≪`eService Capability Interaction Flow`≫ werden hier

mit einer entsprechenden Ressourcenklasse versehen. Daneben wird für jedes Interaktionsmuster auch noch dessen ausführbare Variante repräsentiert (`ExecutableFlowSchemaType`) und mit den dazugehörigen Spezifikationen des BPEL-Kompositionsschemas und dessen Schnittstellen in WSDL- und GWSDL-Format in Beziehung gesetzt. Die letzten Konzepte (≪`exec`≫) markieren den Übergang von der E-Service-Schema- (≪`schema`≫) zur Instanz-Repräsentation (≪`instance`≫), da sie während der Implementierung einer E-Service-Instanz aus dessen Schema generiert werden.

Die Repräsentation von E-Service-Instanzen steht im engen Zusammenhang mit dem Aggregationsvorgang. Das zentrale Konzept ist hier die E-Service-Instanz selbst(`ServiceInstanceType`). In deren Kontext werden Teilnehmer (`ParticipantType`) mit den Rollen des Schemas assoziiert. Dies basiert auf dem Konzept der korrelierten Assoziation. Rollen sind dabei mit einem Schema der zur Korrelation verwendeten Informationen verbunden (`CorrelationSchemaType`). Ein korrespondierender Typ (`ParameterBindingsType`) repräsentiert die Abbildung der Korrelationsinformationen auf Parameter von Operationen der zur Rolle gehörigen Ressourcenklasse. Der assoziierte Teilnehmer ist verpflichtet, diese Ressourcenklasse zu realisieren. Eine Realisierung (`ResourceClassRealisationType`) beinhaltet dabei deren Deployment sowie Allokation. Konstruktion und Deployment der Ressourcenklassen von Capabilities als E-Service Engines erfolgt in automatisierter Weise. Die damit verbundenen Konzepte und Elemente wurden schon im Zusammenhang mit der Repräsentation ausführbarer Interaktionsmuster mittels `ExecutableFlowSchemaType` erläutert. Die Allokation erfolgt auf Basis von Mechanismen und Standardvorgängen des zugrunde liegenden OGSI Grid Service-Komponentenmodells. Das diesbezügliche Basiskonzept ist die permanente Grid Service-Referenz (`HandleType`). Hierfür wird eine erste Spezialisierung (`ResourceInstanceHandleType`) in Bezug auf die Instanzen von Ressourcenklassen (`ResourceInstanceType`) vorgenommen. Eine zweite Verfeinerung (`ResourceRegistryHandleType`) bezieht sich auf spezifische OGSI Registries (`ResourceFactoryRegistryType`). Eine solche Registry muss von jedem Teilnehmer bereitgestellt werden, der eine Menge von Ressourcenklassen realisiert und für jede Ressourcenklasse eine OGSI Factory bereitstellen, mit der Instanzen erzeugt werden können.

Die erläuterte Klassenhierarchie gibt einen ausreichenden Überblick für die folgende Betrachtung der Interaktionen von Plattformkomponenten. Ihre Implementierung basiert auf einer XML-Schema-Definition, in der die zuvor beschriebenen Datenformate als XML-Schema-Typen definiert werden. Diese XML-Schema-Typen können in WSDL-/GWSDL-Schnittstellenbeschreibungen importiert werden und dienen dort als Grundlage für die Spezifikation von Nachrichten. Im Folgenden werden damit die GWSDL-Schnittstellen der ESMS-Plattformkomponenten spezifiziert und deren wechselseitiger Austausch von E-Service-Meta-Informationen beschrieben.

Plattformkomponenten und -Interaktion Die Plattformkomponenten eines FRESCO-ESMS dienen der integrierten E-Service-Entwicklung und Ausführung. Im Einzel-

nen sieht der ESMS-Entwurf drei Komponenten für das E-Service-Schema-Management, E-Service Engine-Management und E-Service-Aggregation vor. Bevor diese Komponenten im nächsten Abschnitt einzeln betrachtet werden, erfolgt zunächst deren Spezifikation im Rahmen der ESMS-Plattform. Grundsätzlich sind Plattformkomponenten spezialisierte ESMS-Komponenten, d. h. dass es sich dabei um Grid Service-Komponenten handelt, die von dem `FrescoPortType` abgeleitet werden. Die resultierenden Grid Service-Plattformkomponenten und ihre Interaktionsbeziehungen werden in Abbildung 5.18 als UML-Komponentendiagramm gezeigt. Tabelle 5.8 konkretisiert die Darstellung durch einen Überblick der GWSDL-Schnittstellen. Im Folgenden werden diese im Zusammenhang erläutert.

Konkret werden Plattformkomponenten als spezialisierte Anwendungskomponenten entworfen. Es wird nämlich antizipiert, dass bei besonders hohen Anforderungen an die Dynamik eines VDLP die Fachfunktionen der Plattformkomponente während der Ausführung und als Teil einer E-Service-Instanz verwendet werden müssen. Z. B. könnte es erforderlich sein, die dynamische Assoziation von korrelierten Rollen in den VDLP zu integrieren und zu automatisieren. Aus diesem Grund werden Plattformkomponenten von E-Service-Ressourcen abgeleitet (d. h. sie erweitern den `FrescoResource portType`) und können so in E-Service-Anwendungssysteme integriert werden.

Die Spezifikation der Komponenten auf Plattformebene bezieht sich auf spezifische Anteile von deren Fachfunktionen, die im Zuge der E-Service Entwicklung und Ausführung als integrierte, automatisierte und damit besonders agile Prozesse herausgestellt werden sollen. Dies sind insbesondere Vorgänge der Implementierung und Nutzung von E-Service-Instanzen, die im Kontext der Kontaktphase der VDL ablaufen. Hierzu gehören im Wesentlichen die (ggf. partielle) Assoziation von Teilnehmern sowie Detailentwurf, Konstruktion, Installation und Allokation von E-Service-Ressourcen. Im Folgenden wird der auf diesen funktionalen Anteilen basierende Kernprozess der ESMS-Plattform betrachtet. Die Ausführungen fokussieren an dieser Stelle die Plattformebene und automatisierte Interaktion bei der Kooperation zwischen ESMS-Komponenten. Spezifische Komponentenfunktionen, deren Nutzung durch einzelne VDPN-Rollen manuell und autonom erfolgt, werden hier wenn nötig nur erwähnt. Deren genauere Betrachtung erfolgt in den anschließenden Sektionen.

Die Implementierung einer E-Service-Instanz ist im FRESCO E-Service-Management-Vorgehensmodell auf Stufe 4 eingeordnet. An dieser Stelle wurden schon potenzielle Kunden und Produktionseinheiten als VDPN-Teilnehmer identifiziert. Die Zusammensetzung dieser Gruppe kann sich im Verlauf des DLP jedoch noch ändern. Das E-Service-Schema wurde ggf. im Zuge wechselseitiger Verhandlungen angepasst und liegt nun in seiner endgültigen Form vor. Der initiale Schritt der Implementierung besteht nun in der *Deklaration* einer E-Service-Instanz. Dieser Schritt erfolgt in manueller Weise durch den VDPN-Broker, der hierzu den Aggregation-Manager verwendet. Der Aggregation-Manager erfragt dabei das entsprechende E-Service-Schema beim Schema-Manager. Letzterer realisiert dazu in seinem `SchemaManagerPortType` eine Ope-

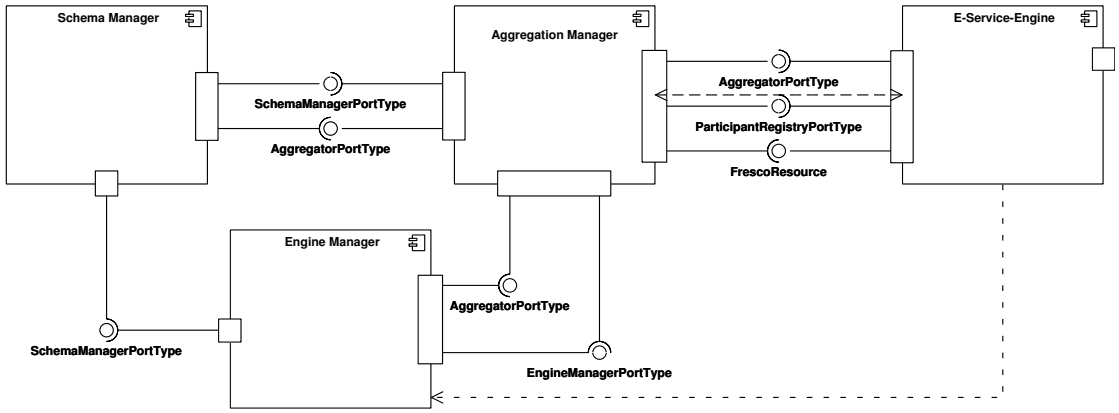

Abbildung 5.18.: FRESCO-TK E-Service-Plattform-Architektur

ration `getServiceSchema()`, die auf Basis eines QName des Schemas eine Referenz auf die persistente E-Service-Schema-Repräsentation (`ServiceSchemaType`) zurückliefert. In diesem Zusammenhang kann sich der Aggregation-Manager auch vom Schema-Manager über die Änderung eines E-Service-Schemas informieren lassen. Er muss sich dazu mittels dessen `registerSchemaChangeListener()`-Operation registrieren und gibt dabei den QName des Schemas sowie den GSH (`HandleType`) seines `AggregatorPortType` an. Der Schema-Manager ruft dann bei einer Änderung des entsprechenden Schemas die `serviceSchemaChanged()` Operation des Aggregation-Managers auf und übergibt eine Referenz auf das Schema (`BaseReferenceType`).

Im Zuge der Deklaration einer E-Service-Instanz bestimmt der Broker eine Aggregationsstrategie mit spezifischen Modellen für die Assoziation von Teilnehmern sowie die Installation und Allokation von deren Anwendungskomponenten. Die Strategie legt für alle Teilfunktionen Zeitpunkte (z. B. zu Beginn oder während des DLP) und Mechanismen (aktiv/passiv, manuell/automatisch) fest. Für die automatische passive Assoziation von Teilnehmern realisiert der Aggregation-Manager einen `ParticipantRegistryPortType`. Dieser erlaubt die Registrierung/Entfernung von Teilnehmern mittels der Operationen `registerParticipant()` und `removeParticipant()`. Teilnehmer werden hier mit einer eindeutigen Kennung (`ParticipantIdType`) referenziert und insbesondere mit einer lokalen OGSI Registry (`ResourceFactoryRegistryType`) in Beziehung gesetzt, in denen Factories und Instanzen der von ihnen bereitgestellten E-Service-Ressourcen zu finden sind. Die korrelierte Assoziation von registrierten Teilnehmern mit einer Rolle des E-Service-Schemas erfolgt mit der Operation `correlatedAssignment()`, wobei die Kennung der E-Service-Instanz (`ServiceInstanceType`) die Rolle (`RoleType`) und Korrelationsinformationen in Form einer Abbildungsforschrift (`ParameterBindingsType`) einer Parameterliste variablen Inhalts (`OGSI::ExtensibilityType`) auf das Korrelationsschema der Rolle sowie die Teilnehmer-Repräsentation (`ParticipantType`) anzugeben sind. Die korrespondierende Operation `correlatedAssignmentByParameterList()` erlaubt ein

GWSDL Element	Typ	Inhalt
FrescoRessource::AggregatorPortType	pt	Operationen des Aggregation-Managers zur Aggregationskontrolle
::correlatedAssociation	op	Anforderung zur Aggregation einer E-Service-Ressource
::correlatedAssociationByParameterList	op	dto. mit alternativem Format
::correlatedDisAssociation	op	Anforderung zur Freigabe einer E-Service-Ressource
::correlatedDisAssociationByParameterList	op	dto. mit alternativem Format
::serviceSchemaChanged	op	Rückruf bei Änderung eines E-Service-Schemas
::commitDeployment	op	Bestätigung des Deployments einer E-Service-Ressource
FrescoRessource::ParticipantRegistryPortType	pt	Operationen des Aggregation-Managers zur Teilnehmerverwaltung
::correlatedAssignment	op	Anfrage einer Rollenzuweisung
::correlatedAssignmentByParameterList	op	dto. mit alternativem Format
::registerParticipant	op	Registrierung eines Teilnehmers
::removeParticipant	op	Verwerfen eines Teilnehmers
FrescoRessource::EnginePortType	pt	Operationen des Engine-Managers zur Verwaltung von E-Service-Engines
::deploy	op	Anforderung zur Konstruktion und Installation einer E-Service-Engine
FrescoRessource::SchemaManagerPortType	pt	Operationen des Schema-Managers zur Verwaltung von E-Service-Schemas
::getExecutableFlowSchema	op	Anfrage eines BPEL-Schemas für einen eService Capability Interaction Flow
::registerSchemaChangeListener	op	Registrierung zur Benachrichtigung bei Änderung eines E-Service-Schemas
::getServiceSchema	op	Anfrage eines E-Service-Schemas

Tabelle 5.8.: FRESCO-TK-Plattformkomponentenschnittstellen

alternatives Parameterformat. Die Rücknahme einer einmal getätigten Zuweisung ist nicht möglich, um eine lückenlose Dokumentation des VDLP zu garantieren. Der Funktionsbereich des Teilnehmermanagements kann insbesondere auch in den VDLP integriert werden. In diesem Fall werden dessen Operationen dann durch eine E-Service Engine gesteuert.

Im weiteren Verlauf der Aggregation steuert der Aggregation-Manager je nach Strategie ggf. die Konstruktion und Installation von E-Service Engines (automatisches passives Deploymentmodell). In diesem Fall kooperiert der Aggregation-Manager mit dem Engine-Manager. Letzterer realisiert in seinem `EnginePortType` eine Operation `deploy()`, mit der die Implementierung einer E-Service Engine angewiesen werden kann. Hierzu gibt der Aggregation-Manager die Capability (`CapabilityType`) und den Teilnehmer (`ParticipantType`) vor. Der Engine-Manager leitet hieraus die verschiedenen zu erbringenden und zu nutzenden Schnittstellen für E-Service Engine Adapter und E-Service-Ressourcen-Proxy der E-Service Engine ab und generiert diese Teile. Ferner konfiguriert er eine gekapselte BPEL Engine mittels WSDL- und BPEL-Spezifikationen der einzelnen Interaktionsmuster der Capability. Die ausführbaren Interaktionsmuster werden durch den Schema-Manager aus dem E-Service-Schema generiert. Um sie zu erhalten, ruft der Engine-Manager die `getExecutableFlowSchema()` Operation des Schema-Managers auf. Dazu muss er die Repräsentation des Interaktionsmusters (`FlowSchemaType`) angeben und erhält die ausführbare Variante (`ExecutableFlowSchemaType`) zurück. Am Ende der Installation bestätigt der Engine-

Manager den Erfolg der Installation durch Aufruf der Operation `commitDeployment()` beim Aggregation-Manager und übergibt diesem eine Repräsentation der realisierten E-Service-Ressource (`ResourceClassRealisationType`).

Sobald mindestens eine E-Service Engine konstruiert und installiert wurde, kann die Ausführung der E-Service-Instanz beginnen. Zum Start der Ausführung erfolgt die Allokation der initialen E-Service Engine durch den Aggregation-Manager. Der Aggregation-Manager führt auch die erste Interaktion mit dieser Engine durch und startet dadurch die Ausführung eines ihrer BPEL-Prozesse. Von da an erfolgt die Steuerung des VDLP ausschließlich durch die E-Service Engine sowie weitere im Verlauf der Ausführung aktivierte Engines. Die Bindung der Engine an die Anwendungskomponente einer Endpunkt-/Rolle-Kombination, mit der laut E-Service-Schema eine Interaktion vorgesehen ist sowie die Bindung im Zuge überhaupt aller Interaktionen zwischen beliebigen Anwendungskomponenten, erfolgt stets durch Konsultation des Aggregation-Managers. Dessen primärer `AggregatorPortType` beinhaltet dazu die Operation `correlatedAssociation()`. Diese benötigt als Parameter die E-Service-Instanz (`ServiceInstanceType`), die Ressourcenklasse (`ResourceClassType`) und Rolle (`RoleType`) des Interaktionspartners sowie Korrelationsinformationen in Form der Abbildungsvorschrift (`ParameterBindingsType`) einer Parameterliste variablen Inhalts (`OGSI::ExtensibilityType`) auf das Korrelationsschema der Rolle. Das Ergebnis ist eine Repräsentation der zuständigen Komponenteninstanz (`ResourceInstanceType`), die den benötigten Endpunkt für die Interaktion realisiert. Diese Repräsentation beinhaltet insbesondere den GSH des `FrescoRessource portType`, anhand dessen die Bindung erfolgen kann. Der Effekt eines Aufrufs von `correlatedAssociation` ist der, dass der Aggregation-Manager alle notwendigen Schritte unternimmt, um für die angefragte Ressourcenklasse/Rolle-Kombination eine Grid Service-Instanz zu erzeugen. Im einfachsten Fall ist diese Instanz schon vorhanden. ggf. kann dies aber auch die Zuweisung der E-Service-Rolle an einen Teilnehmer sowie die Konstruktion und Installation der Anwendungskomponente einschließen. Umgekehrt beinhaltet der `AggregatorPortType` auch eine Operation `correlatedDisAssociation()`, die die Zerstörung der Grid Service-Instanz und damit die Freigabe der damit verbundenen Ressourcen bewirkt. Für beide Operationen existiert eine alternative Variante mit alternativem Parameterformat.

Mit den geschilderten globalen Funktionen und kooperativen Interaktionsmustern lassen sich die kritischen Abschnitte des FRESCO E-Service-Management-Vorgehensmodells teilweise oder bei entsprechender Aggregationsstrategie auch komplett automatisieren. Hierdurch wird die geforderte Agilität des E-Service-Entwicklungsvorgangs erreicht. Die Entwicklungsdauer für individuelle E-Service-Instanzen stellt bei der Bildung von VDU praktisch kein Hindernis dar.

Plattformkomponentenarchitektur und -implementierung Auch zur Implementierung von Plattformkomponenten stellt das FRESCO-TK eine generelle Architektur

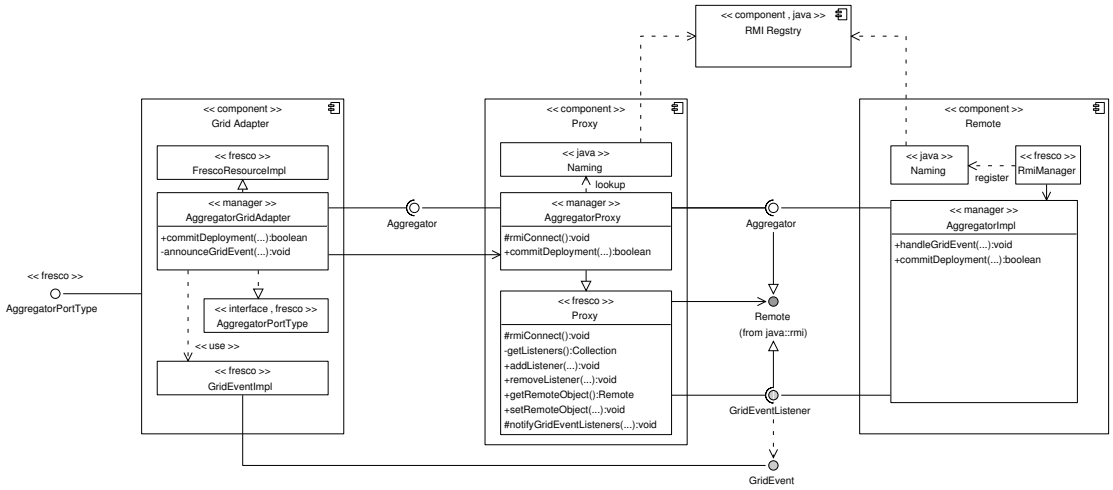

Abbildung 5.19.: FRESCO-TK Grid Connectivity-Architektur

und ein Software-Rahmenwerk bereit. Zum einen regelt die Grid Connectivity-Architektur für Plattformkomponenten das grundsätzliche Vorgehen bei der Implementierung der Grid Service-Schnittstellen. Zum anderen werden durch die XPDL Core-Bibliothek generische Funktionen zum Umgang mit persistenten E-Service-Schema-Repräsentationen in Form von XPDL-Spezifikationen vorgegeben.

Die Implementierung von Grid Service-Komponenten erfolgt in FRESCO-TK auf Basis der vom GT3 Core realisierten Container-Architektur. Hierin werden Grid Service-Schnittstellen mittels eines Software-Rahmenwerks implementiert und dann in der Laufzeitumgebung des Containers ausgeführt. Die vom GT3 Core realisierte Laufzeitumgebung ist jedoch nicht dafür ausgelegt, komplexeAnwendungssysteme auszuführen, die ggf. Funktionen einer Schnittstelle realisieren. Solche Systeme sollten stattdessen autonom ablaufen und in einer optimierten Laufzeitumgebung installiert werden. Dies gilt auch für die FRESCO-TK Plattformkomponenten, die umfangreiche Anwendungssysteme darstellen und zudem über alternative Schnittstellen zur interaktiven Nutzung verfügen. Die Implementierung dieser Komponenten als autonome Anwendungssysteme erfolgt mithilfe einer fundamentalen Verbindungsarchitektur auf Basis separater Java-VM.

Die Grundzüge dieser *Grid Connectivity-Architektur* werden in Abb. 5.19 für das Beispiel der Aggregation-Manager-Plattformkomponente gezeigt. Hierin werden die drei konzeptionellen Komponenten `Grid Adapter`, `Proxy` und `Remote` unterschieden. Der Grid Adapter implementiert die Grid Service-Komponente auf Basis des gleichen Software-Rahmenwerks, das schon weiter oben im Kontext der E-Service-Ressourcen-Architektur beschrieben wurde durch Erweiterung von `FrescoResourceImpl`. Dies erfolgt derart, dass alle Aufrufe an die `Aggregator`-Schnittstelle des Anwendungssystems weitergeleitet werden, das in einer externen Java-VM abläuft. Die `Aggregator`-Schnitt-

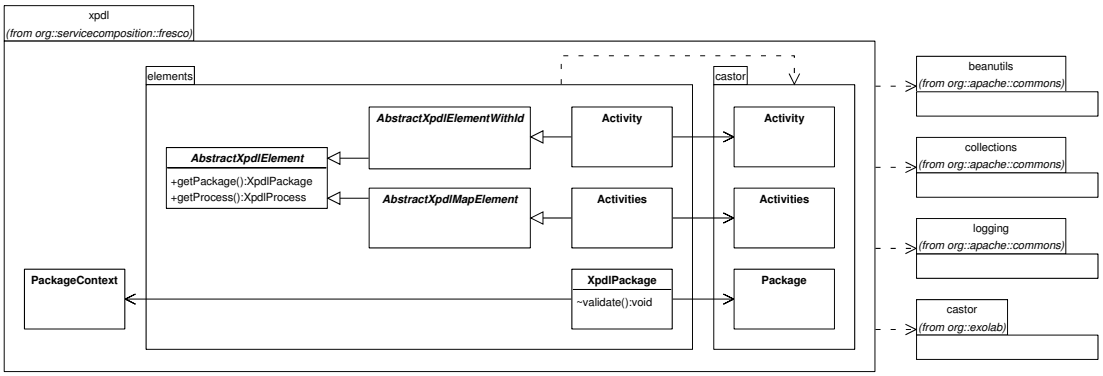

Abbildung 5.20.: Architekturkonzept des XPDL Core

stelle besteht aus den spezifischen Methoden des `AggregatorPortType` der Grid Service-Komponente. Es muss keinerlei Konvertierung vorgenommen werden. Die Anbindung des Anwendungssystems erfolgt über die Proxy-Komponente, die sich ebenfalls im Grid Container befindet. Zu deren Realisierung wird die abstrakte Klasse `Proxy` in Bezug auf die konkrete Plattformkomponente erweitert. Sie bietet u. a. Grundfunktionen zur Benachrichtigung des Anwendungssystems über Ereignisse im Grid Container. Solche Nachrichten werden durch die `GridEvent`-Schnittstelle modelliert, für die auch eine Implementierung `GridEventImpl` bereitsteht. Ereignisse werden im Grid Adapter als `GridEvent` erzeugt und per `notifyGridEventListeners()` an den Proxy gemeldet, der sie daraufhin an das entfernte AS weiterleitet. Zur Kontaktaufnahme mit dem Anwendungssystem muss Remote sowohl die spezifische `Aggregator`-Schnittstelle als auch die generische `GridEventListener`-Schnittstelle implementieren. Beide Schnittstellen erweitern Remote und können daher per RMI bei einem entfernten Objekt der Klasse `AggregatorImpl` aufgerufen werden, die `UnicastRemoteObject` erweitert. Der Aggregator-Proxy verschafft sich eine entsprechende Referenz per `lookup()` in einer festgelegten RMI Registry. Bei der gleichen Registry hat Remote zuvor `AggregatorImpl` registriert. Eine derartige Verbindung erfolgt bei Aktivierung des Grid Adapter durch den GT3 Core Container. Dann wird Remote auch als Empfänger von `GridEvents` registriert und über den Status des Grid Adapter informiert, der nach Ablauf seiner Lebenszeit zerstört wird. Das autonome Anwendungssystem läuft dann weiter und kann bei Aufrufen durch andere Grid Adapter erneut angesprochen werden. Zur Implementierung einer individuellen Plattformkomponente im FRESCO-TK müssen nur die Anteile erstellt werden, die mit deren spezifischen Schnittstellen verbunden sind (≪`manager`≫). Alle anderen Anteile werden entweder vom FRESCO-TK (≪`fresco`≫) oder vom Java RMI Subsystem (≪`java`≫) bereitgestellt.

Verschiedene Plattformkomponenten der ESMS-Plattform müssen zur Implementierung von Fachfunktionen eine Analyse, Manipulation oder Änderung persistenter E-Service-Schema-Repräsentationen vornehmen. Konkret wird eine Möglichkeit benö-

tigt, um XPDL-Dokumente in den Speicher zu laden, zu validieren, auf verschiedene Art zu traversieren und zu durchsuchen, ggf. zu ändern und schließlich als XML-Dokumente zu serialisieren. Der gängige Weg zur Implementierung solcher Funktionen ist die Verwendung eines XML-Bindungsrahmenwerks wie JAXB (Java Architecture for XML Binding, `http://java.sun.com/webservices/jaxb/`) oder Castor. Sie ermöglichen die Generierung einer Klassenhierarchie, die die XML-Struktur abbildet und Methoden zum typsicheren Zugriff auf deren Elemente bereitstellt. Die Grundfunktionen der Bindungsrahmenwerke reichen jedoch nicht aus, denn hier fehlen essenzielle Möglichkeiten in Bezug auf Traversierung und Selektion im XPDL-Dokumentbaum. Insbesondere müssen hier die `ExternalPackage`-Beziehungen zwischen einzelnen Dokumenten berücksichtigt werden, z. B. um alle sichtbaren `Application`-Elemente eines `WorkflowProcess` zu erfragen.

Funktionen und Eigenschaften zur Handhabung der XPDL-Spezifikationen von E-Service-Schemata werden im FRESCO-TK durch die Klassenbibliothek XPDL Core [BP04, S. 96 f.] realisiert. Der XPDL Core basiert grundsätzlich auf einer mittels Castor generierten Klassenhierarchie für XPDL. Deren Klassen werden jedoch mittels leichtgewichtiger Wrapper-Klassen gekapselt, die die benötigte Zusatzfunktionalität als JavaBeans bereitstellen. Abbildung 5.20 zeigt das Architekturprinzip des XPDL Core für das Beispiel von `Activity`-Elementen. Alle Elemente erweitern hier die abstrakte Klasse `AbstractXpdlElement`, die einen `Container` mit Methoden zum direkten Zugriff auf Prozess `getProcess()` und Paket `getPackage()` implementiert. Hiervon sind abstrakte Unterklassen abgeleitet, die spezielle Zugriffsmethoden für Elemente mit eindeutiger Referenz (`AbstractXpdlElementWithId`) oder Aggregationsfunktion (`AbstractXpdlMapElement`) beten. Beispiele für solche Elemente sind `Activity` und `Activities`. Die Berücksichtigung von Beziehungen zwischen `XPDLPackage`-Elementen erfolgt durch einen gemeinsamen `PackageContext`. Weitere Funktionen des XPDL Core umfassen u. a. erweiterte Zugriffsmöglichkeiten (z. B. in Bezug auf Name, Index) auf Element-Eigenschaften auf Basis von Apache Commons-beanutils und die Validierung von `XPDLPackage`-Elementen (`validate()`).

5.3.4. E-Service-Management-Fachfunktionen

Der verbleibende Teil der ESMS-Implementierung betrifft die ESMS-Fachfunktionen. Der Begriff Fachfunktion bezieht sich dabei auf Vorgänge, die sich direkt oder indirekt auf die Anwendungsdomäne beziehen. In Bezug auf die Anwendungsdomäne der Dienstleistungsproduktion umfassen die durch das ESMS bereitgestellten Fachfunktionen die verschiedenen Vorgänge im Entwicklungslebenszyklus eines E-Service zur Planung und Steuerung der virtuellen Produktion von Dienstleistungen.

Die folgenden Sektionen fokussieren die Realisierung konkreter Vorgänge im FRESCO E-Service-Management-Vorgehensmodell und setzen sie zu den am Anfang des Kapitels entworfenen Methoden der E-Service-Entwicklung und -Ausführung in Beziehung. Dies beginnt in Sektion 5.3.4.1 mit dem E-Service-Entwurf durch eine

interaktive grafische Modellierungsumgebung, die die Erstellung von E-Service-PIM und -PSM unterstützt und deren Transformation in E-Service-Schema-Spezifikationen erlaubt. In Sektion 5.3.4.2 folgt eine Komponente, die sowohl interaktive als auch programmatische Methoden zur Handhabung solcher E-Service-Schema-Spezifikationen im Entwicklungslebenszyklus realisiert, was insbesondere ihre strukturelle Validierung, ihren organisationsübergreifenden Zugriff, ihre regelbasierte Änderung sowie ihre Transformation in das Ausführungsformat BPEL betrifft. In Bezug auf die Implementierung von E-Service-Instanzen wird in Sektion 5.3.4.3 die automatische Konstruktion und Installation von E-Service Engines durch die Komponente des Engine-Managers erläutert. Abschließend erfolgt in Sektion 5.3.4.4 eine Betrachtung der Aggregation-Manager-Komponente, die während der gesamten Kontaktphase der VDL die Beziehungen zwischen VDPN-Teilnehmern und ihren Ressourcen verwaltet.

5.3.4.1. FRESCO-TK E-Service-Entwurfswerkzeug

Der E-Service-Schema-Entwurf als initialer FSMV-Schritt wird im FRESCO-TK durch ein interaktives E-Service-Entwurfswerkzeug realisiert. Dessen Implementierung basiert auf dem generischen UML-Modellierungswerkzeug Poseidon [Gen06]. Über die standardmäßigen Erweiterungsmöglichkeiten der UML in Form von Profilen hinaus, bietet Poseidon die Möglichkeit zur Erweiterung der Werkzeugfunktionalität auf Basis eines Plugin-Mechanismus. Im FRESCO-TK wird der Poseidon Plugin-Mechanismus verwendet, um die Modellierung von E-Service-PIM und -PSM auf Basis des entworfenen UML-Profils zu ermöglichen. Des Weiteren wird das Poseidon Software-Rahmenwerk zur Entwicklung eines Codegenerators verwendet, um die UML-Repräsentation von E-Service-PSM in eine textuelle Repräsentation auf Basis von XPDL zu übersetzen. Diese bildet zusammen mit verschiedenen Schnittstellenspezifikationen die Basis für E-Service-Schema-Spezifikationen im FRESCO-TK. Nun wird der Modellentwurf, die Generierung von XPDL-Spezifikationen und deren Zusammenstellung zu E-Service-Schema-Spezifikationen skizziert.

Grafischer E-Service-Modellentwurf Poseidon for UML [Gen06] ist ein Werkzeug zur grafischen Modellierung objektorientierter Systeme im UML-Standard und zum anschließenden Export des grafischen Modells in verschiedene textuelle Repräsentationen (z. B. OO-Programmiersprachen wie Java). Poseidon basiert auf der Open-Source-Software ArgoUML[38] in der Version 8.0 und erweitert diese in Bezug auf verschiedene Funktionen und in Hinblick auf den produktiven Einsatz. Poseidon ist vollständig in der Programmiersprache Java realisiert und kann somit auf verschiedenen Plattformen eingesetzt werden.

Zur interaktiven Modellierung bietet Poseidon eine differenzierte grafische Benutzeroberfläche, die in Abb. 5.21 gezeigt wird. Die Oberfläche entspricht von ihrem

[38] `http://sourceforge.net/projects/argouml`.

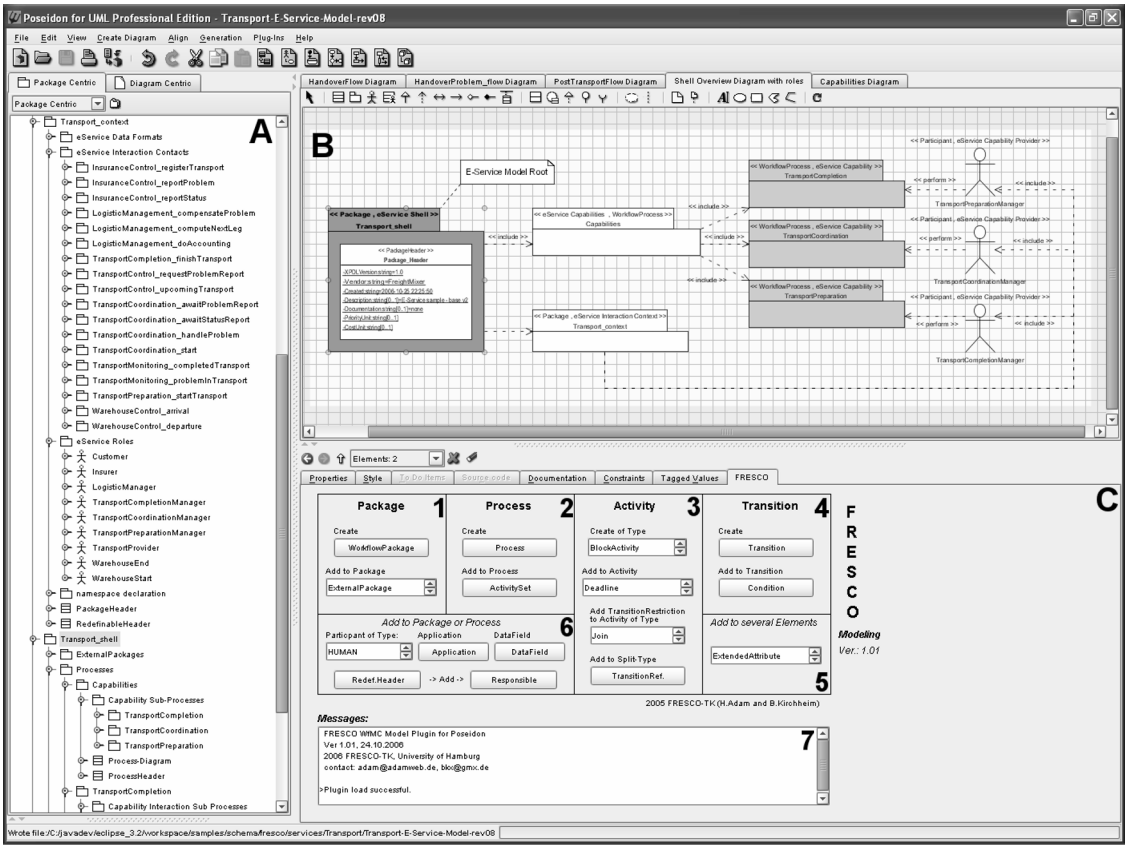

Abbildung 5.21.: Benutzerschnittstelle des FRESCO-Entwurfswerkzeugs

grundsätzlichen Aufbau dem gängigen Standard komplexer GUI-AS, bei denen Grundfunktionen in Menüs und Symbolleisten zugreifbar sind und auf verschiedene Bereiche mit alternativen Sichten angewendet werden können. Die wesentlichen Bereiche des GUI-Systems beinhalten Modellbereich (A), Diagrammbereich (B) und Detailbereich (C). Der Modellbereich zeigt das logische UML-Gesamtmodell in einer Baumstruktur mit alternativen Sortieroptionen. Der Diagrammbereich zeigt verschiedene grafische Perspektiven auf das logische Modell mittels UML-Diagrammen. Der Detailbereich zeigt die logischen und grafischen Eigenschaften von Modellelementen und erlaubt deren Modifizierung. Hierbei werden verschiedene Aspekte in überlagerten Bereichen behandelt.

Die Erweiterung von Poseidon in Bezug auf den Entwurf von E-Service-Modellen basiert auf Methoden zur Erstellung, Modifikation und Transformation spezifischer UML-Modelle zur Modellierung von FRESCO E-Services. Solche Modelle basieren auf den strukturellen Vorgaben des FRESCO E-Service-Metamodells und dessen Abbildung auf UML-Elemente durch das FRESCO UML-Profil. Die fundamentalen strukturellen Vorgaben werden dabei durch das WfMC-konforme Workflow-Meta-

modell vorgegeben. Dementsprechend stellen die Basisfunktionen des FRESCO-TK-Modellierungswerkzeugs Methoden bereit, die prinzipiell für den Entwurf beliebiger WfMC-konformer Workflows verwendet werden können [AK05]. Darauf bauen die Strukturen höherwertiger Metamodelle auf, deren Elemente mit zusätzlichen Stereotypen gekennzeichnet werden. Dies zeigt sich in Abb. 5.21 u. a. am Beispiel der ≪`eService Capabilities`≫ auf Basis eines ≪`WorkflowProcess`≫.

Praktisch zeigt sich das Poseidon-Plugin zunächst durch einen zusätzlichen Detailbereich `FRESCO`. Dieser Bereich stellt grafische Kontrollelemente für die Erzeugung der verschiedenen Elemente des Workflow-Metamodells bereit. Dies beinhaltet Elemente im Zusammenhang von Workflow-Paketen (1,6), -Prozessen (2,6), -Aktivitäten (3), -Transitionen (4) sowie generische Elemente (5). Die Erzeugung von Elementen des Workflow-Metamodells beinhaltet in den meisten Fällen die Erzeugung und Vernetzung einer Reihe von assoziierten UML-Elementen. So werden nach Vorgabe des UML-Profils für viele Arten von Elementen übergeordnete UML-Pakete zu deren Zusammenfassung erstellt. Einige andere Elemente, die als UML-Paket abgebildet werden, erhalten zudem eine zusätzliche Klasse zur Spezifikation von Eigenschaften. Die entstehenden Modelle bestehen aus einer Menge fundamentaler Pakete, die in sich jeweils eine Baumstruktur aufweisen. Die Erzeugung von Elementen erfolgt dann immer in Bezug auf einen spezifischen Knoten innerhalb eines der Modellbäume. Der Zielknoten ist dabei vor der Verwendung eines Kontrollelements des `FRESCO`-Bereichs im allgemeinen Modellbereich auszuwählen. Wird hierbei ein im Sinne des Metamodells unpassender Knoten gewählt, erfolgen Warnhinweise in einem Statusfeld (7). Bei korrekter Auswahl werden die erzeugten UML-Elemente in den Modellbaum eingefügt und im allgemeinen Modellbereich gezeigt. Nur für die Prozessstruktur erfolgt eine automatische Visualisierung durch Aktivitätsdiagramme. Dazu werden Aktivitäten und Transitionen des Workflow-Metamodells auf die korrespondierenden UML-Elemente abgebildet. Zusätzlich erfolgt eine Repräsentation aller Details auf Basis korrespondierender Pakete und Klassen, die per UML-Abhängigkeit verbunden sind. Ergänzende Diagramme können nach Belieben hinzugefügt werden.

Zur Implementierung der beschriebenen Funktionalität mit Hilfe eines Plugin stellt Poseidon ein Java-basiertes Software-Rahmenwerk zur Verfügung. Die resultierende Plugin-Architektur ist in Abb. 5.22 skizziert. Zentraler Bestandteil ist die von `PoseidonModuleInstall` abgeleitete Klasse `WFMCPlugin`, die eine Kontrolle des Ladevorgangs und der Lizenzverwaltung der Erweiterung durch Poseidon ermöglicht. Von hier aus werden auch die beiden wesentlichen Anteile des Plugin kontrolliert: dessen spezifischer Detailbereich und der XPDL-Codegenerator. Beim Laden des Plugin wird der Codegenerator mittels `XPDLCodeGenInstall` in das allgemeine Software-Rahmenwerk eingebunden und steht dann als Option der zentralen Codegenerator-Funktionen von Poseidon zur Verfügung. Der zur Erweiterung gehörige Detailbereich basiert auf einer Erweiterung `WFMCModelPane` der Klasse `TabSpawnable`. Auf Basis dieses Java Swing `JPanel` kann eine freie Gestaltung der GUI für Funktionen des Plugin erfolgen. Diese Funktionen bewirken im Wesentlichen eine Manipulation des Modells.

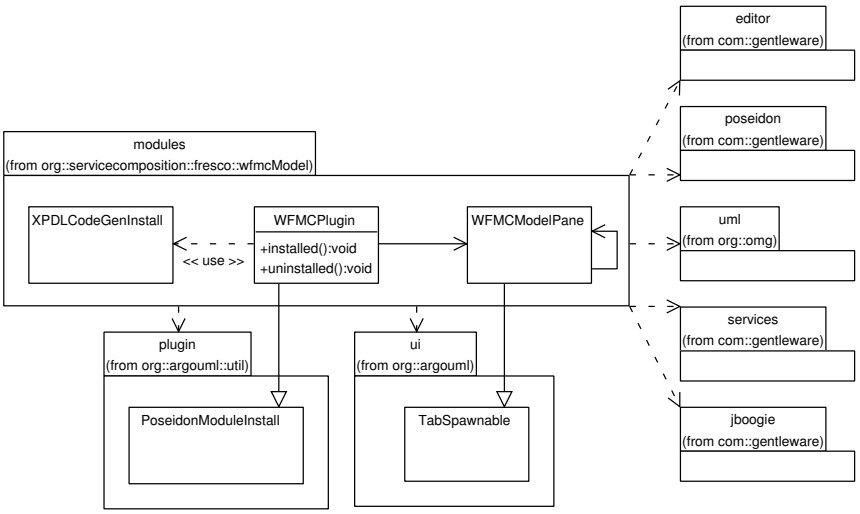

Abbildung 5.22.: UML-Klassendiagramm des FRESCO-TK Poseidon-Plugins

Hierzu stellt Poseidon verschiedene API mit entsprechenden Funktionen zur Verfügung. Dies umfasst zunächst die Repräsentation der UML 1.4-Elemente. Daneben bietet Poseidon eine API zur Manipulation des zu entwerfenden Modells sowohl in Bezug auf das logische Modell und dessen Darstellung im Modellbereich sowie das grafische Modell und dessen Darstellung im Diagrammbereich. Mittels dieser Schnittstellen werden die einzelnen Kontrollelemente des erweiterten Detailbereichs zur Konstruktion spezifischer UML-Modelle implementiert.[39]

Generierung von XPDL-Spezifikationen Eine entscheidende Eigenschaft des Modellierungswerkzeugs ist dessen Fähigkeit zur Codegenerierung. Die Notwendigkeit zur rapiden Abwicklung des E-Service-Entwicklungsprozesses im Kontext von VDU führte zum Entwurf einer Methodologie im Sinne modellgetriebener Entwicklung. Hierbei werden FRESCO E-Service-Modelle im Zuge des Entwicklungsprozesses in eine XPDL-konforme Repräsentation überführt. Die Automatisierung dieses Schritts ist für die Agilität der Methodik von entscheidender Bedeutung. Poseidon unterstützt dies durch ein Software-Rahmenwerk zur Entwicklung spezialisierter Codegeneratoren. Im FRESCO-TK wird von dieser Möglichkeit durch Implementierung eines XPDL-Codegenerators für das FSM und insbesondere dessen Workflow-Metamodell Gebrauch gemacht.

Der implementierte Codegenerator erzeugt XPDL-Repräsentationen für Teilbäume im E-Service-Modell. Der Referenzpunkt für eine solche XPDL-Repräsentation ist der obligatorische `PackageHeader` des Wurzelpakets. Auf dieser Basis können E-Service-Modelle explizit in eine Menge von XPDL-Repräsentationen überführt werden, wobei

[39] Weitere Details zur Implementierung finden sich in [AK05].

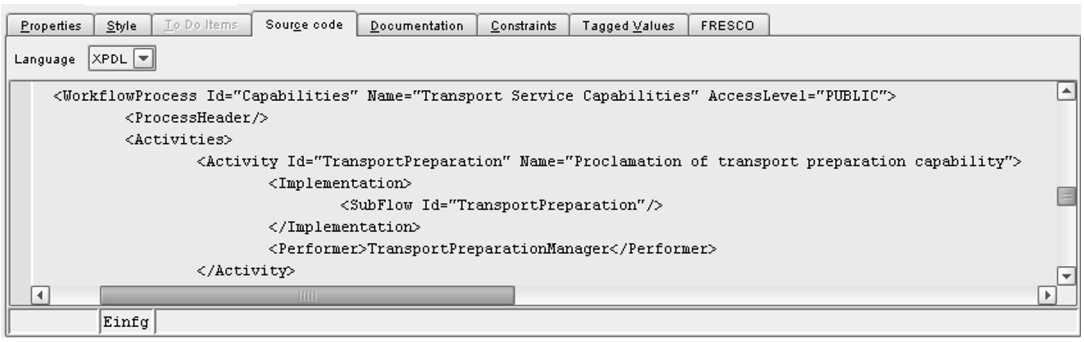

Abbildung 5.23.: Code-Vorschau im FRESCO-Entwurfswerkzeug

für jeden Teilbaum ein eigenständiges XPDL-Dokument generiert wird. Darüber hinaus findet eine implizite Generierung von ggf. partiellen XPDL-Repräsentationen statt, die schon während des Entwurfs jederzeit angezeigt werden können. Abb. 5.23 zeigt den entsprechenden Detailbereich der Poseidon-GUI für einen Ausschnitt des XPDL Codes einer E-Service-Shell.

Eine Besonderheit der expliziten Generierung ist die damit verbundene syntaktische Code-Validierung auf Basis von Konformitätstests der resultierenden XML-Dokumente. Hierzu wurde das XML-Schema von XPDL im Hinblick auf die ausformulierte XPDL-Spezifikation des WfMC vervollständigt. Dabei wurden verschiedene Bedingungen u. a. in Bezug auf die Verwendung eindeutiger Bezeichner für Attribute, die der Referenzierung von Elementen dienen, hinzugefügt. Der Konformitätstest generierter XPDL-Repräsentationen gegen das erweiterte Schema deckt logische Modellierungsfehler auf, bevor diese im Verlauf der E-Service-Entwicklung zu Folgefehlern führen.

Zur Implementierung des Codegenerators wurde das Software-Rahmenwerk von Poseidon verwendet. Dieses Rahmenwerk basiert auf der Velocity Template Engine von Apache [Apa06]. Diese gibt ein Format für textuelle Schablonen vor und liefert ein Software-Rahmenwerk zu deren automatischer Vervollständigung durch Daten eines Anwendungssystems. Die Präsentationslogik wird dabei weitgehend als Teil der Schablonen implementiert, die verschiedene Kontrollstrukturen und Zugriffe auf Java-Objekte beinhalten können. Im Fall von Poseidon werden die Schablonen mit UML-Modelldaten gefüllt und beziehen sich meist auf textuelle Artefakte eines Programmiermodells (z. B. Java-Klassen). Ein neuer Codegenerator wird im Wesentlichen durch Schablonen für das Zielformat und einen Steuermechanismus zu deren Aufruf implementiert.

Abb. 5.24 zeigt die Architektur für den Steuermechanismus des XPDL-Codegenerators. Der Mechanismus wird durch die Klasse `XPDLCodeGenInstall` mittels der Poseidon API und über den `CodeGenerationConnector` in die globalen Codegenerator-Funktionen eingegliedert. U. a. werden dabei die Steuerung des Generierungsprozesses sowie Eigenschaften für dessen interaktive Bedienung festgelegt. In Bezug auf die

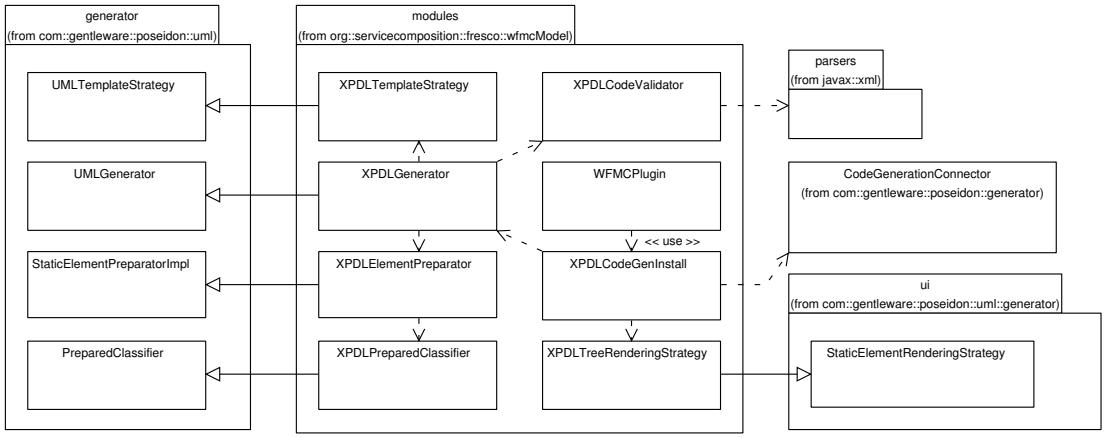

Abbildung 5.24.: UML-Klassendiagramm des FRESCO-TK XPDL-Codegenerators

interaktive Bedienung der expliziten Generierung wird mittels der abgeleiteten Klasse `XPDLTreeRenderingStrategy` die Darstellung eines Auswahlbaums definiert, mit dem der Benutzer bestimmen kann für welche Teile des Modells Code generiert werden soll. Die ebenfalls abgeleitete Klasse `XPDLGenerator` bestimmt die eigentliche Codegenerierung. Hier werden die Aufbereitung und der Zugriff von Informationen der UML-Modellelemente festgelegt (`XPDLElementPreparator`, `XPDLPreparedClassifier`) und die Schablonen zu deren Aufnahme bestimmt (`XPDLTemplateStrategy`). Ein wesentlicher Anteil der Implementierung ist die Aufbereitung von Klassenelementen. Genauer bezieht sich dies auf das spezielle Klassenelement des `PackageHeader`, das als Ansatzpunkt der Codegenerierung gewählt wurde. Die abgeleitete Klasse `XPDLPreparedClassifier` stellt in diesem Sinne Methoden bereit, um Knoten in Teilbäumen eines E-Service-Modells zu suchen, die mit spezifischen XPDL-Elementen korrespondieren. Ferner werden Methoden bereitgestellt, um aus den Knoten die Inhalte korrespondierender XPDL-Elemente abzuleiten. Diese Inhalte werden durch eine Schablone als textuelle XPDL-Elemente dargestellt. Der Ausschnitt der Schablone in Listing 5.6 deutet deren Aufbau an.

Die XPDL-Schablone lehnt sich eng an die Struktur von XPDL-Dokumenten an. Hierbei werden statische Informationen wie u. a. die Einleitung in Form von stets konstanten Namensraumdeklarationen sowie die festgelegten Namen von Elementen und Attributen mitsamt der zugehörigen Klammerung fest in der Schablone kodiert. Innerhalb der Elemente erfolgt für variable Anteile eine Referenzierung von Java-Objekten, die den Zugriff auf Informationen aufbereiteter Modellelemente ermöglichen. Dies erfolgt z. B. durch Aufruf von `preparedModelElement.packageHeaderCode()` für die Angaben im `PackageHeader`. Zudem wird der Name des XPDL-Dokuments mittels `preparedModelElement.getFilename()` aus dem Paketnamen abgeleitet. Die Bearbeitung der Schablone ist sequenziell und durch Kontrollstrukturen reglementiert. Die Reihenfolge der Elemente richtet sich grundsätzlich nach der hierarchisch-sequenziel-

```
  #if ($preparedModelElement.getStereotypeAsString() == "PackageHeader")
  $fileNameHolder.setFileName("$preparedModelElement.getFilename()")
  <?xml version="1.0" encoding=üs-ascii"?>
  <Package $preparedModelElement.packageCode()
      xmlns="http://www.wfmc.org/2002/XPDL1.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.wfmc.org/2002/XPDL1.0
      http://wfmc.org/standards/docs/TC-1025_schema_10_xpdl.xsd">
  ## ******************** PackageHeader ********************
  <PackageHeader>
  $preparedModelElement.packageHeaderCode()
  </PackageHeader>
...
```

Listing 5.6: Schablone zur XPDL-Generierung (Ausschnitt)

len Anordnung, die im XPDL-Schema spezifiziert wird. Zusätzlich werden für Mengen typgleicher Elemente (z. B. Aktivitäten, Datenfelder etc.) Schleifen vorgegeben. Die Bedingung am Anfang der Schablone stellt sicher, dass die Codegenerierung immer relativ zu einem `PackageHeader` im Modell erfolgt.[40]

E-Service-Schema-Spezifikation E-Service-Schemata sind abstrakte Modelle, aus denen beliebig viele konkrete E-Service-Instanzen mit identischer VDLP-Struktur abgeleitet werden können. In der modellgetriebenen FRESCO-Methodik zur E-Service-Entwicklung gehen E-Service-Schemata aus dem Entwurf des plattformspezif. Modells eines E-Services in UML hervor. Dies erfolgt primär durch Generierung einer XPDL-Repräsentation. Diese XPDL-Repräsentation wird dabei in verschiedene Teile untergliedert, die konzeptionell durch das FSM bestimmt sind und sich praktisch aus den hierarchischen Anteilen eines PSM in Bezug auf verschiedene Workflow-Pakete ergeben. Hinzu kommen dann noch verschiedene Schnittstellenspezifikationen der Web Service-Komponenten und Kompositionen, die im PSM als Kontaktpunkte referenziert werden und praktisch mittels beliebiger WS-Entwicklungswerkzeuge erzeugt werden können. Alle Teile zusammen ergeben eine E-Service-Schema-Spezifikation, deren Struktur in Abb.5.25 dargestellt wird.

Die verschiedenen Teile einer E-Service-Spezifikation sind als XML-Dokumente gegeben. Das zentrale Dokument `service_shell.xpdl` beinhaltet die XPDL-Repräsentation der E-Service-Shell (≪`eService Shell`≫). Hierin werden u. a. Capabilities und deren Interaktionsmuster deklariert, wobei Elemente (Rollen und Interaktionsprozesse) in anderen XPDL-Dokumenten referenziert werden. Diese XPDL-Dokumente werden als ≪`externalPackages`≫ referenziert. Mit jeder deklarierten Capability geht zudem eine Schnittstellenspezifikation der dazugehörigen Anwendungskomponente auf Plattformebene einher, die als GWSDL-Dokument (`capability.gwsdl`) beigefügt werden muss. Die Deklaration der Capability und aller weiteren Kontaktpunkte von Assets und

[40] Weitere Details zur Implementierung finden sich in [AK05].

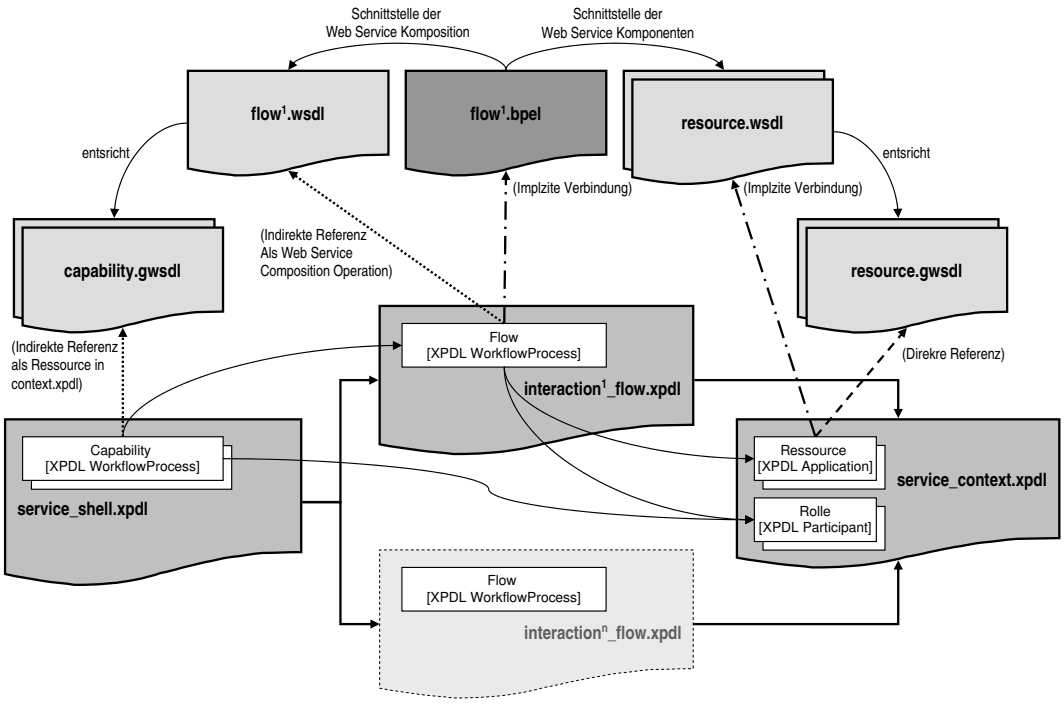

Abbildung 5.25.: Struktur einer E-Service-Schema-Spezifikation

Demands als Anwendungskomponenten mit zugehöriger Schnittstellenbeschreibung erfolgt im ≪`eService Interaction Context`≫, der in einem separaten XPDL-Dokument (`service_context.xpdl`) spezifiziert wird. Des Weiteren werden hier einheitliche Rollen (≪`eService Role`≫) und Datentypen (≪`eService Data Format`≫) deklariert. Der E-Service-Kontext wird auch in die XPDL-Spezifikationen der verschiedenen Interaktionsprozesse von Capabilities (`interaction_flow.xpdl`) als ≪`externalPackage`≫ eingebunden. Hier werden die Rollen, Kontaktpunkte und Datentypen zur Spezifikation jeweils genau eines ≪`eService Capability Interaction Flows`≫ verwendet. Um die Konstruktion von Anwendungskomponenten für Capability-Spezifikationen in Form von E-Service Engine-Komponenten zu ermöglichen, werden für jeden Interaktionsprozess noch Spezifikationen der von diesem bereitgestellten (`flow.wsdl`) und verwendeten (`resource.wsdl`) Schnittstellen in SOC-Format benötigt. Bei der automatischen Konstruktion wird dann für jeden Interaktionsprozess ein BPEL-Schema (`flow.bpel`) generiert und dessen Schnittstelle (`flow.wsdl`) mit BPEL-spezifischen Erweiterungen ergänzt.

Alle Dokumente zusammen werden in einem E-Service-Schema-Spezifikationsarchiv zusammengefasst. Hierbei gelten Konventionen für die Archivstruktur, die Benennung von Dokumenten sowie die Verwendung von konsistenten Namensräumen innerhalb der Teilspezifikationen.[41]

[41] Struktur und Konventionen von E-Service-Schema-Spezifikationsarchiven zusammen mit

5.3.4.2. FRESCO-TK E-Service-Schema-Manager

Der FRESCO-TK E-Service-Schema-Manager dient der Erfassung, Bereitstellung, Modifikation und Transformation von E-Service-Schema-Spezifikationen [BP04, S.49 ff.]. Diese Funktionen werden durch das Schema-Manager-Anwendungssystem auf verschiedenen Ebenen erbracht. Zum einen dient es als interaktives Werkzeug, das die Ingenieure von VDPN-Broker und -Providern in verschiedenen Phasen bei der Entwicklung von E-Service-Schemata unterstützt. Zum anderen implementiert es die Spezifikation der bereits definierten Plattformkomponente und trägt so zum automatisierten Entwicklungsprozess von E-Service-Instanzen bei.

Die Implementierung des Anwendungssystems gliedert sich in modulare Anteile einer Schema-Manager-Architektur [BP04, S.89 ff.]. Diese baut auf den Grundlagen der allgemeinen Architektur für Plattformkomponenten auf und erweitert diese im Wesentlichen um drei eigenständige Module. Hierzu gehört zunächst ein Web-basiertes Repository für E-Service-Schema-Spezifikationen. Des Weiteren wird ein BPEL-Generator für Interaktionsprozesse innerhalb einer E-Service-Schema-Spezifikation realisiert. Schließlich implementiert eine Engine zur Interpretation und Anwendung von XPDL-Transformationsregeln die systematische Änderung von E-Service-Schema-Spezifikationen.

E-Service-Schema-Management im FRESCO-TK Das FRESCO-TK E-Service-Schema-Manager-Anwendungssystem unterstützt den E-Service-Entwicklungslebenszyklus im Sinne des FRESCO E-Service-Management-Vorgehensmodells schwerpunktmäßig in den zwei Bereichen des vorläufigen Entwurfs von E-Service-Schemata und der agilen Implementierung von E-Service-Instanzen. Die Implementierung von E-Service-Instanzen erfolgt im Kontext des weiter oben beschriebenen automatisierten Prozesses in Kooperation verschiedener Plattformkomponenten. Die Aufgaben der E-Service-Schema-Manager-Komponente, die primär in der Bereitstellung programmatischer Schnittstellen zur Anfrage von E-Service-Schema-Spezifikationen und Generierung von BPEL-Schemata für spezifische Interaktionsprozesse von Capabilities bestehen, wurden dort bereits erläutert.[42]

Der Entwurf von E-Service-Schemata ist hingegen ein überwiegend manueller Prozess, der mit interaktiver Werkzeugunterstützung durchgeführt wird. Er beginnt auf Stufe 2b) des FSMV mit dem partiellen Entwurf von E-Service-PIM und PSM mittels des FRESCO-Entwurfswerkzeugs durch den FRESCO-Broker und FRESCO-Provider (FSMV `PB/PP-2.3`). Bei der anschließenden Validierung (`PB/PP-2.4`) bietet das Entwurfswerkzeug schon die Möglichkeit zur Konformitätsprüfung der generierten XPDL-Dokumente. Die komplette Analyse der logischen Zusammenhänge zwischen den Anteilen einer E-Service-Schema-Spezifikation fällt jedoch in den Aufgabenbereich des Schema-Managers. Dieser bietet daher eine Benutzerschnittstelle, um

einem umfassenden Beispiel sind als Anhang A beigefügt.

[42] Siehe Sektion 5.3.3.2, S. 399.

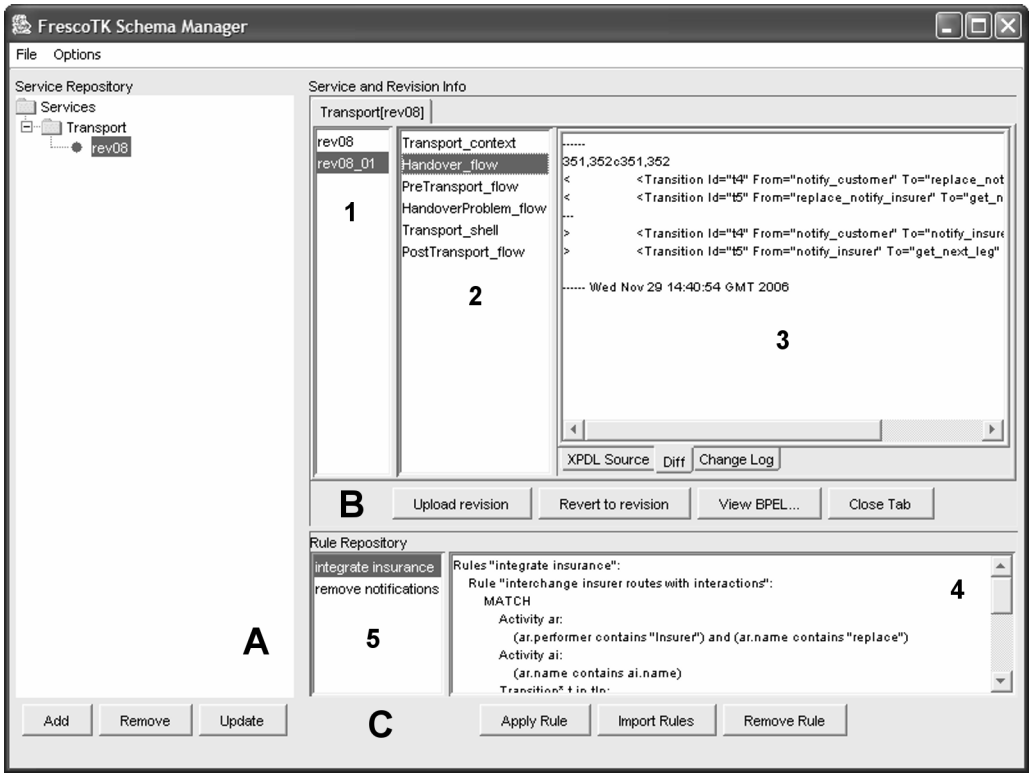

Abbildung 5.26.: Benutzerschnittstelle des E-Service-Schema-Managers

die Schema-Spezifikationen zu laden und einer erweiterten strukturellen Analyse zu unterziehen. Nach erfolgreicher Validierung müssen die ggf. partiellen E-Service-Schema-Spezifikationen dann im strategischen Dienstleistungsnetzwerk publiziert werden, so dass alle Netzwerkunternehmen sie einsehen und wechselseitig abgleichen können (PB/PP-2.5). Die Publikation mittels eines Repository ist ebenfalls Aufgabe des Schema-Managers und wird über dessen Benutzerschnittstelle interaktiv gesteuert. Der verbleibende Schritt des interaktiven Entwurfs ist die E-Service-Schema-Anpassung (PB/PP-5.1). Hierzu implementiert der Schema-Manager die im FRESCO-Entwurf vorgesehene Methode regelbasierter Workflow-Transformation und macht sie interaktiv nutzbar. Dies erfolgt durch eine Umgebung für die interaktive Erstellung und Auswahl von Transformationsregeln, für deren Anwendung auf E-Service-Schema-Spezifikationen, für die Inspektion und Validierung der Ergebnisse und für die Verwaltung resultierender Revisionen inkl. der Möglichkeit zum Rollback.

Konkret werden die interaktiven Funktionen des Schema-Manager-Anwendungssystems mittels der in Abb. 5.26 gezeigten Benutzerschnittstelle bereitgestellt. Diese unterteilt sich in drei primäre Bereiche, die den Funktionsgruppen Repository (A), Analyse (B) und Transformation (C) zugeordnet sind. Der Repository-Bereich erlaubt die Kontrolle eines organisationsübergreifenden Repository für E-Service-Schema-

Spezifikationen. Das Repository unterscheidet neben den Schemata verschiedener E-Services auch verschiedene Revisionen des gleichen Schemas. Der momentane Zustand des Repository mit allen Schemata und Revisionen wird im Repository-Bereich als Baum gezeigt. Diese Anzeige wird beim Start des Schema-Managers aus dem Repository abgefragt und kann dann später mit einem Kontrollelement im Repository-Bereich aktualisiert werden. Des Weiteren können hier neue E-Service-Schemata in Form von Spezifikationsarchiven in das Repository geladen und Schemata im Repository gelöscht werden. Die Auswahl eines Schemas in der Baumanzeige öffnet dessen XPDL-Spezifikation im Analysebereich. Hierbei können auch Schema-Revisionen gewählt werden und jede Auswahl erzeugt einen neuen Reiter. Die Kontrollelemente des Analysebereichs erlauben die Sichtung und Validierung von Schemata. Für jedes Schema werden dessen partiellen XPDL-Spezifikationen gelistet (2) und können in einem Editor-Bereich (3) eingesehen werden, der auch Stichwortsuche unterstützt. Per Knopfdruck kann die Generierung von BPEL-Repräsentationen für Interaktionsprozesse des Schemas im Sinne der im ersten Teil des Kapitels entworfenen Abbildung angewiesen werden.[43] Dabei werden alle logischen Beziehungen zwischen den Definitionen der XPDL-Anteile und SOC-Schnittstellen der Spezifikation validiert. Das Ergebnis besteht in einer BPEL- und einer SOC-Spezifikation, die in einem neuen Fenster angezeigt werden und die Konsistenz der Spezifikation signalisieren. Der Transformationsbereich dient zur Auswahl und Anwendung der im ersten Teil des Kapitels entworfenen Workflow-Transformationsregeln auf die XPDL-Spezifikationen eines Schemas.[44] Einzelne Regelmengen können hier zur späteren Anwendung importiert oder wieder entfernt werden (4). Vor der Anwendung kann eine Sichtung von Regelmengen erfolgen (5). Die Anwendung erfolgt dann für die Auswahl im Analysebereich und führt zu einer neuen vorläufigen Revision eines Schemas (1). Diese Revision kann im Analysebereich manuell und automatisch validiert werden. Zur manuellen Validierung können die durchgeführten Änderungen entweder als Liste oder innerhalb der Spezifikation („Diff“-Ansicht) eingesehen werden. Die BPEL-Generierung führt zu einer automatischen strukturellen Analyse. Bei fehlerhaftem Ergebnis kann die Revision rückgängig gemacht werden. Ansonsten kann sie weiter transformiert oder in das Repository übernommen werden.

E-Service-Schema-Manager-Architektur und -Implementierung Die interaktiven und program-matischen Funktionen, die der Schema-Manager im Kontext des FSMV erfüllt, lassen sich in die drei Bereiche (1) organisationsübergreifender Zugang zu E-Service-Schema-Spezifikationen, (2) systematische Änderung von E-Service-Schema-Spezifikationen durch regelbasierte Transformation und (3) Analyse von E-Service-Schema-Spezifikationen und Detailentwurf von E-Service-Instanzen durch Generierung von BPEL-Schemata gliedern.

[43] Vergleiche Sektion 5.2.1.2, S. 337.
[44] Vergleiche Sektion 5.2.1.2, S. 326.

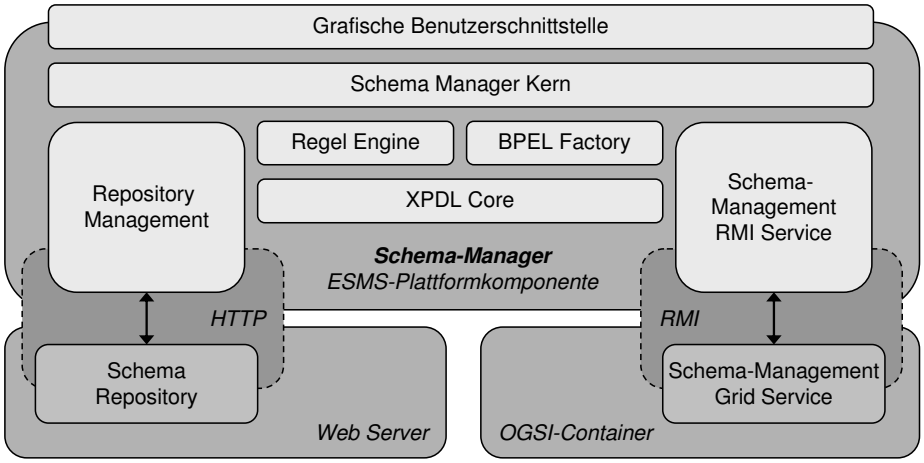

Abbildung 5.27.: E-Service-Schema-Manager-Gesamtarchitektur

Die drei Bereiche werden als getrennte Module implementiert, die in der Schema-Manager-Gesamtarchitektur miteinander verknüpft und mittels verschiedener Schnittstellen bereitgestellt werden. Abbildung 5.27 zeigt eine Übersicht der Architektur. Fundamentaler Bestandteil ist die schon beschriebene XPDL Core-Klassenbibliothek zur Handhabung zusammenhängender XPDL-Spezifikationen. Hierauf baut zum einen die Regel-Engine zur Implementierung der Workflow-Transformation auf (2). Sie realisiert u. a. einen Parser für Transformationsregelmengen und implementiert den Transformationsalgorithmus für XPDL-Spezifikationen. Zum anderen wird der XPDL Core von der BPEL Factory genutzt. Diese analysiert damit die XPDL-Spezifikation eines E-Service-Schemas und gewinnt zusammen mit der Analyse zugehöriger SOC-Spezifikationen die notwendigen Informationen zur Generierung einer BPEL-Repräsentation (3). Schließlich erfolgt die Publikation von E-Service-Schema-Spezifikationen (1) durch die zwei zusammenhängenden Module des Repository-Managements und Schema-Repository. Das Repository-Management-Modul implementiert Funktionen zur Registrierung und Entfernung von E-Service-Schemata verschiedener Revisionen. Dies basiert auf dem Schema-Repository-Modul, das die Bereitstellung von E-Service-Schema-Spezifikationen im Web realisiert. Ein struktureller Schema-Manager-Kern integriert die Module und führt deren Funktionalität verschiedenen Schnittstellen zu. Dies ist zum einen eine Java Swing-basierte grafische Benutzerschnittstelle und zum anderen der Schema-Management Grid Service auf Basis der schon beschriebenen RMI-basierten Grid Connectivity-Architektur. Im Folgenden wird die Implementierung der drei fundamentalen Module im Einzelnen skizziert.[45]

E-Service-Schema Repository Für die Publikation von E-Service-Schema-Spezifikationen im VDPN wird gefordert, dass komplette Spezifikationsarchive mit allen

[45]Eine genauere Beschreibung der Implementierung findet sich in [BP04, S.96 ff.].

XPDL-, WSDL-, GWSDL- und BPEL-Anteilen und in verschiedenen Revisionen derart verfügbar gemacht werden, dass sie in programmatischer und interaktiver Weise von allen Netzwerkunternehmen zugegriffen werden können. Der entsprechende Ansatz im FRESCO-TK besteht in einem sehr einfachen Web-basierten Repository, das durch den Schema-Manager kontrolliert wird. Das Repository wird als Web-AS auf Basis des Apache Tomcat-Applikationsservers realisiert. Speicherung, Entfernung, Zugriff und Abfrage von Dateien erfolgen über HTTP. Der Zugriff auf Dokumente der Spezifikation erfolgt per `HTTP GET` unter Angabe von dessen URL und muss im Tomcat Server nicht implementiert werden. Die Speicher-, Lösch- und Abfragefunktionen werden hingegen als Servlets implementiert. Die Kontrolle dieser Servlets erfolgt aus dem Schema-Manager heraus mittels eines HTTP-Client-Moduls, das die HTTP-Kommunikation kapselt und dem Schema-Manager-Kern eine abstrakte Schnittstelle zu den Repository-Funktionen bietet. Die Implementierung von Servlets und Client erfolgt auf Basis der beiden Apache Commons Software-Rahmenwerke FileUpload und HttpClient. Diese erlauben die einfache Realisierung von `HTTP GET`- und `POST`-Interaktionen. Es werden insbesondere auch mehrteilige `POST`-Nachrichten unterstützt, die zum Versand kompletter Spezifikationsarchive verwendet werden.

Regel-Engine Die Regel-Engine ist als Software-Rahmenwerk implementiert, das ein Objektmodell für Transformationsregeln, Parser zur Übersetzung textueller Regelspezifikationen und eine Umsetzung des beschriebenen Transformationsalgorithmus auf Basis des XPDL Core beinhaltet. Die programmatische Anwendung des Rahmenwerks zur Transformation von XPDL-Spezifikationen mittels einer Regelmenge erfolgt mittels der Klassen `RuleSet`, `RuleSetApplication` und `MatchSelector`. `RuleSet` beinhaltet statische Methoden zur Übersetzung einer Regelmenge in das Objektmodell. Das resultierende `RuleSet`-Objekt dient zusammen mit einem `MatchSelector` als Parameter zur Erzeugung einer `RuleSetApplication`. Die Klasse beinhaltet verschiedene Methoden zur Anwendung der Regelmenge auf eines oder mehrere `XpdlProcess`-Objekte bzw. einen `PackageContext` des XPDL Core. Abbildung 5.28 deutet die Zusammenhänge in einem Ausschnitt der Klassenhierarchie an.

Die `apply()` Methoden wenden die verschiedenen Regeln einer Regelmenge nacheinander mittels Erweiterungen der abstrakten `RuleApplication`-Klasse auf die ggf. verschiedenen Prozesse an. Dies beinhaltet zunächst die Auswertung des Prozessselektors (`MatchProcessSelector` einer `Rule`) und ggf. der Suchmuster (`ElementMatches` und `ListMatches` einer `Rule`), bevor schließlich die Ersetzungsvorschrift (`Assigns` einer `Rule`) mit der Erzeugung, Modifikation und Entfernung von Prozesselementen angewendet wird. Die Auswertung unterscheidet sich nach Semantik der Suche. Im Falle von `MATCH_ONCE`-Semantik wird bei multiplen Treffern die `selectMatch()`-Methode einer individuellen Implementierung der `MatchSelector`-Schnittstelle aufgerufen. Diese kann im einfachsten Fall den ersten Treffer wählen oder auch eine Interaktion mit dem Benutzer zur manuellen Auswahl anstoßen.

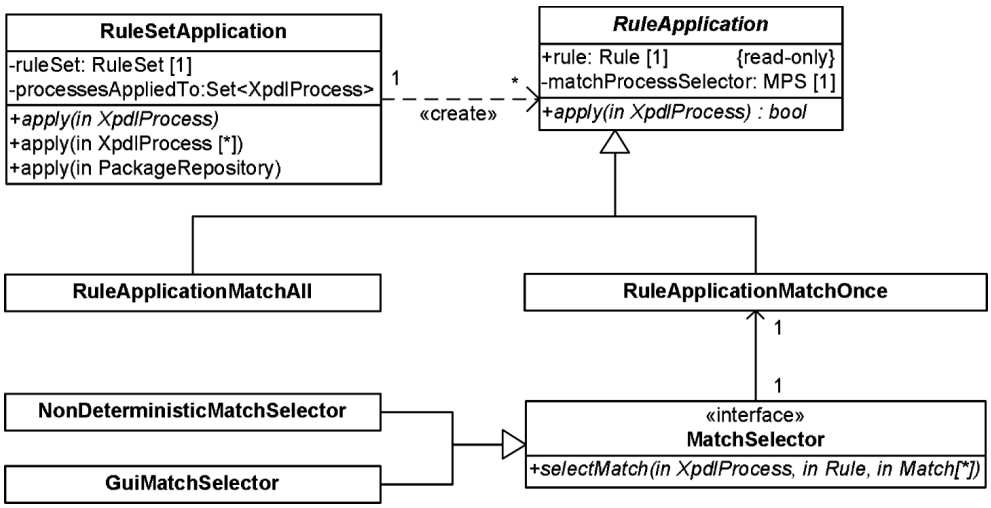

Abbildung 5.28.: Transformationsregelanwendung der Rule-Engine [BP04, S. 98]

Bei der Auswertung von Suchmustern werden gemäß Schritt zwei des Algorithmus zunächst die Basismengen ermittelt, was direkt mittels Zugriffsmethoden der XPDL Core-Elemente geschieht. Die Bildung von Trefferkandidaten mittels Permutation der Basismengen (Schritt drei) erfolgt mit der Klassen `CrossProductIterator` zur Bildung aller Kombinationen von Elementen der Basismengen und `MatchIterator` zu deren Reduktion auf solche mit unterschiedlichen Elementen. Die Evaluation von Konditionen der Suchspezifikation (Schritte vier bis sieben) basiert auf einem Namensraum, der als `Map` realisiert ist. Zur Auswertung des Prozessselektors enthält diese nur das Prozess-Element und wird dann durch eine Abbildung der Namen im Suchmuster auf die Elemente eines Trefferkandidaten erweitert. Die Resultate von Auswertungen einzelner Konditionen werden konjunktiv vereinigt. Zur Anwendung der Ersetzungsvorschrift (Schritt acht) werden dem Namensraum eines Trefferkandidaten die neuen Elemente hinzugefügt und deren Eindeutigkeit in Bezug auf die Basismenge, ggf. mittels Zähler-Präfix für das identifizierende Attribut, sichergestellt. Schließlich werden die Ersetzungsspezifikationen angewendet.

BPEL Factory Das BPEL Factory Modul des Schema-Managers realisiert einen Generator für BPEL-Schemata auf Basis von `eService Capability Interaction Flow`-Prozessen einer E-Service-Schema-Spezifikation. Zur Anwendung des Generators wird lediglich dessen `BpelFactory` Klasse benötigt, die programmatisch oder interaktiv per Kommandozeile benutzt werden kann. Deren `transformToBpel()`-Methode wird mit einem `XpdlProcess`-Objekt des XPDL Core sowie verschiedenen URIs zum Zugriff auf zugehörige SOC-Spezifikationen aufgerufen und liefert zwei XML-Dokumente mit dem BPEL-Schema und der SOC-Spezifikation zurück. Der hierzu notwendige Vorgang der Generierung beinhaltet zunächst die Analyse von XPDL- und SOC-Dokumenten der

E-Service-Schema-Spezifikation. Auf Basis der gewonnenen Informationen kann die Konstruktion eines BPEL-Schemas sowie einer zugehörigen SOC-Spezifikation nach Vorgabe des weiter vorne entworfenen Abbildungskonzepts erfolgen.

Zur Implementierung des Generierungsvorgangs sind Ansätze auf Basis von Schablonen (z. B. XSLT[46]) weniger gut geeignet.[47] Da die komplexen Zusammenhänge innerhalb von E-Service-Schema-Spezifikationsarchiven eine aufwendige Analyse multipler Dokumente erfordern, wurde stattdessen Java verwendet. Hierzu werden grundsätzlich Repräsentationen von XPDL-, BPEL- und SOC-Spezifikationen als Objektmodelle benötigt. Zur Zeit der Implementierung war diesbezüglich lediglich die Java API for WSDL (WSDL4J) verfügbar. Für BPEL war hingegen keine vergleichbare Lösung erhältlich. Daneben ergaben sich in Bezug auf den Einsatz von XML Binding APIs ähnliche Probleme wie bei XPDL.[48] Aus diesem Grund wurde das benötigte BPEL-Objektmodell auf Basis des generischen XML-Objektmodells XOM eigenständig entwickelt. Die Handhabung von XPDL-Spezifikationen basiert auf dem XPDL Core.

Die Implementierung des BPEL-Generators setzt sich im Wesentlichen aus Entitäts-, Abbildungs-, Kontext-, Serialisierungs- und Ausnahmeklassen zusammen. Entitätsklassen repräsentieren BPEL-spezifische Elemente, die während des Generierungsvorgangs erzeugt und am Ende als XML-Dokumente serialisiert werden. Dies beinhaltet Klassen für die Elemente von BPEL-Schemata sowie BPEL-spezifische Elemente für SOC-Spezifikationen als Erweiterung von WSDL4J. Abbildungsklassen regeln die Generierung von Zielelementen (BPEL und WSDL) auf Basis einer Analyse von Ausgangselementen (XPDL und WSDL). Kontextklassen dienen als Informationsquelle für die Abbildungsklassen. Sie bieten Zugriff auf die Ausgangselemente sowie auf die sukzessive hinzukommenden Zielelemente. Serialisierungsklassen erlauben die Serialisierung von Entitätsklassen als XML-Dokumente. BPEL-Entitätsklassen implementieren hierzu eine einfache Schnittstelle (`IElement`) für XML-spezifische Ausgaben, die mittels des XOM-Rahmenwerks angesprochen wird. Im Falle der BPEL-spezifischen SOC-Erweiterungen werden entsprechende Serialisierungsklassen in WSDL4J eingebunden. Die verbleibenden Ausnahmeklassen dienen zur Signalisierung einer Vielzahl möglicher struktureller Fehler in E-Service-Schema-Spezifikationen. Derartige Fehler werden vom Schema-Manager-Kern an die GUI geleitet und geben wichtige Hinweise für den E-Service-Schema-Entwurf.

Der Generierungsvorgang basiert auf dem in Sektion 5.2.1.2 entworfenen Algorithmus zur Kontrollflussabbildung. Hierbei wird zunächst eine einzelne `BpelFlow`-Entitätsklasse erzeugt. In diesem Rahmen erfolgt die Übersetzung von Aktivitäten, Transitionen und Transitionsbedingungen des `XpdlProcess` nach Vorgabe des Algorithmus. Hierbei spielt die Übersetzung von `XpdlActivity`-Elementen eine Schlüsselrolle, da sie alle internen und externen Datenflussaspekte einschließt. Im Kontext eines

[46] XSL Transformations (`http://www.w3.org/TR/xslt`).
[47] Siehe hierzu [BP04, S.100].
[48] Vgl. Sektion 5.3.3.2, S. 403.

`eService` `Capability` `Interaction` `Flow` des E-Service-Schemas repräsentieren Aktivitäten `Interactions` und sind stets mit einer `Role` (XPDL `Participant`) und einem `Contact` (XPDL `Application`) im XPDL-Prozess sowie mit `portType` und `operation` einer SOC-Schnittstelle assoziiert. Aus diesen Informationen werden neben dem entsprechenden `invoke-` oder `receive-` u. a. auch `variable-`, `message-`, `assign-`, `partner-`, `correlationSet-`, `serviceLinkType-`, `property-` und `propertyAlias`-Elemente sowie Namensraumdeklarationen im BPEL-Schema und der dazugehörigen SOC-Spezifikation abgeleitet. All diese Elemente werden ausschließlich im Zusammenhang mit der Übersetzung der XPDL-Aktivität erzeugt, d. h. dass XPDL-Definitionen ohne Beziehung zu einer Aktivität nicht ins BPEL-Schema übernommen werden. Wenn auf diese Weise der komplette Kontrollfluss Übersetzt wurde, sind die Speicherrepräsentationen des BPEL-Schemas und der SOC-Spezifikation komplett und werden mittels der Serialisierungsklassen als XML-Dokumente geschrieben.

5.3.4.3. FRESCO-TK E-Service Engine-Manager

E-Service Engine-Management ist ein integraler Bestandteil der automatisierten E-Service-Instanz-Implementierung während der E-Service-Konstruktion auf Stufe 4 des FSMV. Der Vorgang ist im Wesentlichen dadurch motiviert, den E-Service-Entwicklungsprozess in seiner kritischen Phase zu beschleunigen. Dies basiert darauf, Capabilities, die den primären Ansatzpunkt zur Änderung von VDLP darstellen, in E-Service-Anwendungssystemen als E-Service Engine-Komponenten automatisch zu konstruieren und zu installieren. Grundlage hierfür ist die im Rahmen der FRESCO-Plattform entworfene[49] und in der FRESCO-TK Plattformarchitektur technisch konkretisierte[50] generische E-Service Engine-Architektur. Auf Basis dieses Rahmenwerks umfasst E-Service Engine-Management die Analyse von E-Service Capabilities zum Entwurf interner E-Service Engine Komponenten in Form von Engine Adaptern, BPEL-Prozessen und Ressourcen-Proxies, die Generierung von Code zu deren Implementierung sowie die Konfiguration der beteiligten Ausführungsumgebungen und die Installation der Komponenten darin. Die Aktivitäten des E-Service Engine-Managements werden von der E-Service Engine-Manager-Plattformkomponente gesteuert und zum Teil als automatische Mechanismen implementiert. Im Folgenden wird zunächst dessen Gesamtarchitektur im Kontext der vielfältigen beteiligten Technologien dargestellt. Dann wird die zentrale Funktion der E-Service Engine-Konstruktion erläutert. Schließlich werden die interaktiven Aspekte des Engine-Managements anhand der zugehörigen Benutzerschnittstelle beschrieben.

E-Service Engine-Management Architektur Die E-Service Engine-Manager-Architektur ist insbesondere durch ihre Vermittlerrolle zwischen den verschiedenen

[49] Siehe Sektion 5.2.2.2, S. 371.
[50] Siehe Sektion 5.3.3.1, S. 395.

technischen Komponenten der E-Service Engine-Architektur geprägt. Dies betrifft zum einen die externe E-Service-Plattform-Architektur, in die sich das E-Service Engine-Management als Aspekt bei der kooperativen Entwicklung von E-Service-Instanzen eingliedert. In diesem Kontext kooperiert und kommuniziert die Engine-Management-Komponente mit den anderen Plattformkomponenten. Dieser Prozess, der insbesondere die eingehende Anfrage zur Konstruktion und Installation einer E-Service Engine-Komponente durch den Aggregation-Manager und die ausgehende Anfrage zur Generierung von ausführbaren BPEL-Schemata an den Schema-Manager beinhaltet, wurde in Sektion 5.3.3.2 bereits ausführlich dargestellt. Zum anderen bezieht sich die Vermittlerrolle auf die verschiedenen Plattformen, die der internen E-Service Engine-Architektur zugrunde liegen und die zur Konstruktion entsprechender Komponenten einbezogen werden müssen. Konkret sind das die in Abbildung 5.16 schon gezeigten Ausführungsumgebungen für E-Service Engine Adapter, BPEL-basierte Web Service-Komponenten und E-Service-Ressourcen-Proxies; die internen Komponenten einer E-Service Engine.

Engine Adapter sind Grid Service-Komponenten, die die E-Service Engine mittels erweitertem `FrescoResource portType` für andere Anwendungskomponenten zugreifbar machen. Sie benötigen dazu einem OGSI-konformen Container, der im FRESCO-TK durch den GT3 Core gegeben ist. Zur Entwicklung von Adaptern bietet der GT3 Core verschiedene Werkzeuge auf Basis von Apache Ant. Hierzu zählen u. a. ein Schnittstellenübersetzer zur Erzeugung von Client Stubs und ein Deployment-Mechanismus. Des Weiteren steht ein Software-Rahmenwerk zur Verfügung, das weiter oben schon im Zusammenhang mit der E-Service-Ressourcen-Architektur eingeführt wurde. Der Engine-Manager generiert zur Adapter-Konstruktion Code auf Basis des erweiterten FRESCO-TK Rahmenwerks für E-Service-Ressourcen und verwendet dabei verschiedene der GT3 Core-Mechanismen.

BPEL Web Service-Komponenten implementieren die Interaktionsmuster von E-Service Capabilities. Die dazu notwendigen BPEL-Schemata und erweiterten SOC-Schnittstellen werden vom Schema-Manager generiert. Deren Ausführung basiert auf einer BPEL-Ausführungsumgebung, die im FRESCO-TK in Form von IBMs IBM BPEL Java Run Time (BPWS4J) vorliegt. BPWS4J ist eine Web Service Composition Middleware u. a. auf Basis von Apaches Axis Simple Object Access Protocol Server. Sie liegt als Web-Anwendungssystem vor und wird im FRESCO-TK innerhalb des Tomcat Web-Applikationsservers ausgeführt. Als Prototyp realisiert BPWS4J zwar eine funktionstüchtige Ausführungsumgebung für BPEL-Prozesse, verfügt aber nur über rudimentäre Administrationsmechanismen. So ist hier insbesondere das Deployment nur in manueller Form möglich. Der Engine-Manager sieht hierzu einen interaktiven Prozess mit manuellem Eingriff vor.

Ressourcen-Proxies sind Web Service-Komponenten, die es BPEL-Komponenten erlauben mit E-Service-Ressourcen in Verbindung zu treten, die als erweiterte Grid Service-Komponenten vorliegen. Sie nehmen dazu Nachrichten von einer BPEL-Komponente entgegen, lösen die darin enthaltenen Bindungsinformationen mithilfe des

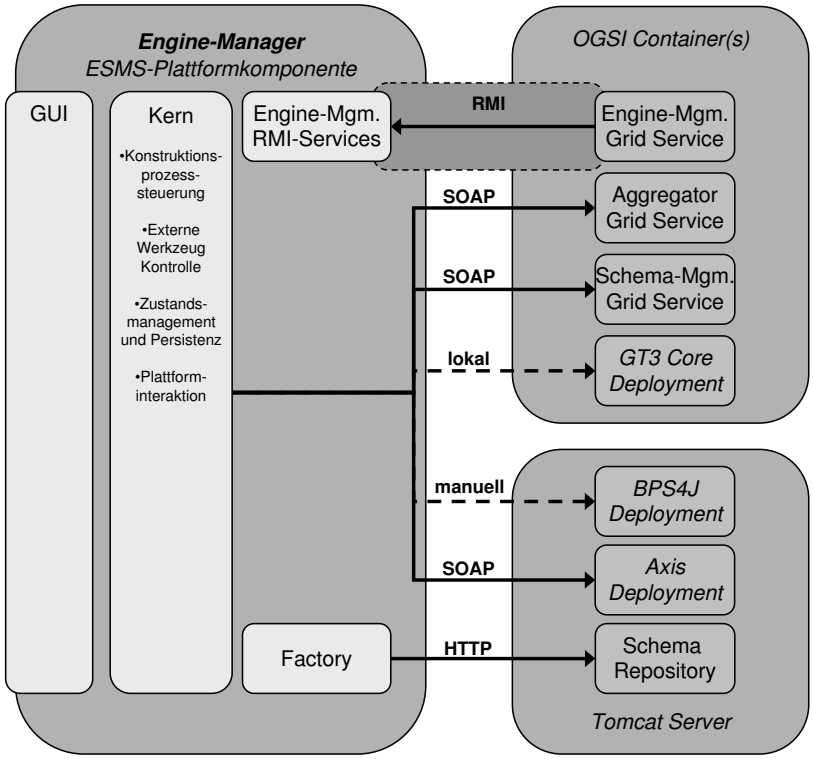

Abbildung 5.29.: Architektur des FRESCO-TK Engine-Managers

Aggregation-Managers auf und senden sie an die ermittelte Grid Service-Komponente weiter. Die Realisierung dieser Funktionalität erfordert lediglich fundamentale Web Service Middleware-Mechanismen in Form eines SOAP Servers, wofür im FRESCO-TK Apache Axis auf Basis von Tomcat verwendet wird. Konform zum GT3 (das selbst auf Axis basiert) werden hier ein Schnittstellenübersetzer und ein JAX-RPC-konformes Software-Rahmenwerk bereitgestellt, was der Engine-Manager zur Konstruktion von Proxies verwendet. Darüber hinaus bietet Axis Laufzeitmechanismen zum Deployment von Proxy Web Services.

Abbildung 5.29 zeigt die Gesamtarchitektur des Engine-Managers im Überblick. Auf der rechten Seite sind die oben beschriebenen Interaktionen mit Technologiekomponenten der Plattform- und Engine-Architektur angedeutet. Die Engine-Manager-Komponente basiert auf der generischen Grid Connectivity-Architektur und läuft somit in einer autonomen Java VM mit Anbindung an den GT3 OGSI Container via RMI. Die wichtigsten Funktionen zur Steuerung des ganzheitlichen Entwicklungsprozesses, zur Kontrolle von Entwicklungswerkzeugen der verschiedenen Plattformen, zur persistenten Sicherung des Entwicklungszustandes und zur Interaktion mit den kooperierenden Plattformkomponenten sind im Kernmodul konzentriert und werden mittels einer grafischen Benutzeroberfläche überwacht und gesteuert. Die in sich geschlossene Funktion der Codegenerierung für Adapter und Proxy-Komponenten

mitsamt Zugriff und Analyse von Komponentenschnittstellen ist hingegen in dem externen Modul der Engine-Factory ausgelagert. Gleich werden zunächst dieses Modul und der Gesamtprozess aus interaktiver Benutzersicht beschrieben.

Konstruktion von E-Service Engines Der zentrale Bestandteil der E-Service Engine-Manager-Architektur ist die E-Service Engine Factory. Sie dient zur Codegenerierung für E-Service Engine Adapter-Komponenten und E-Service-Ressource Proxy-Komponenten.

Das Prinzip dieser beiden Vorgänge wird in Abbildung 5.30 skizziert. Die Konstruktion von Proxies basiert auf der in der E-Service-Schema-Spezifikation enthaltenen Schnittstellenbeschreibung der E-Service-Ressourcen mittels GWSDL. Der Engine-Manager-Kern lädt die Schnittstellenbeschreibung aus dem Schema-Manager-Repository. Er nutzt dann die Apache Ant-API, um das im GT3 Core enthaltene GWSDL2Java-Werkzeug zu starten, das für die `portTypes` der Ressourcen Java Stub-Klassen (`ResourcePortType.java`, ...) generiert. Schließlich ruft er die `Generator`-Klasse des Factory-Moduls auf um den Proxy Code (`ResourceProxy.java`) zu generieren. Dieser beinhaltet zunächst den Java Code der Komponente. Der Code enthält statische Anteile, um zusätzliche Standard-Parameter der Aufrufe auszulesen, die Korrelationsinformationen enthalten. Diese Meta-Informationen werden mittels der FRESCO-TK `AggregatorHelper`-Klasse an den Aggregation-Manager geschickt, der u. a. den GSH der letztendlichen Ressource zurückliefert. Code zur Bindung an die Grid Service-Komponente der Ressource und die Weiterleitung von Operationsaufrufen wird dynamisch auf Basis von Introspektion der Stub-Klassen und Analyse der Schnittstellenbeschreibungen gebildet. Zu dem Code wird dann noch ein passender Deployment Descriptor (`Proxy.wsdd`) generiert und alles zusammen an den `AdminService` des Axis-Servers gesendet. Dieser liefert daraufhin die SOC-Schnittstellenbeschreibung des Proxy inklusive Bindungsinformationen und Endpunkt.

Die Konstruktion des Adapter verläuft parallel zu der des Proxy. Hier werden zunächst alle SOC-Schnittstellenspezifikationen von BPEL-WS (nach dessen Deployment inkl. Bindungsinformationen und Endpunktreferenz) der E-Service Engine (korrespondierend zu den multiplen Interaktionsmustern einer Capability) verwendet, um mittels Axis WSDL2Java Web Service Stubs (`FlowPortType.java, ...`) zu generieren. Im Anschluss wird Java Code (`Adapter.java`) für den Adapter generiert. Dies beinhaltet wiederum statischen Code zur Implementierung von Standard-Parametern. Es handelt sich dabei um die Kennung der E-Service-Instanz, die immer zur Korrelation von BPEL-Prozessinstanzen verwendet wird. Der Code liest diese Information aus dem Adapter aus, der sie als spezifisches `frescoRessourceState` SDE beinhaltet, das bei der Allokation vom Aggregation-Manager zugewiesen wird. Die zu implementierenden Methoden werden dann wiederum dynamisch ermittelt. Insbesondere aggregiert der Adapter alle Operationen der ggf. multiplen BPEL-WS einer Engine und dient somit als zentraler Zugangspunkt der Capability. Als weiteres Artefakt wird wiederum

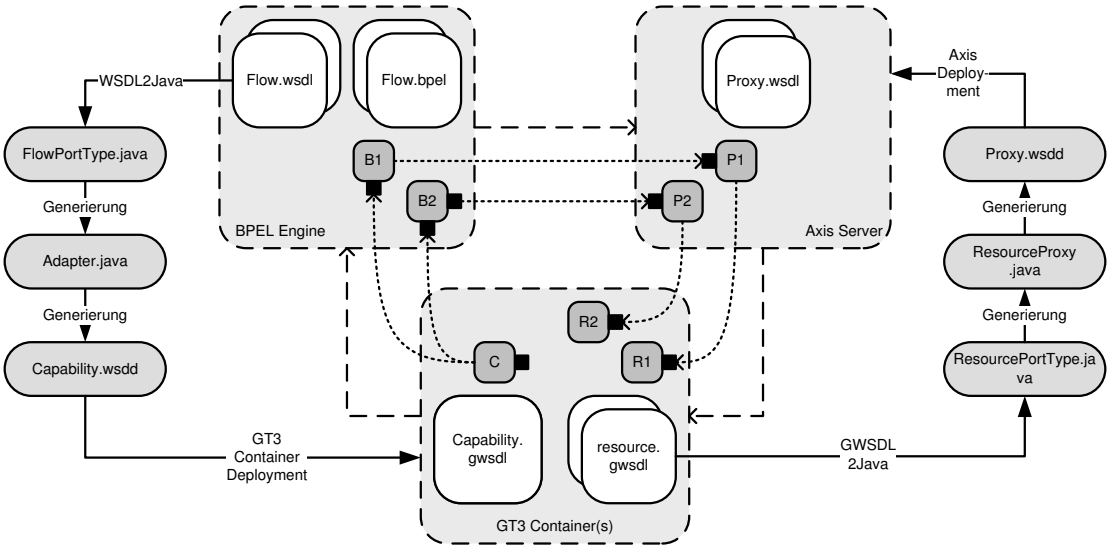

Abbildung 5.30.: Konstruktions- und Deployment-Prozess für E-Service Engines

ein Deployment Descriptor generiert. Alles zusammen wird mit den Ant-basierten Werkzeugen des GT3 Core im OGSI Container installiert.

E-Service Engine-Management Auf Basis der E-Service Engine-Manager-Plattformkomponente ist E-Service Engine-Management ein hochgradig automatisierter Vorgang. Der Vorgang ist jedoch nicht vollkommen automatisiert. Für verschiedene Aspekte ist ein gewisser Grad interaktiver Kontrolle wünschenswert. Andere Aspekte benötigen manuelle Eingriffe. Aus diesem Grund beinhaltet der E-Service Engine-Manager eine grafische Benutzerschnittstelle, die in Abb. 5.31 gezeigt wird.

Die Engine-Manager-GUI gliedert sich in die drei Bereiche der Capability-Tabelle (A), der BPEL-Schema-Tabelle (B) und der Entwicklungsprozess-Überwachung (C). Im Zuge der E-Service-Instanz-Implementierung läuft der Engine-Manager bei einem VDLP-Provider ab. Der Aggregation-Manager sendet diesem bei Bedarf Anfragen zum Deployment einer Capability der E-Service-Schema-Spezifikation. Diese Anfragen werden in der Capability-Tabelle gesammelt. Um den Entwicklungsprozess zu starten, wählt der Provider eine Capability aus der Tabelle aus und weist per `Deploy Resource` (1) die Konstruktion und Installation von Ressourcen-Proxies an. Der komplexe Vorgang kann in allen Schritten überwacht und ggf. abgebrochen werden (2). Am Ende macht der Engine-Manager Kopien der SOC-Spezifikationen von Ressourcen-Proxies im lokalen Dateisystem. Der nächste Schritt besteht in der Beschaffung und dem lokalen Kopieren der zugehörigen BPEL-Schemata und SOC-Spezifikationen der Capability-Interaktionsmuster vom Schema-Manager per `Load BPEL` (3). An diesem Punkt erfolgt das manuelle Deployment der BPEL Web Services im BPWS4J anhand der lokalen Kopien von BPEL- und SOC-Spezifikationen. Als Ergebnis wird für jeden

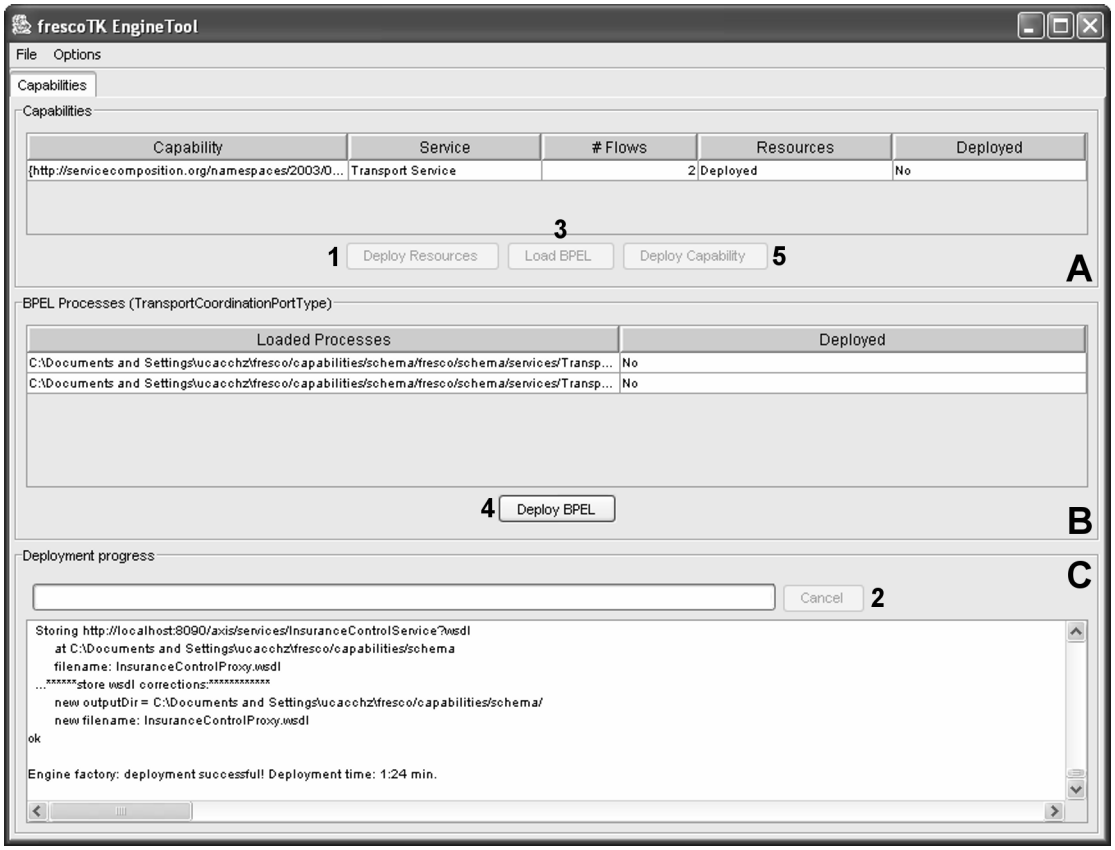

Abbildung 5.31.: Benutzerschnittstelle des E-Service Engine-Managers

BPEL Web Service eine Schnittstellenbeschreibung mit Bindungsinformationen zur Verfügung gestellt, deren URL per GUI an den Engine-Manager zu übergeben ist (4). Im Anschluß daran kann die Entwicklung des Adapter per `Deploy Capability` (5) angewiesen werden, was den Entwicklungsprozess abschließt. Abschließend ist die Capability dann bereit für die Allokation durch den Aggregation-Manager und für die Steuerung des E-Service.

5.3.4.4. FRESCO-TK E-Service-Aggregation-Manager

E-Service-Aggregation-Management dient zur Koordination von Komponenten der ESMS-Plattform bei Implementierung und Ausführung von E-Service-Instanzen. Es bildet dadurch das zentrale Instrument für FRESCO-Broker zur Steuerung und Kontrolle der Kontaktphase virt. Dienstleistungen. Die wichtigsten Aspekte des Vorgangs betreffen die Verwaltung von E-Service-Instanzen und teilnehmenden Netzwerkunternehmen bzw. Kunden sowie deren Beziehungen in Bezug auf die Übernahme von Rollen des E-Service-Schemas und Erfüllung implizirter Verpflichtungen zur Bereitstellung von Ressourcen.

Im FRESCO-TK wird der Vorgang des Aggregation-Managements durch eine Implementierung des weiter oben entworfenen Konzepts zur dynamischen Aggregation von E-Service-Ressourcen[51] als Plattformkomponente realisiert. Zur Erläuterung dieser Aggregation-Manager-Komponente wird zunächst der von ihr unterstützte interaktive Vorgang erläutert. Im Anschluss werden die Aggregation-Manager-Architektur beschrieben und verschiedene Details der Implementierung hervorgehoben.

FRESCO-TK E-Service-Aggregation-Management Die Funktionen des Aggregation-Managers im kooperativen Vorgang der Implementierung und Ausführung von E-Service-Instanzen auf Plattformebene des ESMS wurden weiter oben bereits genau spezifiziert.[52] Sie beinhalten im Wesentlichen Assoziation, Deployment und Allokation der Ressourcen von Teilnehmern einer E-Service-Instanz und erfolgen als Gesamtvorgang im Sinne einer Aggregationsstrategie. Je nach Strategie sind die Teilvorgänge der Aggregation mehr oder weniger weit automatisiert. Gleiches gilt für die Steuerung und Kontrolle des Gesamtvorgangs. Im Extremfall verläuft das Aggregation-Management als vollautomatisch gesteuerte Interaktionen zwischen Plattform- und Anwendungskomponenten. In anderen Fällen ist jedoch ein gewisser Grad an interaktiver Kontrolle notwendig oder es sind manuelle Eingriffe notwendig. Das andere Extrem ist dann ein manueller Prozess, dessen Ergebnisse in der Plattform interaktiv registriert werden.

Der FRESCO-TK E-Service-Aggregation-Manager implementiert das Aggregation-Management-Konzept derart, dass beliebige Aggregationsstrategien zwischen den beiden Extremformen realisiert werden können. Er bietet dazu eine Benutzerschnittstelle, mittels der eine manuelle Form des Aggregation-Managements interaktiv mit der Plattform synchronisiert werden kann. Darüber hinaus existiert eine API-Schnittstelle zur programmatischen Realisierung von Aggregationsstrategien sowie Assoziations-, Deployment- und Allokationsmodellen. Deren ggf. manuelle Anteile können wiederum über die GUI mit der Plattformebene synchronisiert werden. Im Folgenden wird zunächst die Interaktion mit der Aggregation-Manager-GUI bei einer manuellen Form der Aggregation beschrieben. Die API wird danach im Zusammenhang mit der Implementierung der Plattformkomponente vorgestellt.

Abbildung 5.32 zeigt den Aufbau der Aggregation-Manager-GUI, die drei primäre Bereiche umfasst. Der *Aggregationsbereich (A)* beinhaltet verschiedene Unterbereiche (1) mit Tabellen (2) den aggregationsrelevanten Elementen von E-Service-Instanzen und bietet Steuerelemente (3) für die damit zusammenhängenden Aggregationsfunktionen. Darunter zeigt der *E-Service-Baum (B)* die strukturellen Zusammenhänge aller erfassten Elemente aus verschiedenen Perspektiven. Im unteren Bereich schließt der *Detailbereich (C)* die GUI ab. Hierin wird ein umfassender Report aller registrierten Informationen für das im E-Service-Baum ausgewählte Element dargestellt.

[51] Siehe Sektion 5.2.2.2, S. 366.
[52] Siehe Sektion 5.3.3.2, S. 399.

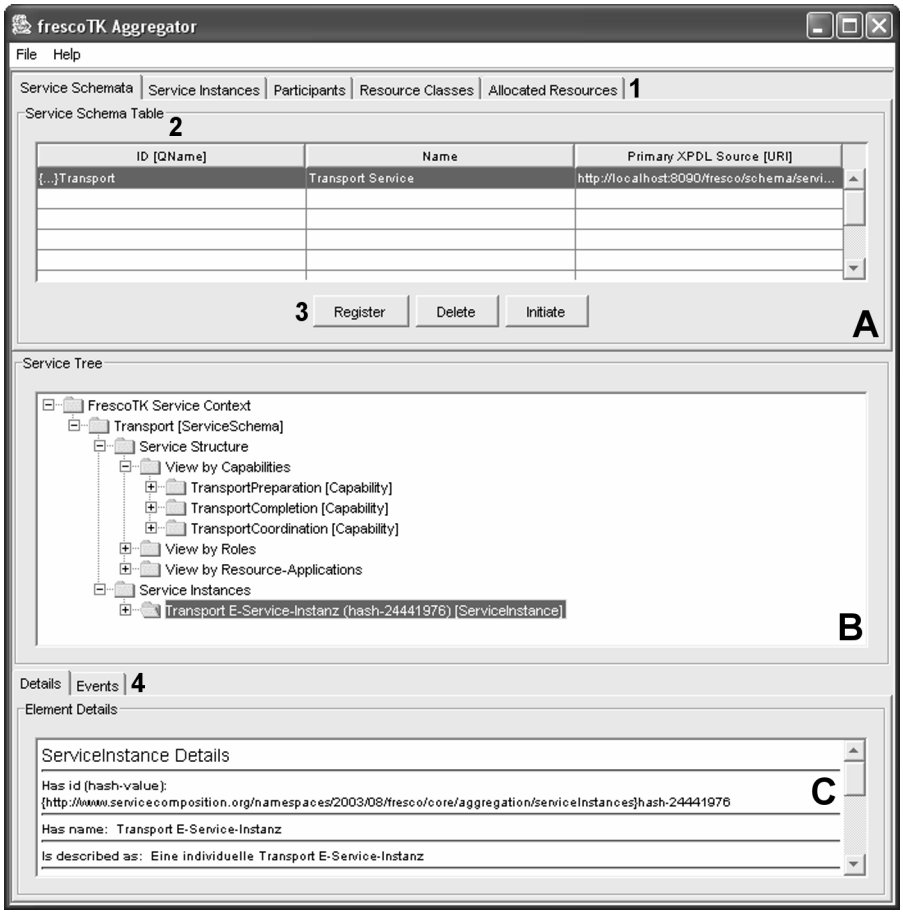

Abbildung 5.32.: Benutzerschnittstelle des E-Service-Aggregation-Managers

Des Weiteren existiert ein Unterbereich (4) zur Protokollierung von Interaktionen zwischen Plattformkomponenten.

Das manuelle Aggregation-Management basiert im Wesentlichen auf den Funktionen des Aggregationsbereichs. Tabelle 5.9 zeigt eine Übersicht der hierin enthaltenen Tabellen und Funktionen. Der manuelle Vorgang beginnt mit der Registrierung eines E-Service-Schemas (`Register`). Hierbei ist der qualifizierte Bezeichner des E-Service und ggf. die URL des Spezifikationsarchivs im E-Service Repository anzugeben. Falls Letztere fehlt, wird diese bei der Schema-Manager-Plattformkomponente erfragt. Zu diesem Zeitpunkt müssen alle ggf. notwendigen Änderungen der Spezifikation abgeschlossen sein. Grundsätzlich können hier alle einsatzbereiten Schemata im strategischen Dienstleistungsnetzwerk erfasst werden. Zur Durchführung von Änderungen sind diese zunächst zu entfernen (`Delete`) und danach erneut zu registrieren. Der Übergang in die VDL-Kontaktphase wird dann durch die Initiierung eines Schemas markiert (`Initiate`). Hierbei werden insbesondere die Aggregationsstrategie sowie die Assoziations-, Deployment- und Allokationsmodelle festgelegt,

die im weiteren Verlauf zum Einsatz kommen sollen. Der Aggregation-Manager unterstützt hier eine Standardstrategie sowie die manuelle Aggregation. Die Standardstrategie „Lazy-Aggregation“ führt alle notwendigen Aggregationsschritte zum spätest möglichen Zeitpunkt aus. Das Assoziationsmodell ist entweder automatisch per `Participant≐Registry` Grid Service-Schnittstelle portType oder manuell. Es ist jedoch stets passiv, denn der Aggregation-Manager meldet die Notwendigkeit zur Assoziation einer bestimmten Rolle lediglich und wartet dann auf die Durchführung dieser Aktivität. Das Deploymentmodell ist ebenfalls stets passiv und kann entweder manuell oder mithilfe des Engine-Managers für Capabilities automatisch erfolgen. Beim Allokationsmodell kann schließlich zwischen einer passiven manuellen oder einer aktiven automatischen Variante gewählt werden. Letztere basiert auf OGSI Grid-Mechanismen und nutzt Repositories und Factories, die die Teilnehmer für ihre Ressourcen zur Verfügung stellen müssen. Nach Initiierung eines Schemas wird die erzeugte Instanz mit eindeutigem Bezeichner in der entsprechenden Tabelle eingetragen und wartet auf Aktivierung der Aggregationsstrategie (`Start Aggregation Strategy`). Mit dem Start der Strategie beginnt die Erbringung der Dienstleistung. Falls nötig kann diese Phase durch Stop der Aggregation unterbrochen werden (`Stop Aggregation Strategy`). Vor Beginn oder nach Abbruch der Aggregation kann auch die Instanz entfernt werden (`Delete`). Für laufende E-Services ist dies nicht sinnvoll und wird unterbunden.

Nach dem Start der Aggregationsstrategie wird früher oder später die Zuweisung von Rollen nötig. Dies ist spätestens dann der Fall, wenn die Anfrage nach einer Ressource eintrifft, kann jedoch je nach Strategie auch früher geschehen. In diesem Fall wird eine Aufforderung an den Benutzer (manuelles Modell) oder ein System (automatisches Modell) erzeugt. Unabhängig davon kann die Zuweisung jedoch auch früher erfolgen. Dazu ist als Erstes ein Teilnehmer mitsamt seiner lokalen OGSI-Plattform zu registrieren (`Register`). Letzteres beinhaltet den GSH einer Registry, in der Factories für E-Service-Ressourcen zu finden sind. Zur Information kann der Inhalt dieser Registry abgefragt und ausgegeben werden (`Query Registry`). Grundsätzlich kann der Aggregation-Manager als Repository für Kunden und Produktionseinheiten des strategischen Dienstleistungsnetzwerks genutzt werden, so dass eine Registrierung von Teilnehmern nicht immer wieder neu erfolgen muss. Für registrierte Teilnehmer kann die Zuweisung einer Rolle erfolgen (`Assign Role`). Dabei müssen insbesondere die Korrelationsinformationen zur Rolle festgelegt werden. Standardmäßig ist das die Kennung der E-Service-Instanz. Weitere Werte können je nach Spezifikation des E-Service-Schema hinzukommen und müssen per Dialog spezifiziert werden. Nach der Zuweisung werden die Verpflichtungen der Rolle in Bezug auf die Realisierung von Ressourcen geprüft und für den Teilnehmer vermerkt. Dieser muss für jede Ressource eine entsprechende Anwendungskomponente bereitstellen. Handelt es sich bei der Ressource um ein Asset oder Demand, so wird vorausgesetzt, dass die entsprechende Komponente bereits im System des Teilnehmers installiert ist. Handelt es sich bei der Ressource um eine Capability, so wird genau das Gegenteil angenommen. In

GUI Funktionen	Typ	Inhalt
Service Schemata	Tabelle	Übersicht einsatzfähiger E-Service-Schemas im Dienstleistungsnetzwerk
Register	Funktion	Hinzufügen eines gebrauchsfertigen E-Service-Schemas
Delete	Funktion	Entfernen eines E-Service-Schemas
Initiate	Funktion	Deklarieren einer E-Service Instanz (Kontaktphase beginnen)
Service Instances	Tabelle	Übersicht aller E-Service Instanzen im Dienstleistungsnetzwerk
Start Aggregation Strategy	Funktion	Aggregation der E-Service-Instanz beginnen (= E-Service erbringen)
Stop Aggregation Strategy	Funktion	Aggregation der E-Service-Instanz beenden (= E-Service abbrechen)
Delete	Funktion	Entfernen einer E-Service-Instanz (nur falls nicht aktiv)
Participants	Tabelle	Übersicht registrierter Produktionseinheiten im Netzwerk und Klienten
Register	Funktion	Registrierung eines Teilnehmers inkl. OGSI-Registry
Assign Role	Funktion	Zuweisung einer Rolle und Korrelationsinformationen i.B.a. E-Service-Instanz
Delete	Funktion	Entfernen eines Teilnehmers (nur falls unbeteiligt)
Deployment	Funktion	Anweisung zum Deployment einer zu realisierenden E-Service-Ressource
Query Registry	Funktion	Übersicht von Einträge der OGSI-Registry eines Teilnehmers
Resource Classes	Tabelle	Übersicht der aller E-Service-Ressourcen-Schnittstellen im Netzwerk
Allocated Resources	Tabelle	Übersicht alloziierter E-ServiceRessourcen Instanzen im Netzwerk
Allocate	Funktion	Manuelle Allokation einer realisierten E-Service-Ressource
Free	Funktion	Manuelle Freigabe einer alloziirten E-Service-Ressource

Tabelle 5.9.: Interaktive Funktionen des E-Service-Aggregation-Managers

diesem Fall muss ein Deployment der spezifischen Anwendungskomponente beim spezifischen Teilnehmer durchgeführt werden (`Deployment`). Hierbei kann eine automatische Durchführung der Installation durch den Engine-Manager angewiesen werden. Die verschiedenen Klassen von Ressourcen (Assets, Demands und Capabilities) des strategischen Dienstleistungsnetzwerks werden in einer eigenständigen Tabelle anhand ihrer Schnittstellenspezifikationen zur Information gelistet. Sie werden nicht eigenständig registriert, sondern leiten sich aus den registrierten Service-Schemata ab. Die letzte Tabelle verwaltet die Instanzen der Ressourcenklassen, die momentan im Kontext einer E-Service-Instanz aktiv genutzt werden. Solche Instanzen beziehen sich immer auf die zu realisierenden Ressourcen eines Teilnehmers in Bezug auf eine seiner Rollen. Spätestens wenn eine Anfrage nach einer Ressource dieser Rolle mit den entsprechenden Korrelationsinformationen vorliegt, muss deren Allokation durchgeführt werden (`Allocate`). Dies kann wiederum manuell erfolgen, wobei dann der GSH der entsprechenden Komponenteninstanz per Dialog einzugeben ist. Der Normalfall sollte jedoch die automatische Allokation mittels Grid Service Factory der Komponente sein, die der Aggregation-Manager dann eigenständig durchführt. Umgekehrt müssen Ressourcen je nach Strategie früher oder später wieder freigegeben werden (`Free`). Im Verlauf der Ausführung der E-Service-Instanz wiederholen sich die obigen Schritte bis zum Abschluss des virt. Dienstleistungsprozesses.

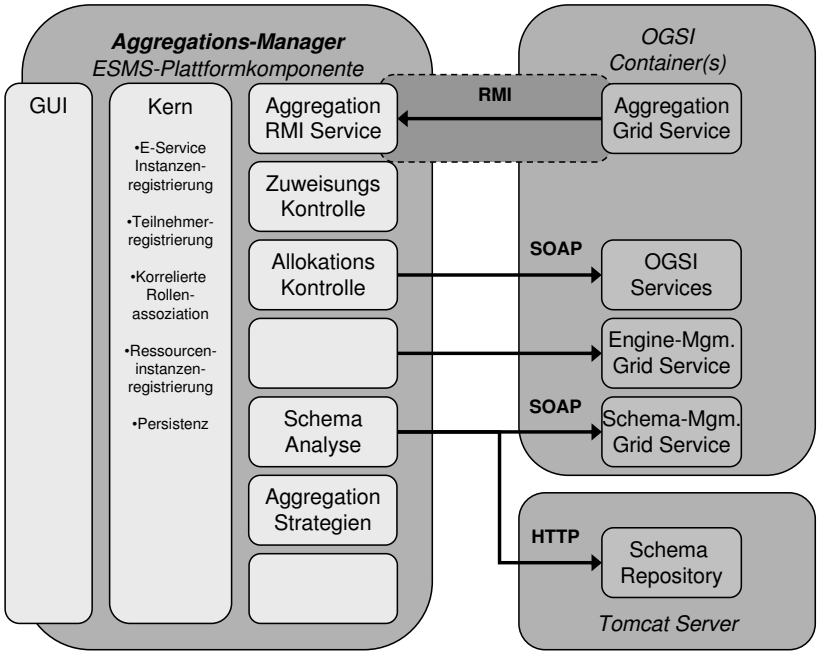

Abbildung 5.33.: FRESCO-TK Aggregation-Manager-Architektur

E-Service-Aggregation-Manager-Implementierung Der E-Service-Aggregation-Manager basiert auf einer Erweiterung der generischen Grid Connectivity-Architektur für FRESCO-TK Plattformkomponenten, die in Abb. 5.33 gezeigt wird.

Das Aggregation-Manager-Anwendungssystem läuft in einer eigenständigen Java-VM. Es implementiert die von der Plattformarchitektur geforderten Funktionen auf Basis von RMI. Ein zugehöriger GT3-Container beinhaltet zwei Proxies, die die `AggregatorPortType-` und `ParticipantRegistryPortType`-Schnittstellen durch Weiterleitung von Operationsaufrufen an RMI-Methoden implementieren.

Die Schema-Management-Funktionen basieren auf einem Objektmodell und verschiedenen Datenbanken für E-Service-Schemata und -Instanzen, die zusammen den Systemkern bilden. Das Objektmodell bildet eine erweiterte Variante des im Kontext der Plattformarchitektur vorgestellten Datenmodells für die transiente Repräsentation von E-Service-Meta-Informationen in XML-Nachrichten zwischen Plattformkomponenten.[53]. Klassen des Aggregation-Managers mit Entsprechung auf Plattformebene beinhalten Methoden zur Erzeugung einer Repräsentation im Plattformformat. Hierdurch können die Datenstrukturen des Kerns unmittelbar als Web Service-Nachrichten versendet werden.

Auf Basis des Objektmodells erfolgt mittels des Schema-Analysemoduls die Konstruktion einer Speicherrepräsentation von E-Service-Schemata. Das Analysemodul greift dazu auf das E-Service-Schema Repository zu und nutzt einen spezifischen E-

[53]Siehe hierzu Abbildung 5.17, S. 398.

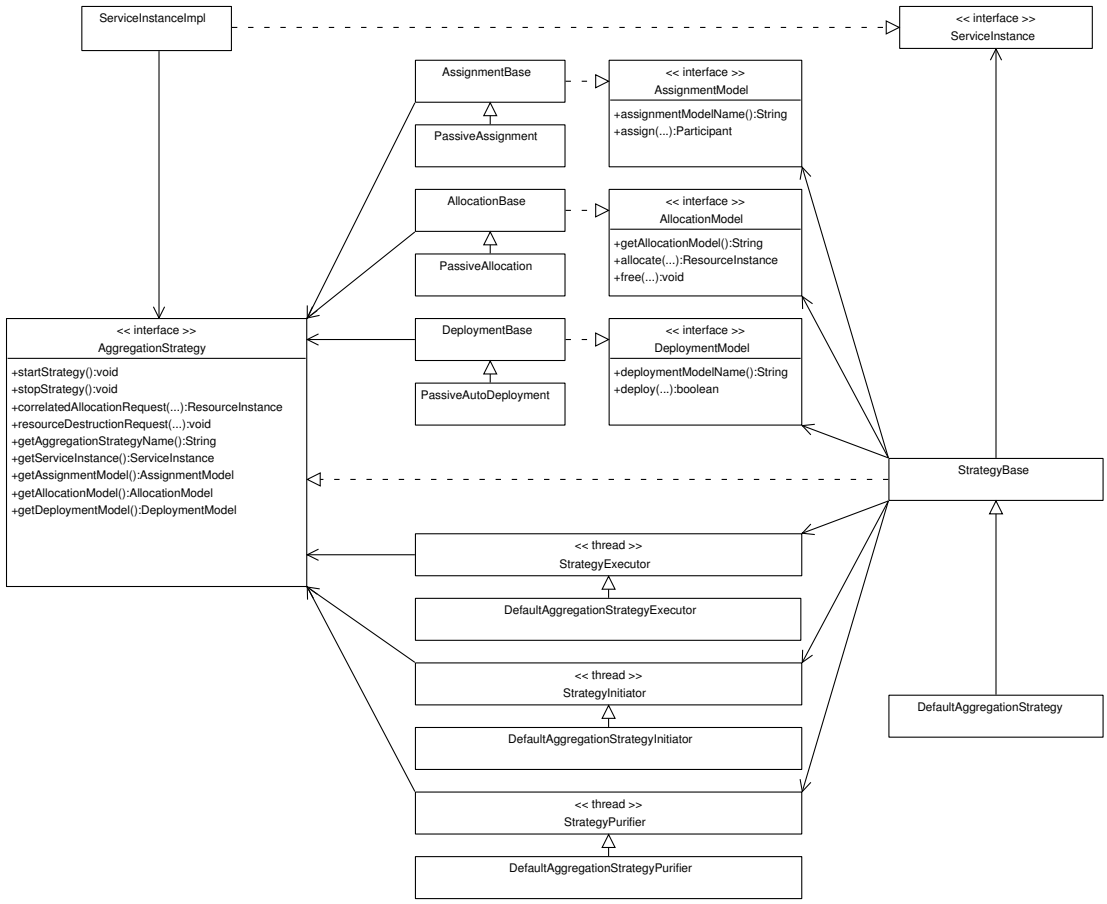

Abbildung 5.34.: Software-Rahmenwerk für Aggregationsstrategien

Service-Schema Parser für die XPDL- und WSDL- Spezifikationen im FSM-Format. Der Parser benutzt dabei den XPDL Core sowie die WSDL4J API als Bindungsrahmenwerke für die beiden Typen von XML-Dokumenten.

Die zentrale Funktion des Aggregation-Managements ist *korrelierte Rollenassoziation*; die Anfrage des GSH für eine spezifische E-Service-Instanz, Ressourcenklasse, Rolle und Korrelationsinformationen. Diese Referenz ist entweder schon vorhanden und wird dann in den Datenbanken des Kerns abgefragt, oder sie ist noch nicht vorhanden, was den Aggregationsvorgang im Sinne der spezifischen Aggregationsstrategie in Gang setzt.

Um die individuelle Erweiterung zu erlauben, bietet der Aggregation-Manager ein Software-Rahmenwerk für Aggregationsstrategie, das in Abb. 5.34 gezeigt wird. Eine Strategie (`AggregationStrategie`) ist darin durch eine Schnittstelle spezifiziert, die neben dem Start und Stop des Aggregationsvorgangs vor allem Methoden zur korrelierten Allokation und zur Ressourcenfreigabe vorgibt. Eine Basisimplementierung

(`StrategyBase`) sorgt für die Integration von Strategie in den Aggregation-Manager-Kern, wobei eine Strategie jeweils mit einer E-Service-Instanz (`ServiceInstance`) in Verbindung steht für die sie arbeitet. Die wesentlichen Bestandteile einer Strategie sind drei Typen von Threads, die unabhängig und parallel zum Rest des Aggregation-Managers den Aggregationsvorgang vollziehen. Hierbei werden drei Phasen unterschieden. Ein `StrategyInitiator` Thread startet, sobald der Aggregationsvorgang einer E-Service-Instanz gestartet wird. Ein `StrategyExecutor` Thread übernimmt jeweils eine spezifische Anfrage. Ein `StrategyPurifier` Thread erlaubt die Durchführung abschließender Aktivitäten nach Beendigung des Aggregationsvorgangs für die E-Service-Instanz. Im FRESCO-TK sind eine manuelle und eine `DefaultAggregationStrategy` realisiert.

Für die Durchführung von Rollenzuordnung, Deployment und Allokation werden einer Strategie individuelle Modelle zugewiesen. Diese erbringen stets grundlegende Operationen, die in einer Schnittstelle festgelegt sind. Die Art der Durchführung kann jedoch ganz unterschiedlich sein. Hier sind die oben beschriebenen aktiven oder passiven sowie automatischen oder manuellen Varianten implementiert. Aggregationsmechanismen basieren auf entsprechenden Kontrollmodulen, die u. a. die Anbindung externer Mechanismen der Grid- und ESMS-Plattform und die Ausführung der Operationen im Kern realisieren.

5.4. Zusammenfassung

In den vorangegangenen Sektionen wurde das vorher erarbeitete konzeptionelle Rahmenwerk zur Unterstützung virt. Dienstleistungsproduktion durch Dienstleistungsvirtualisierung mit E-Services durch ein technisches Rahmenwerk ergänzt, das praktische Mittel zur Umsetzung des E-Service-Managements mittels E-Service-Management-Systemen (ESMS) bereitstellt. Der Fokus lag auf einer Untersuchung Service-orient. ESMS in Bezug auf Entwicklungsmethoden und Ausführungsplattformen für E-Services. Dabei wurden auf der einen Seite Prinzipien zur Kombination einer allgemeinen Methodik modellgetriebener Entwicklung von E-Service-Schemata mit dem generellen Entwurf einer Grid-basierten Ausführungsumgebung für E-Service-Instanzen entwickelt. Auf der anderen Seite wurde die Kombination einer konkrete Entwicklungsmethodik für die Generierung und Transformation von FRESCO E-Service-Schema-Spezifikationen aus UML-basierten Modellen und hin zu BPEL-basiertem Code mit einer OGSI-basierte Plattform zur integrierten Entwicklung und Ausführung von E-Service-Instanzen sowohl als Verfeinerung der allgemeinen Prinzipien entworfen als auch prototypisch implementiert.

Durch die hier erarbeiteten Entwurfsprinzipien für Service-orient. ESMS wurde ein konzeptioneller Beitrag in Form von generellen Richtlinien in Bezug auf die Realisierung Service-orient. Anwendungssysteme zur Prozessintegration im Kontext virt. Unternehmen geleistet. Diese Richtlinien beinhalten Strategien in Hinblick auf

die Berücksichtigung von (I) Dynamik und Fluktuation virtueller Leistungsprozesse durch agile Entwicklung flexibler Anwendungssysteme auf Basis modellgetriebener Methodik und (II) Identität und Konsistenz virtueller Ressourcen durch Repräsentation auf Grundlage Grid-basierter Erweiterungen Service-orient. Basistechniken. Mit den Richtlinien zur modellgetriebenen E-Service-Entwicklung wurden spezifische Modelle prozessbasierter Anwendungssysteme auf verschiedenen Ebenen identifiziert und deren Beziehungen und wechselseitige Transformation im Entwicklungsprozess aufgezeigt. In den Richtlinien zur Grid-basierten Ressourcenverwaltung wurden Architekturmuster für Komponenten prozessbasierter Anwendungssysteme in Form von Ressourcen und Engines sowie deren virtuelle Laufzeitumgebung entworfen, die das Zusammenspiel verschiedener OGSA-Mechanismen zur organisationsübergreifenden Verwaltung von Prozessinstanzen vorgeben.

Mit dem Entwurf des FRESCO-ESMS und der Implementierung des FRESCO-TK wurden konkrete Beiträge in Form von Methoden und Mechanismen für eine exemplarische Realisierung der Konzepte virt. Dienstleistungsproduktion geleistet. Mit dem ESMS-Entwurf erfolgte als Erstes die Umsetzung der generellen Richtlinien zur modellgetriebenen E-Service-Entwicklung in Hinblick auf (I) den grafischen Entwurf von E-Service-PIM und -PSM nach Vorgabe des FSM auf Basis eines UML-Profils und Generierung von XPDL-basierten textuellen Repräsentationen als E-Service-Schema-Spezifikationen, (II) Generierung einer ausführbaren Form von E-Service-Schema-Spezifikationen auf Basis einer Abbildung der Workflow-Sprache XPDL und deren Service-orient. Erweiterungen im FSM auf die WS-Kompositionssprache BPEL sowie (III) die systematische Änderung von E-Service-Schema-Spezifikationen durch eine Transformationsregelsprache für XPDL und einen entsprechenden Workflow-Transformationsalgorithmus.

Als zweiter Aspekt des ESMS-Entwurf erfolgte die Umsetzung der generellen Richtlinie zur Grid-basierten Ressourcenverwaltung, wobei erweiterte Grid Service-Komponenten sowohl Anwendungskomponenten für E-Services als auch Plattformkomponenten für die Realisierung von Mechanismen zur integrierten E-Service-Entwicklung und -Ausführung bilden. Konkret resultierte dies in drei Plattformkomponenten für (I) E-Service-Schema-Management zum E-Service-Entwurf durch Registrierung, Validierung, regelbasierte Modifikation und BPEL-Transformation von E-Service-Schema-Spezifikationen, (II) E-Service Engine-Management zur E-Service-Implementierung durch automatisierte Konstruktion und Installation von E-Service Engine-Komponenten sowie (III) E-Service-Aggregation-Management zur E-Service-Ausführung durch Verwaltung von Teilnehmern und Ressourcen sowie dynamische Bindung von E-Service-Komponenten.

Schließlich wurde eine prototypische Implementierung der ESMS-Entwürfe durch das konkrete technische Rahmenwerk des FRESCO-TK durchgeführt. Dies resultierte zunächst in einer horizontalen Basisinfrastruktur für VDU. Konkrete Ergebnisse umfassen COTS-Komponenten für OGSI- und BPEL-Ausführungsumgebungen sowie

Erweiterungen in Bezug auf ESMS-Plattform- und =Anwendungs=Architektur=Spezifikationen, Software-Rahmenwerke und Programmiermodelle. Aufbauend auf die horizontale Infrastruktur resultierten vertikale ESMS-Fachfunktionen in Form der drei Plattformkomponenten sowie eines autonomen grafischen Entwurfswerkzeugs für E-Service-Modelle mit zugehörigem Code-Generator für XPDL-basierte E-Service-Schema-Spezifikationen.

6. Einsatz von E-Services im Logistikszenario

6.1. Zielsetzung

Das Ziel des 6. Kapitels ist die Anwendung und Evaluation der bis hierher erarbeiteten konzeptionellen und technischen Instrumente. Hierzu soll das FRESCO-Rahmenwerk auf seine Potenziale zur Unterstützung des in Kapitel 2 beschriebenen Logistikszenarios untersucht werden. Kapitel 6 bildet damit den zweiten Teil der praktischen Betrachtungen und schließt gleichzeitig den inhaltlichen Teil der Arbeit ab.

Zu Beginn soll in Sektion 6.2 die Evaluationsmethode beschrieben werden. Dabei werden zunächst verschiedene mögliche Varianten zur Evaluierung von IV-Techniken angesprochen. Im Hinblick auf den Gegenstand der Evaluation in der vorliegenden Arbeit wird dann die zu überprüfende Hypothese herausgestellt und die Wahl experimenteller Methoden erläutert. Dies betrifft insbesondere eine Fallstudie mit dem entwickelten FRESCO-Rahmenwerk.

Für diese Fallstudie soll in Sektion 6.3 eine Grundlage bereitet werden, indem eine erweiterte Analyse des Logistikszenarios in Bezug auf die Annahmen des konzeptionellen FRESCO-Rahmenwerks erfolgt. Dies beinhaltet die Einordnung der FreightMixer-Organisationsstruktur in das konzeptionelle Modell virtueller Dienstleistungsproduktion in FRESCO. Des Weiteren ist die im Geschäftsfall des Szenarios beschriebene Form der virtuellen Dienstleistungsproduktion mit dem FRESCO E-Service-Management-Vorgehensmodell abzugleichen. Schließlich soll die Virtualisierung des Dienstleistungsprozesses geprüft werden, indem dessen Elemente in Hinblick auf die Kategorien des konzeptionellen FRESCO VDLP-Modells analysiert und klassifiziert werden.

Im Anschluss beschreibt Sektion 6.4 die Durchführung der Fallstudie selbst und erläutert dazu das Management von E-Services in einem Durchlauf des E-Service-Entwicklungslebenszyklus im Rahmen des Geschäftsfalls. Dabei soll zwecks Detailanalyse der Aspekt der Frachtübergabe fokussiert werden. Dieser ist inhaltlich sowie in Form von abgeschlossenen VDLP-Anteilen darzustellen. Aufbauend soll die Erstellung eines E-Service-Modells und Generierung XPDL-basierter Spezifikationen erfolgen. Des Weiteren soll die Änderung dieser Capabilities in Bezug auf verschiedene operationale und strategische Aspekte mitsamt Transformation der XPDL-Spezifikationen gezeigt werden. Schließlich soll eine Betrachtung des Managements von Frachtübergabe-Engine-Komponenten mitsamt Generierung, Konstruktion und Deployment sowie der Aggregation und Ausführung eines konkreten Frachtübergabeprozesses im Rahmen des Geschäftsfalls erfolgen.

Am Ende ist in Sektion 6.5 eine Bewertung der experimentellen Ergebnisse durchzuführen. Hierbei sollen die Beobachtungen bei der Umsetzung des exemplarischen

Geschäftsfalls mittels Anwendung des FRESCO-Rahmenwerks mit dem ursprünglichen Szenario verglichen werden.

6.2. Beschreibung der Evaluationsmethode

Grundprinzip wissenschaftlichen Arbeitens ist die Absicherung von Untersuchungsergebnissen. Im Kontext einer Ingenieurdisziplin wird meist die Hypothese aufgestellt, dass eine zugrunde gelegte Theorie zu einer in irgendeiner Weise effektiven Technik führt. Diese Effektivität ist dann zu beweisen, indem geeignete Attribute der Technik gemessen werden, die den gewünschten Effekt erfassen. Die Erfassung dieser Daten erfolgt dabei durch Experimente [ZW98]. Generell wird dabei der Effekt eines *Faktors* (z. B. einer Technik) in Bezug auf spezifische Attribute ermittelt. Experimente mit festgelegten Attributen erfolgen durch spezifische *Behandlungen* von *Subjekten*. Hierbei sind Replikation, lokale Kontrolle, Beeinflussung und Zeitpunkte von Experimenten zu beachten.

In Bezug auf die verschiedenen Möglichkeiten zur experimentellen Validierung von IV-Techniken stellen Zelkowitz und Wallace [ZW97] eine Taxonomie mit 12 möglichen Ansätzen auf. Grundsätzlich unterscheiden sie zwischen *observierenden*, *historischen* und *kontrollierten* Methoden. Die observierenden Methoden *Project Monitoring*, *Case Study*, *Assertion*[1] und *Field Study* sind jeweils passive Beobachtungen von Subjekten ohne Einfluss auf die aktuellen Behandlungen. Historische Methoden, bei denen in ebenfalls passiver Weise Daten aus vergangenen Behandlungen genutzt werden, umfassen *Literature Search*, *Legacy*, *Lessons Learned* und *Static Analysis*. Schließlich bilden die kontrollierten Methoden *Replicated*, *Synthetic*, *Dynamic Analysis* und *Simulation* aktive Beobachtungen mit direkter Beeinflussung der Behandlung.

Der vorliegenden Arbeit liegt die Hypothese zugrunde, dass Service-orient. Techniken effektiv zur Unterstützung von Planungs- und Steuerungsfunktionen des Produktionsmanagements in virt. Dienstleistungsunternehmen eingesetzt werden können. Als Grundlage hierfür wurde eine Theorie aufgestellt, die Service-orient. Dienstleistungsvirtualisierung und Service-orient.-Entwicklung von virtuellen Dienstleistungen zur Planung und Steuerung der Produktion in virt. Dienstleistungsunternehmen als konzeptionelles Rahmenwerk vereint. Auf Basis dieser Theorie wurde eine Technik Service-orient. Entwicklung und Ausführung von virtuellen Dienstleistungen im Kontext der IV-Infrastruktur von virt. Dienstleistungsunternehmen als technisches Rahmenwerk realisiert.

Um zu zeigen, dass die entwickelte Technik effektiv ist, bedient sich die vorliegende Arbeit zweier Experimente. Das erste Experiment kann als Lessons Learned-Methode eingestuft werden und liegt als externe Vorgabe zugrunde. Hierbei wurden qualitative Daten verschiedener vergangener Projekte ausgewertet und als FreightMixer-Szenario

[1] Hierbei ist der Beobachter gleichsam Subjekt, was vermieden werden sollte.

zusammengefasst.[2]. Die Daten dieses Experiments geben die relevanten Attribute einer effektiven Technik zur virtuellen Dienstleistungsproduktion vor und legen deren Sollwerte fest. Zugleich gibt das Szenario den Rahmen für die Durchführung des zweiten Experiments vor. Dieses wird mittels einer observierenden Methode im Rahmen der vorliegenden Arbeit durchgeführt, die als spezielle Form der Case Study aufgefasst werden kann. Hierbei wird der Effekt betrachtet, den die Anwendung des entwickelten Rahmenwerks unter den Bedingungen des Szenarios hat. Das Rahmenwerk ist dabei gleichzeitig Faktor und Subjekt. Die Behandlung wird alleine durch die externe Vorgabe des Szenarios bestimmt. Zur Validierung der Hypothese ist dann zu zeigen, dass die Daten des zweiten Experiments dem Rahmenwerk Attribute bescheinigen, die in ihrer Qualität den Vorgaben des ersten Experiments entsprechen oder diese übertreffen.

6.3. FRESCO-basierte VDL-Produktion im Logistikszenario

Die Anwendung des FRESCO-Rahmenwerks auf ein beliebiges praktisches Szenario erfordert zunächst dessen Analyse. Hierbei ist grundsätzlich zu klären, wie die Organisationsstruktur im konkreten Fall mit dem konzeptionellen organisatorischen Modell von FRESCO in Einklang gebracht werden kann. Das bedeutet, dass Rollen identifiziert und Funktionen zugeordnet werden müssen, für deren Ausübung das FRESCO-Rahmenwerk die entsprechenden Instrumente liefert. Zu den Instrumenten gehört auf oberster Ebene das FRESCO E-Service-Management-Vorgehensmodell. Dies gibt auf der einen Seite den generellen Rahmen für das Management der virtuellen Dienstleistungsproduktion auf Basis von virt. Dienstleistungen vor. Auf der anderen Seite beschreibt es die konkrete Vorgehensweise für dessen Unterstützung durch einen Service-orient. Entwicklungslebenszyklus für E-Services. Damit geht die Analyse der im Szenario vorkommenden Dienstleistungen einher. Hierbei ist deren Eignung zur Virtualisierung zu prüfen, was in erster Linie die Struktur ihres Dienstleistungsprozesses betrifft. Dieser ist derart aufzubereiten, dass er auf Basis des FRESCO E-Service-Metamodells modelliert werden kann.

Als Nächstes wird in Sektion 6.3.1 die Organisationsstruktur von FreightMixer analysiert. Im Anschluss erfolgt in Sektion 6.3.2 eine aufbauende Analyse der Transportdienstleistung im Geschäftsfall von Shoe&Shoe in Bezug auf deren Abwicklung im Kontext des FSMV und deren Virtualisierung als FRESCO E-Service.

6.3.1. Organisation der VDL-Produktion

Der betrachtete Anwendungsfall für das FRESCO-Rahmenwerk bezieht sich auf das Szenario des Logistikdienstleisters FreightMixer.[3] FreightMixer betreibt ein

[2]Die Ergebnisse dieses Experiments wurden in Sektion 2.4.2 ausführlich dargestellt.
[3]Siehe Sektion 2.4.2, S. 76.

logistisches Dienstleistungsnetzwerk. Dieses ist als virtuell organisiertes Hub-and-Spoke-Transportnetzwerk ausgelegt. FreightMixer nimmt darin die Rolle eines reinen Informationsvermittlers ein. Die Leistung des Netzwerks ergibt sich aus den logistischen Teilleistungen von Netzwerkunternehmen. Sie bestehen aus den klassischen Haupt- und Nebendienstleistungen der Logistik. Das Netzwerk selbst basiert auf verschiedenen geschlossenen „E-Markets“ des Logistiksektors. Diese begründen einen Vertrauenskontext und bieten FreightMixer grundlegende Instrumente für das Netzwerkmanagement. Auf dieser Basis agiert FreightMixer als Broker und Leader eines orthogonalen Dienstleistungsnetzwerks. Dessen Netzwerkstruktur ist dynamisch, zentriert und intern hierarchisch geführt. In dessen Rahmen plant und steuert FreightMixer die virtuelle Produktion von Gütertransporten. Die Planung von Gütertransporten basiert auf fundierten Methoden zur Planung des Hub-and-Spoke-Systems [ZW00]. Sie weist eine klar begrenzte Flexibilität in Bezug auf optionale Direktverkehre auf. Zudem soll die Erweiterung von Transportdienstleistungen durch Integration optionaler Nebendienstleistungen möglich sein. Im Rahmen dieser Freiheitsgrade konfiguriert FreightMixer virtuelle Leistungsbündel zum Angebot auf den E-Markets. Auf Anfrage von Kunden werden Leistungsbündel individualisiert und im Netzwerk als virt. Dienstleistungsproduktionsnetzwerke realisiert. FreightMixer formiert dieses auf Basis der E-Markets und regelt die Koordination der kooperativen Vorgänge einzelner Produktionseinheiten durch Planung und Steuerung des ganzheitlichen DLP.

Das FRESCO-Rahmenwerk beruht auf der Grundlage eines umfangreichen konzeptionellen Modells. Dieses legt die Konzepte von Dienstleistungen, deren Produktion in einfachen und virt. Unternehmensnetzwerken sowie insbesondere deren Virtualisierung zur Produktionsplanung und -steuerung genau fest. Die Anwendung der FRESCO-Technologie in einem konkreten Szenario bedingt die Identifikation der FRESCO-Basiskonzepte in diesem Szenario. Dabei wird nicht vorausgesetzt, dass das Szenario der FRESCO-Organisationsstruktur genau entspricht. Die Identifikation von Konzepten im Szenario entspricht vielmehr einer Analyse von dessen Organisationsstruktur. Als Voraussetzung für die Anwendung der FRESCO-Technologie sollten sich dabei wesentliche Übereinstimmungen erkennen lassen. Falls dies zutrifft, ist die Organisationsstruktur weiterzuentwickeln. Das FRESCO-Modell virt. Dienstleistungsproduktion gibt dabei das Ziel vor.

Im FreightMixer-Szenario liegt ein strategisches Netzwerk vor, in dem Dienstleistungen durch virt. Dienstleistungsproduktionsnetzwerke erbracht werden. Damit ist die notwendige Organisationsstruktur zum Einsatz der FRESCO-Technologie grundsätzlich schon gegeben. Kunden und Netzwerkunternehmen sind hierin unmittelbar als `VDPN-Clients` und `-Provider` zu identifizieren. Demnach wird vorausgesetzt, dass Kunden wie Shoe&Shoe eine Dienstleistung einsehen und auswählen, diese dann in koordinierter Weise nutzen und schließlich inhaltlich verwerten. Netzwerkunternehmen vollziehen die Produktionsplanung und -steuerung für Logistikleistungen der Knotenebene und fügen sich dabei in eine übergeordnete koordinative Rege-

lung ein. FreightMixer spielt simultan die Rollen des `VDPN-Brokers` und `-Koordinators`. Dies beinhaltet Produktionsplanung und -steuerung logistischer Leistungsbündel auf Netzwerkebene. All dies trifft für das Szenario zu. Der wesentliche Nutzen dieser Zuordnung besteht darin, dass das konzeptionelle Modell eine Realisierungsstrategie für die Rollenfunktionen aufzeigt, die nun grundsätzlich auf das FreightMixer-Szenario übertragen werden kann. Diese Realisierungsstrategie stützt sich auf Konzepte eines virtuellen Dienstleistungsbegriffs. Hierbei werden zum einen funktionale Kategorien abgeleitet, die sich bei der Dienstleistungsvirtualisierung als Nutzeneffekt des IT-Einsatzes ergeben. Zum anderen wird gezeigt, wie sich diese funktionalen Kategorien zur Unterstützung der Produktionsplanung und -steuerung auf Knoten- und Netzwerkebene von VDPN einsetzen lassen.

Aufbauend auf die abstrakten Betrachtungen des konzeptionellen VDPN-Modells definiert das erweiterte FRESCO-Modell eine verfeinerte Sicht der Realisierungsstrategien, die insbesondere auch als technisches Rahmenwerk mit konkreten Methoden und Mechanismen umgesetzt sind. In der FRESCO-Sicht ändern sich die Zuständigkeiten der einzelnen Rollen geringfügig. Daneben werden die Rollenfunktionen in Bezug auf deren Bezug zu Konzepten des FRESCO VDL-Modells in Form von Assets, Capabilities und Demands konkretisiert. In diesem Sinne müssen FRESCO-Clients wie Shoe&Shoe (technisch) in der Lage sein, Interaktionsmuster in Form von Capabilities zu verarbeiten und selbst entsprechende Kontaktpunkte in Form von Demands zur Verfügung zu stellen. FreightMixer als FRESCO-Broker obliegt die Entwicklung von Interaktionsmustern in Form von Capabilities in Abstimmung mit FRESCO-Providern. Ferner ist es FreightMixers Aufgabe, die Dienstleistungsproduktion auf Netzwerkebene mittels Durchsetzung von Capabilities auf Basis gegebener Techniken zu steuern. Die Aufgabe von FRESCO-Providern besteht zunächst in der eigenverantwortlichen Planung und Steuerung der Produktion auf Knotenebene in Form von Assets. ggf. können Interaktionsmuster für diese Assets in Form von Capabilities geplant werden. Diese Capabilities müssen dann aber in Abstimmung mit FreightMixer geplant und gesteuert werden. Dieser Aspekt geht potenziell über die bisherige Kooperationsbeziehung der Netzwerkunternehmen im Szenario hinaus. Ob die dadurch notwendigen organisatorischen Erweiterungen im Szenario realisierbar sind, lässt sich nicht nachvollziehen. Es lässt sich aber feststellen, dass die Verwendung von Provider Capabilities im FRESCO-Modell eine Option darstellt. Falls dies unerwünscht oder organisatorisch nicht zu realisieren ist, kann darauf verzichtet werden. In diesem Fall müssen FRESCO-Provider lediglich Assets planen und steuern. Zudem kann festgestellt werden, dass die technischen Anforderungen, die mit den Rollenfunktionen im FRESCO-Modell verbunden sind, im FreightMixer-Szenario gegeben sind. Der Einsatz einer „E-Service“-Technologie ist hierin nämlich explizit vorgesehen und diese Technologie liegt in Form des technischen FRESCO-Rahmenwerk auch vor.

Als Fazit der organisatorischen Analyse kann festgehalten werden, dass die Organisationsstruktur der virt. Dienstleistungsproduktion in FRESCO mit dem FreightMixer-Szenario kompatibel ist. Es lassen sich sowohl Akteure zur Übernahme der notwendigen Rollen identifizieren als auch deren grundsätzliche Fähigkeit zur Ausübung der damit verbundenen Funktionen verifizieren. Das FreightMixer-Netzwerk erfüllt somit die organisatorischen Voraussetzungen, um das FRESCO-Rahmenwerk zur virt. Dienstleistungsproduktion anzuwenden. Dies setzt dann jedoch voraus, dass auch das spezifische FRESCO-Dienstleistungsmodell sowie die zugehörige Form des Produktionsmanagements Anwendung finden. Nun wird untersucht, inwiefern dies im FreightMixer-Szenario praktikabel ist.

6.3.2. Management der VDL-Produktion

Der wesentliche Aspekt der virtuellen Dienstleistungsproduktion in FRESCO besteht in der Repräsentation virtueller Dienstleistungen als E-Services. Das Management der Dienstleistungsproduktion bedient sich dabei in Bezug auf die Produktionsplanung und -steuerung des FRESCO E-Service-Management-Vorgehensmodells. Als Nächstes wird in Sektion 6.3.2.1 die Struktur des FRESCO E-Service-Management-Vorgehensmodells mit dem Ablauf des FreightMixer-Dienstleistungsmanagements im Allgemeinen sowie mit dem Ablauf des Shoe&Shoe Geschäftsfall im Speziellen verglichen. Im Anschluss wird in Sektion 6.3.2.2 die Virtualisierung von FreightMixer-Dienstleistungen sowie deren Repräsentation auf Basis des FRESCO E-Service-Metamodells am Beispiel des Shoe&Shoe Geschäftsfalls untersucht.

6.3.2.1. VDL-Lebenszyklus

Das FreightMixer-Szenario beschreibt den konkreten Fall eines virt. Dienstleistungsproduktionsnetzwerks. Um dessen virt. Dienstleistungsproduktion zu verstehen, ist es hilfreich, diese in die Kategorien des allg. VDPN-Lebenszyklus einzuordnen.[4] Diese „Normierung“ der Vorgänge erlaubt dann im nächsten Schritt eine Abbildung auf das FRESCO E-Service-Management-Vorgehensmodell.[5] Hierdurch wird ersichtlich, wie die FRESCO-Methoden und -Mechanismen im konkreten Fall Anwendung finden können. Das Resultat ist ein FRESCO-spezifisches Produktionsmanagementmodell für FreightMixer, das wiederum mit den Vorgaben des Szenarios verglichen und bewertet werden kann.

FreightMixer produziert Logistikdienstleistungen auf Basis eines virt. Hub-and-Spoke-Systems. Das Unternehmen agiert als Broker und Leader in einem strategischen Dienstleistungsnetzwerk multipler Logistikdienstleister. Ihm obliegt hier die Produktionsplanung und -steuerung auf Netzwerkebene. Grundlage bildet eine strategische Planung des Hub-and-Spoke-Systems mit Hub, Spokes und Transportverbindungen

[4] Siehe hierzu 4.4.1.2 und Abb. 4.28, S. 289.
[5] Siehe hierzu 4.4.2.2 und Abb. 4.31, S. 306.

sowie der damit direkt und indirekt einhergehenden logistischen Teildienstleistungen. Auf Basis der fundamentalen Hub-and-Spoke-Infrastruktur werden dabei zum einen potenzielle Direktverkehre und deren Bedingungen auf Basis gegebener Planungsmethoden bestimmt.[6] Zum anderen werden mit den Mitteln des Dienstleistungsmarketings verschiedene Leistungsbündel potenzieller logistischer Gesamtdienstleistungen ermittelt.[7] In beiden Fällen werden dabei virt. Dienstleistungsproduktionsnetzwerke identifiziert, was der 1. Stufe von deren Lebenshistorie entspricht. Weitere Planungsaktivitäten sind der 2. Stufe zuzuordnen. Konzeption, Anforderungsanalyse und Entwurf des VDPN und der damit einhergehenden Dienstleistung betreffen hier insbesondere die Gestaltung der logistischen Leistungsprozesse und die Regelung der Koordination verschiedener Logistikdienstleistungen in diesen Prozessen. Gleichzeitig muss FreightMixer auch die Selektion konkreter Logistikdienstleister für das strategische Netz als Teil des Netzwerkmanagements vollziehen. An diesem Punkt ist die Vorkontaktphase für FreightMixer-Dienstleistungen beendet. Deren nachfolgende Kontaktphase wird durch Anbahnung eines Geschäftsfalls eingeleitet, was der 3. Stufe der Lebenshistorie entspricht. Hier geht eine Kundenanfrage für eines der Leistungsbündel zusammen mit spezifischen Anforderungen bei FreightMixer ein. Im konkreten Fall ist das die Anfrage von Shoe&Shoe. Auf dieser Basis nimmt FreightMixer ggf. Anpassungen der Produktionsplanung in Hinblick auf spezifische Anforderungen vor, was im Lebenszyklusmodell den Beginn von Stufe 4) markiert. Für die angepasste Planung erfolgt dann eine Allokation von Logistikdienstleistern für das VDPN. Im Szenario geschieht das durch einen Auktionsmechanismus.[8] Im Anschluss an die Vereinbarung geht das VDPN in die operative Phase der Nutzung über. Hierzu ist die geplante Dienstleistung auf Stufe 4) als E-Service zu implementieren. Auf dessen Basis wird die Produktion der Dienstleistung im VDPN gesteuert. Kooperative Aktivitäten der Logistikdienstleister und des Kunden werden durch FreightMixer freigegeben, kontrolliert und gesichert. Sie laufen dann jede für sich in Selbstabstimmung des jeweiligen Logistikdienstleisters bzw. Kunden ab. Zudem übernimmt FreightMixer wesentliche Anteile der Informationslogistik zwischen den Akteuren. Am Ende der Produktion beginnt in Stufe 5) die Nachkontaktphase der Dienstleistung und das VDPN wird stillgelegt, d. h. die Formation konkreter Logistikdienstleister löst sich auf. Der Kunde nutzt nun die Wirkung der Dienstleistung, was sich im Fall von Shoe&Shoe als Präsentation der gelieferten Schuhe darstellt. FreightMixer evaluiert

[6] Die Planung des Hub-and-Spoke-Systems und dessen potenzieller Direktverkehre wird im Rahmen der vorliegenden Arbeit nicht vertieft. Hier sind nur die daraus resultierenden Rahmenbedingungen geregelter Flexibilität von Interesse. Für eine Betrachtung der Planung siehe [ZW00].

[7] Das Dienstleistungsmarketing wird weder im Szenario vertieft noch soll es hier Gegenstand der Betrachtung sein. Grundsätzliche Aspekte des Dienstleistungsmarketings werden z. B. bei Corsten [Cor98] beschrieben.

[8] Die konkrete Allokation auf Basis von Auktionen begründet eine komplementäre Fragestellung und wird im Folgenden nicht weiter vertieft.

auf Stufe 5) den Produktionsverlauf, nimmt ggf. Anpassungen an der Planung vor und tritt mit diesen in die 2. Stufe des nächsten Lebenszyklus ein.

Ein allg. (Service-orient.) E-Service-Entwicklungslebenszyklus zeigt auf, wie sich das Management der Dienstleistungsproduktion auf allen Stufen des VDPN-Lebenszyklus mittels (Service-orient.) IT-Repräsentation des Dienstleistungsprozesses unterstützen lässt.[9] Diese allgemeinen Konzepte werden im FRESCO E-Service-Management-Vorgehensmodell für den spezifischen Fall Service-orient. E-Services im Sinne des FRESCO E-Service-Metamodells konkretisiert. Das FSMV liefert hierbei konkrete Anweisungen zur Durchführung des Produktionsmanagements auf Netzwerkebene mittels Methoden und Mechanismen eines FRESCO E-Service-Management-Systems. Hierbei wird der Kernbereich des VDPN-Lebenszyklus von Stufe 2b) bis Stufe 5) fokussiert, in dem sich die wesentlichen Wirkungen der Technologie zeigen. Dem geht im allg. Modell die Stufe 2) voraus. Hiermit ist die Vorgehensweise zur Planung und Analyse von E-Services im Einklang mit Konzeption, Anforderungsanalyse und Entwurf der Dienstleistungen im strategischen Netzwerk verbunden. Der Entwurf der Dienstleistung als Geschäftsprozess bildet die primäre Anforderung an den nachfolgenden Entwurf des FRESCO E-Service. Im FreightMixer-Szenario erfolgt dies im Rahmen der strategischen Planung des Hub-and-Spoke-Systems sowie aufbauender logistischer Leistungsbündel. An diesem Punkt beginnt der FRESCO-spezifische E-Service-Entwicklungslebenszyklus für FreightMixer. Dies beinhaltet folgende Schritte:

- **Stufe 2b:** Der initiale Schritt des Vorgehensmodells besteht im Entwurf von E-Service-Modellen für FreightMixer-Dienstleistungen. Aus den oben geschilderten organisatorischen Gründen wird eine E-Service-Variante gewählt, die auf Provider-spezifische Capabilities verzichtet. Hierdurch entfällt die Abstimmung paralleler Entwurfsvorgänge zwischen FreightMixer und verschiedenen Logistikdienstleistern in Bezug auf globale und lokale Interaktionsmuster. Stattdessen identifiziert FreightMixer passende Assets mit den Mitteln der E-Markets. Auf dieser Grundlage erfolgt der Entwurf von Capabilities und Demands im Sinne der Vorgaben des Dienstleistungsentwurfs. Der Entwurf erfolgt durch Werkzeugunterstützung zur Modellierung und Validierung. Die Ergebnisse werden von FreightMixer als E-Service-Schema-Spezifikationen im strategischen Dienstleistungsnetzwerk zur Einsicht durch die Logistikdienstleister veröffentlicht. Zugleich werden entsprechende Dienstleistungsangebote auf den E-Markets zur Einsicht durch Kunden zugänglich gemacht.

- **Stufe 3:** Aus den Dienstleistungsangeboten auf dem E-Market resultieren Kundenanfragen. Diese können wie im Fall von Shoe&Shoe individuelle Anforderungen beinhalten. FreightMixer passt in diesem Fall das Leistungsbündel und die E-Service-Schema-Spezifikation situationsbedingt an. Dazu werden die

[9] Ein allgemeines Modell hierfür wurde in Sektion 4.4.1.2 erstellt. Siehe auch Abb. 4.30, S. 298.

FRESCO-Mechanismen zur Änderung von E-Service-Schema-Spezifikationen verwendet. Im Fall von Shoe&Shoe ist das Leistungsbündel um eine Versicherungsteilleistung zu ergänzen. Die Koordination mit dem Versicherer muss durch entsprechende Interaktionen geregelt werden. Dies wird durch Transformationsregeln erreicht, die entsprechende Änderungen am E-Service-Schema nach a priori entworfenen Konsistenzkriterien durchführen. Für die resultierenden Assets werden dann Logistikdienstleister im FreightMixer-Netzwerk identifiziert. Dies erfolgt orthogonal auf Basis von E-Markets. Bei Übereinkunft aller Parteien erfolgt eine entsprechende Konfiguration der operativen Produktionssteuerung. Hierzu werden E-Service-Schema-Spezifikation und Teilnehmer registriert. Zu einem ausgemachten Zeitpunkt wird dann die operationale Phase eingeleitet. Im Fall von Shoe&Shoe geschieht das unmittelbar, denn der Transport soll schnellstmöglich erfolgen.

- **Stufe 4:** Der Übergang in die operationale Phase erfolgt mithilfe verschiedener FRESCO-Mechanismen zur Implementierung von E-Service-Instanzen. Dies beinhaltet zunächst den physikalischen Entwurf von Steuerprozessen für die Capabilities von FreightMixer als BPEL-Schemata. Des Weiteren gehört dazu die Konstruktion von E-Service Engines sowie deren Installation auf FreightMixers ESMS-Plattform. Von Shoe&Shoe und den Logistikdienstleistern wird erwartet, dass sie Demands und Assets als E-Service-Ressourcen auf ihren lokalen ESMS-Plattformen bereitgestellt haben. Auf diese Weise ist der zeitliche Aufwand der Implementierung zu vernachlässigen. FreightMixer startet dann den VDLP der Transportdienstleistung und im Zuge dessen werden erste Interaktionen mit Teilnehmern durchgeführt. Im Beispiel beinhaltet dies die Abfrage von FreightMixers Logistik-Planungs-AS nach der ersten Teilstrecke, die Freigabe dieses Transports beim zuständigen Frachtführer und die Benachrichtigung von Shoe&Shoe über die Zeit der Abholung. Im weiteren Verlauf setzen sich VDLP und DLP parallel bis zum Abschluss fort. ggf. sind dabei manuelle Managementaktivitäten einzubeziehen, die außerhalb der Kontrolle des VDLP ablaufen sollen. Diese können entweder durch E-Service-Ressourcen gekapselt werden oder als manuelle Anteile des Aggregationsmechanismus einfließen. Z. B. kann die Aggregation von E-Service-Ressourcen für die Interaktion mit der Versicherung rechtzeitig im Voraus einen manuellen Administrationsvorgang zum Abschluss einer Police in Gang setzen.

- **Stufe 5:** Nach Abschluss des Transports kann FreightMixer die Metadaten der Ausführung analysieren. Dies beinhaltet z. B. Protokolle der E-Service Engines oder individuell gesammelte Audit-Informationen. Die Ergebnisse der Analyse fließen in die Planung des Hub-and-Spoke-Systems und die Entwürfe des E-Service ein. Für Letzteres bietet FRESCO wiederum das Instrument des regelbasierten Transformationsmechanismus. Im exemplarischen Geschäftsfall könnte

die Analyse z. B. einen überhöhten Aufwand auf Grund der Tracking-Funktion diagnostizieren. FreightMixer kann dann Transformationsregeln formulieren, um in allen Interaktionsmustern die Benachrichtigungen von Kunden zu entfernen. Das derart geänderte E-Service-Schema bildet die Basis für optimierte Dienstleistungsangebote auf den E-Markets.

Die oben beschriebene Abbildung zeigt, dass der FreightMixer VDPN-Lebenszyklus grundsätzlich durch das FRESCO E-Service-Management-Vorgehensmodell unterstützt wird. Gleichzeitig ist zu bemerken, dass der generische FRESCO-Ansatz hier nicht ganz ohne situationsbedingte Modifikationen anwendbar ist. Das umfassende Konzept zur parallelen Abstimmung und Ausführung von Interaktionsprozessen bzw. Capabilities durch multiple VDPN-Akteure ist für das betrachtete Szenario potenziell zu komplex. Stattdessen reicht in diesem Fall eine vereinfachte Variante mit organisatorisch zentralisierter Führung aus. Auch die Phasenstruktur des Vorgehensmodells wurde in Bezug auf die VU-Anbahnung optimiert. Das FSMV zeigt sich dabei jedoch als ausreichend flexibel.

Ein potenzielles Hindernis ist die Forderung nach Implementierung von Demands als E-Service-Ressourcen durch den Kunden. Auf der einen Seite erlaubt dies eine Integration mit seiner internen IT-Infrastruktur und Anwendungssystemen. Auf der anderen Seite verhindert dies aber auch die spontane Nutzung neuer Dienstleistungen, da der Zeitaufwand für die Entwicklung der E-Service-Ressourcen zu hoch wäre. Eine potenzielle Lösung dieses Problems besteht in der Implementierung von Demands als Web-Anwendungssystemen. Dies wäre von FreightMixer durchzuführen und müsste sich in die bestehende modellgetriebene Entwicklung eingliedern. Die Integration von Kunden in den VDLP würde dann manuell mittels eines Web-Browsers erfolgen. Auf Kundenseite entfällt dadurch die Implementierung. Diese Lösungsmöglichkeit soll im Rahmen der vorliegenden Arbeit nur erwähnt, nicht jedoch realisiert werden. Im Folgenden sei daher angenommen, dass der Demand im Shoe&Shoe Geschäftsfall einem Standard entspricht und Shoe&Shoe daher grundsätzlich über entsprechende Schnittstellen verfügt.

6.3.2.2. VDL-Struktur

Im Zuge der hier stattfindenden Analyse zur Anwendbarkeit der FRESCO E-Service-Technologie als Unterstützung für das Produktionsmanagement im FreightMixer-Szenario, fehlt zum Abschluss noch eine Untersuchung der Virtualisierung von FreightMixer-Dienstleistungen. Bisher zeigten sich grundsätzliche Übereinstimmungen in Bezug auf die Organisationsstruktur und das Vorgehensmodell. D. h. zum einen, dass die Akteure des Szenarios die Rollen der virt. Dienstleistungsproduktion ausfüllen. Zum anderen genügt der Ablauf der virt. Dienstleistungsproduktion dem exemplarischen Geschäftsfall. Im Zuge dessen muss die FreightMixer-Dienstleistung dann grundsätzlich als FRESCO E-Service repräsentiert werden. Es ist nun am Beispiel der

exemplarischen Transportdienstleistung zu zeigen, ob und wie die Überführung in einen FRESCO E-Service gelingt.

Im exemplarischen Geschäftsfall des Szenarios fragt Shoe&Shoe eine individuelle Transportdienstleistung bei FreightMixer an. Es handelt sich dabei um einen interkontinentalen Stückguttransport von großem Wert als Expresszustellung. Das Dienstleistungsnetzwerk von FreightMixer verfügt über das Potenzial zur Erbringung eines entsprechenden Leistungsbündels. FreightMixer selbst besitzt das Know-How zur Planung des Transports im Rahmen seines Hub-and-Spoke-Systems. Zudem hat FreightMixer die Integration von Versicherungsleistungen als virtuelles Leistungsbündel antizipiert und ist in der Lage, seinen fundamentalen DLP entsprechend anzupassen. Das Ergebnis strategischer Planung liegt u. a. als Geschäftsprozessmodell vor. Hierbei existiert zumindest ein Basismodell des fundamentalen DLP sowie u. a. ein Prozessfragment für die Integration des Versicherungsleistung. Der resultierende Gesamtprozess für die von Shoe&Shoe nachgefragte Dienstleistung wurde in Abb. 2.30[10] gezeigt. Der dort dargestellte Prozess ist zunächst im Sinne der fundamentalen DLP-Struktur gegliedert. Entsprechend werden Teilprozesse im Front und Back Office sowie zur Steuerung unterschieden. Ferner sind verschiedene funktionale Anteile farblich gekennzeichnet. Der fundamentale Dienstleistungsprozess des Transports setzt sich aus den weißen und hellgrauen Aktivitäten zusammen. Dabei markieren hellgraue Aktivitäten die Vorbereitung und den Abschluss des Transports in Abstimmung mit dem Start- und Ziellager des Kunden. Die weißen Aktivitäten dienen hingegen der Abstimmung verschiedener Frachtführer auf den Teilstrecken zwischen Hub und Spokes. Schließlich wird die logistische Zusatzleistung der Versicherung durch die dunkelgrauen Aktivitäten in den Prozess integriert. Die Aktivitäten im DLP beziehen sich im Wesentlichen auf die physikalischen Verrichtungen des Transports. Dazu kommen administrative Aufgaben der logistischen Steuerung und Versicherung. Allesamt sind durch die üblichen logistischen Informationsflüsse verknüpft.[11]

Das FRESCO-Dienstleistungsmodell basiert auf einer Abwandlung der strukturellen Basiskategorien. Hierbei werden Steueraktivitäten sowie die gesamte Prozessstruktur als zusammenhängende Capabilities eingeordnet. Back und Front Office Aktivitäten werden hingegen als Assets und Demands klassifiziert.[12] Die resultierende DLP-Struktur bietet insbesondere Vorteile in Bezug auf deren Virtualisierung. Dies erfolgt für Capabilities mittels informationstechnischer Abbildung der Interaktionsprozesse. Ferner werden die Kontaktpunkte der Interaktion mit Assets und Demands in eine IT-Repräsentation überführt. In FRESCO wird die Virtualisierung des DLP mittels Service-orient. Techniken im Detail durch das FRESCO E-Service-Metamodell definiert. Aufbauend auf einem fundamentalen Workflow-Prozessmodell werden Capabilities auf WS-Interaktionsprotokolle und -Kompositionsschemata abgebildet. Web Services

[10] Siehe S. 83.
[11] Siehe Sektion 2.4.1.2, S. 71.
[12] Siehe hierzu Sektion 4.2.2.2 sowie Abb. 4.12, S. 235.

selbst bilden die Kontaktpunkte von Assets und Demands ab. Mit diesem Hintergrund erfolgt die Virtualisierung eines DLP in zwei Schritten. Zunächst ist dessen Struktur in Hinblick auf das FRESCO-Dienstleistungsmodell zu interpretieren. Dabei sind die primären Strukturelemente zu identifizieren. Als Zweites sind verschiedene Aspekte in Bezug auf die Interaktionsprozesse im DLP zu analysieren. Dazu gehört u. a. die Identifikation des ganzheitlichen Interaktionsprozesses und dessen Gliederung in abgeschlossene Teile. Auch in Bezug auf die Kontaktpunkte sind zusammenhängende Einheiten zu bestimmen. Eng damit verbunden ist die Identifikation von Rollen der Interaktion, die nicht unbedingt direkt mit denjenigen der Geschäftsprozesse konform gehen müssen. Allgemein sind die fundamentalen Service-orient. Entwurfsprinzipien starker Kohäsion und loser Kopplung anzustreben. Ausgehend von dieser Analyse ist dann der Entwurf eines E-Service-Modells mittels der Elemente des FSM zu vollziehen. Hierbei ist eine Vorgehensweise zu empfehlen, die bei den fundamentalen Strukturelementen ansetzt. Aufbauend kann eine schrittweise Verfeinerung bis hin zur E-Service-Schema-Spezifikation erfolgen.

Für den Fall des Shoe&Shoe Dienstleistungsprozesses sind die FRESCO-Strukturelemente leicht zu erkennen. Assets sind hier primär durch Transporte auf Teilstrecken sowie durch die Versicherungsfunktion gegeben. Hinzu kommt die logistische (Routen-)Planung, die zwar von FreightMixer selbst erbracht wird, im E-Service-Modell jedoch als abgeschlossener funktionaler Aspekt berücksichtigt werden sollte. Dies gilt insbesondere unter dem Gesichtspunkt, dass in einem verfeinerten Prozessmodell noch weitere logistische Funktionen identifiziert werden können. Die Gruppierung der Back Office-Funktionen kann durch entsprechende Rollen erreicht werden.

Auf der Seite des Front Office sind Funktionen des Order-Tracking und der Lagerverwaltung abzugrenzen und durch entsprechende Rollen kenntlich zu machen. Der wesentliche Aspekt der Virtualisierung liegt aber in der expliziten Erfassung und Strukturierung der DLP-Interaktionsabläufe durch Capabilities. Hierbei gehen die vier primären Aufgaben aus dem Geschäftsprozessmodell hervor. Hierzu gehören die Vor- und Nachbereitung von Transporten insbesondere durch Koordination von Start- und Ziellager mit den Frachtführern der ersten und letzten Teilstrecke. Ferner ist im Hub-and-Spoke-System die Übergabe von Transporten verschiedener Teilstrecken abzustimmen. Schließlich müssen in Problemfällen Abstimmungen zwischen Kunden, Frachtführern und ggf. der Versicherung stattfinden. Jede Aufgabe ist mit einem Interaktionsprozess verbunden, der die beteiligten Rollen mittels Steuerung von Interaktionen koordiniert. Eine weitere Strukturierung kann dann in Bezug auf die inhaltliche Nähe der Teilbereiche erfolgen. In diesem Sinne werden die beiden Prozesse der Hub-and-Spoke-Koordination zu einer Capability zusammengefasst. Die übrigen Prozesse bilden eigenständige Capabilities. Auch Capabilities können auf konzeptionelle Rollen verteilt werden, die eine Option zur späteren Dezentralisierung der Steuermechanismen eröffnen. Im Beispiel wird mit jeder Capability eine eigenständige Rolle verbunden. Das Ergebnis der beschriebenen Abbildung bildet eine Grobstruktur des FRESCO-spezifischen virt. Dienstleistungsprozesses für die

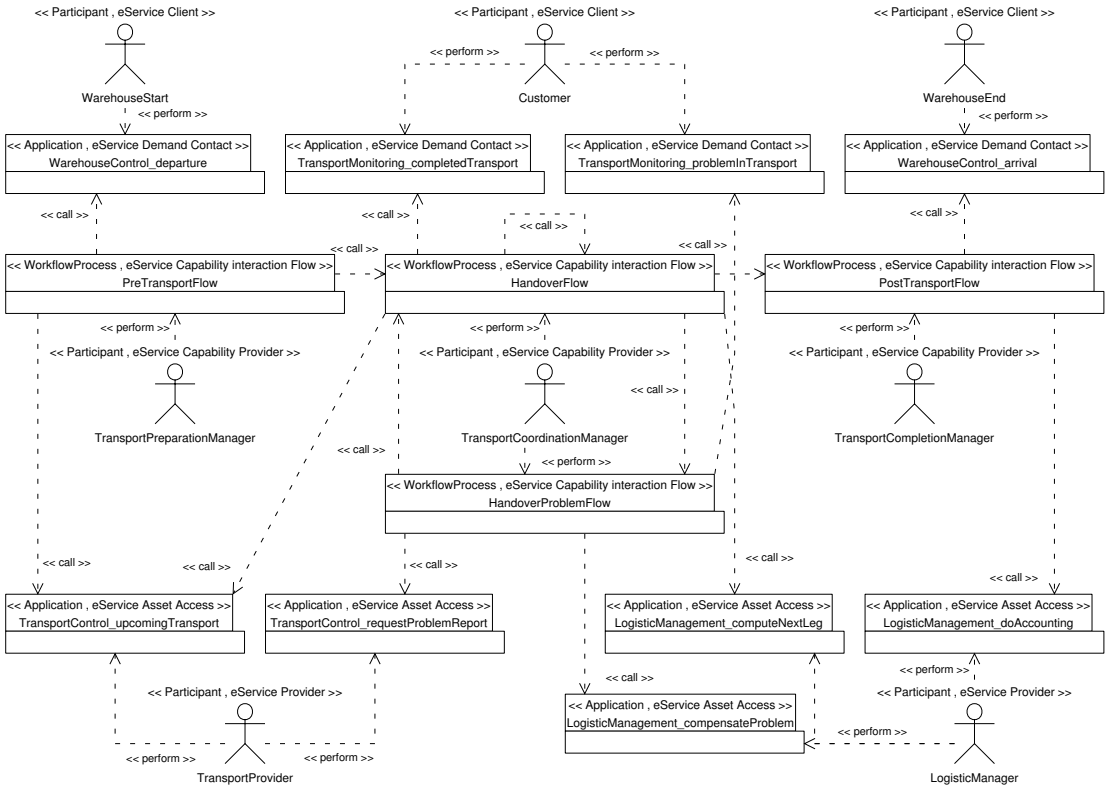

Abbildung 6.1.: Grobstruktur des FreightMixer-VDLP

Shoe&Shoe Dienstleistung. Abbildung 6.1 zeigt diese Struktur als UML-Diagramm auf Basis des FRESCO-Profils. Zur besseren Übersicht wurde die Versicherungsteilleistung ausgelassen.

6.4. E-Service-Management im Geschäftsfall

Die Betrachtung der vorangegangenen Sektionen hat die Anwendbarkeit der FRESCO E-Service-Technologie zur Unterstützung des Produktionsmanagements im FreightMixer-Szenario auf theoretischer Ebene gezeigt. Hierbei wurde das konzeptionelle FRESCO-Rahmenwerk auf die spezifische Form virt. Dienstleistungsproduktion im FreightMixer-Netzwerk abgebildet und in Bezug auf Organisationsstruktur, Vorgehensmodell und Dienstleistungsstruktur evaluiert. In den folgenden Sektionen soll nun die Anwendung der FRESCO E-Service-Technologie auf praktischer Ebene gezeigt werden. Hierbei wird das technische FRESCO-Rahmenwerk in seiner prototypischen Implementierung als FRESCO-TK eingesetzt. Es soll nun demonstriert werden, wie mit dessen Mechanismen und Werkzeugen spezifische Anteile des FreightMixer-Produktionsmanagements realisiert werden können. Dies erfolgt im Rahmen eines

Durchlaufs des E-Service-Entwicklungslebenszyklus im exemplarischen Geschäftsfall von Shoe&Shoe.

Im Folgenden wird in Sektion 6.4.1 zunächst der Entwurf von E-Service-PIM und -PSM gezeigt. Dabei wird der Frachtübergabeaspekt im Kern der Transportdienstleistung fokussiert. Im Anschluss erfolgt eine Betrachtung der regelbasierten Änderung von E-Services in Sektion 6.4.2. Am Ende beschreibt Sektion 6.4.3 die Implementierung und Ausführung der individuellen E-Service-Instanz.

6.4.1. Entwurf und Spezifikation

Der initiale Schritt des FRESCO E-Service-Management-Vorgehensmodells besteht im Entwurf eines E-Service-Modells nach Vorgabe eines detaillierten Dienstleistungsentwurfs. Dieser sollte hierfür als Geschäftsprozessmodell vorliegen. Die Grundlagen des E-Service-Entwurfs für das FreightMixer-Szenario wurden bereits in der letzten Sektion erarbeitet. Hierfür wurde das als Vorwärtsprozess in Grobstruktur vereinfachte Geschäftsprozessmodell des FreightMixer-Szenarios als Vorgabe genommen. Das Ergebnis war der Entwurf einer Grobstruktur für den virt. Dienstleistungsprozess von FreightMixer-Dienstleistungen. Diese bildet den Kernaspekt eines plattformunabhäng. Modells für FreightMixer-E-Services.

Die VDLP-Grobstruktur zeigt einen Querschnitt von Aspekten des FreightMixer-PIM. Der systematische PIM-Entwurf beinhaltet darüber hinaus eine detaillierte Betrachtung verschiedene Bereiche, die sukzessive spezifiziert werden müssen. Dies beinhaltet auf oberster Ebene Pakete für den Interaktionskontext (≪`eService Interaction Context`≫), die Shell (≪`eService Shell`≫) sowie die Interaktionsmuster (≪`eService Capability Interaction Pattern`≫). Die wichtigsten Aspekte des Interaktionskontextes bilden die Abstraktion von Teilnehmern der Dienstleistung bzw. Produktionseinheiten des VDPN durch Rollen (≪`eService Role`≫) sowie die Definition von Kontaktpunkten zur Interaktion mit Assets, Demands und Capabilities (≪`eService Interaction Contact`≫). Ein weiterer Aspekt besteht in der Deklaration von Datentypen (≪`eService Data Format`≫), die jeweils einen Teil einer Nachricht zum Versand an die Kontaktpunkte bestimmen. Die Shell bietet eine generische Container-Struktur für die Erfassung spezifischer Modellaspekte. Als spezifische Anteile werden hierin zudem die Capabilities des E-Service (≪`eService Capability`≫) deklariert. Hierbei werden für jede Capability die verantwortliche Rolle und die zugehörigen Interaktionsmuster festgelegt. Interaktionsmuster definieren schließlich Abläufe von Interaktionen im Rahmen von Capabilities, die durch deren verantwortliche Rollen gesteuert werden. Die hierzu spezifizierten Interaktionsprozesse stützen sich auf die im Interaktionskontext festgelegten Rollen und Kontaktpunkte.

Im Fall von FreightMixer wird der E-Service-Entwurf zunächst für die fundamentale Transportdienstleistung ohne sekundäre Logistikdienstleistungen vorgenommen. Wie schon bei der Herleitung der VDLP-Grobstruktur (Abb. 6.1) in der letzten Sektion beschrieben wurde, können hier Assets, Demands und Capabilities aus der

Geschäftsprozessstruktur (Abb. 2.30, S. 83) abgeleitet werden. Konkret lassen sich Kontaktpunkte in Bezug auf die Abwicklung von Transportstrecken, logistische Planung, Abstimmung mit Start- und Ziellager, Tracking sowie Steuerung des DLP bei Initialisierung, Hub-and-Spoke-Durchlauf und Abschluss des Transports identifizieren. Auf Basis dieser Kategorien werden die entsprechenden Rollen `TransportProvider`, `LogisticManager`, `WarehouseStart`, `WarehouseEnd`, `Customer`, `TransportPreparationManager`, `TransportCoordinationManager` und `TransportCompletionManager` definiert. Jede Rolle ist mit einem oder mehreren Kontaktpunkten verbunden. Die Kontaktpunkte im Zusammenhang mit Assets und Demands wurden in Abb. 6.1 gezeigt. Hinzu kommen Kontaktpunkte der Capabilities, die in der Abbildung aus Gründen der Übersichtlichkeit ausgelassen wurden. Ebenfalls nicht gezeigt sind die zugehörigen Datentypen. Da das Szenario hierzu nur wenige Vorgaben macht, wurden nur einfache Typen für Kennungen von Transportstrecken, -status, -problemen und Kunden sowie Zeitangaben etc. berücksichtigt. Rollen, Kontaktpunkte und Datentypen zusammen ergeben den Interaktionskontext.

Der Interaktionskontext ist als Paket zusammen mit den Paketen einzelner Interaktionsmuster in die Shell des FreightMixer E-Service-Modells eingebunden. Neben dieser Container-Funktion werden im Shell-Paket auch informale Metadaten des E-Service wie Angaben zur Versionierung, Konformität, Erstellungsdatum, Besitzer etc. erfasst. Die wichtigste Funktion besteht jedoch in der Deklaration der drei Capabilities `TransportPreparation`, `TransportCoordination` und `TransportCompletion`. Diese setzen die entsprechenden Steuerfunktionen des FreightMixer-Geschäftsprozesses mitsamt der Prozessstruktur des logistischen Informationsflusses um. Die Capabilities werden in der Shell auf oberster Ebene Rollen zugeordnet, die im Interaktionskontext definiert wurden. In einer zweiten Ebene wird jeder Capability eine Menge von Interaktionsmustern zugeordnet. Hierbei beinhalten die `TransportPreparation` und `-Completion` Capabilities jeweils eine und die `TransportCoordination` Capability zwei Interaktionsmuster. Letztere umfassen – wie schon beschrieben – die inhaltlich komplementären Interaktionsmuster zur Koordination von Frachtführern im Hub-and-Spoke-System während des Normalbetriebs und bei Problemfällen. Die konkrete Struktur der Muster wird jeweils in einem eigenständigen Paket spezifiziert, das die Shell referenziert. Das im letzten Kapitel gezeigte Bild des FRESCO-Entwurfswerkzeugs gibt einen Eindruck der interaktiven Modellierung von Interaktionskontext und Shell.[13] Dort werden fundamentale Zusammenhänge der Shell in einem UML-Klassendiagramm verdeutlicht. Im Modellbereich ist u. a. das Interaktionskontext-Paket (`Transport_context`) mit allen Kontaktpunkten und Rollen des Modells zu sehen. Zudem werden hier die Pseudoprozesse des Shell-Pakets (`Transport_context`) zur Deklaration von Capabilities gezeigt.

Der nächste Schritt der Modellerstellung besteht im detaillierten Entwurf der Interaktionsmuster. Der detaillierte VDLP von FreightMixer in seiner Gesamtheit

[13] Siehe Abb. 5.21, S. 408.

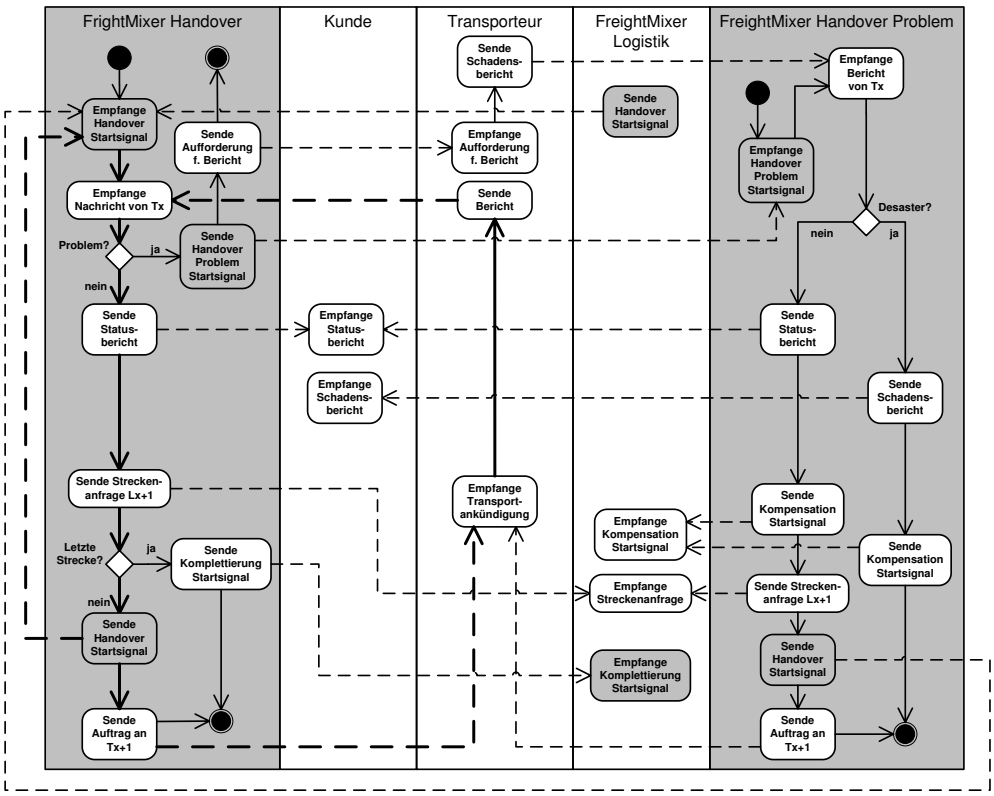

Abbildung 6.2.: Transportkoordination (Handover) im FreightMixer-VDLP

ist jedoch inhärent komplex. Eine detaillierte Analyse und Spezifikation dieses Gesamtprozesses ist im Rahmen der vorliegenden Arbeit nicht praktikabel. Dies ist jedoch auch nicht notwendig. Zur konkreten Darstellung der Funktionsweise des E-Service-Rahmenwerks reicht die detaillierte Betrachtung von Teilaspekten des DLP. Hierbei soll die Abwicklung zusammengesetzter Transporte im Rahmen des Hub-and-Spoke-Systems fokussiert werden. Insbesondere wird die Übergabe von Fracht zwischen einzelnen Frachtführern inkl. Fehlerbehandlung im Detail aufgezeigt.[14]

Die diesbezüglichen Geschäftsprozesse lagen im FRESCO-Projekt als externe Anforderung der Projektpartner vor. Hieraus wurde eine Choreografie abgeleitet, die in Abbildung 6.2 gezeigt wird. Der Ablauf beruht auf zwei kooperierenden Steuerprozessen, die am linken und rechten Rand in hellgrau dargestellt sind. Links steuert der Handover-Prozess die Frachtübergabe zwischen zwei Teilstrecken im Hub-and-Spoke-Netz. Der Prozess zur Übergabe zwischen zwei Teilstrecken x und $x+1$ wird angestoßen, sobald Transporteur T_x den Transport auf Strecke x begonnen hat (Empfange Handover-Startsignal). Den Auftrag hierzu erhält T_x entweder im

[14]Für die Frachtübergabe wird im Folgenden zum Teil der englische Begriff *Handover* verwendet.

Zuge der Transportvorbereitung, die hier nicht gezeigt ist, oder vom vorangegangenen Handover-Prozess. FreightMixer wartet dann auf eine Nachricht von T_x. Diese Nachricht signalisiert entweder die Ankunft des Transports am Ende von x oder ein Problem während des Transports. Im Falle der Ankunft sendet FreightMixer einen Statusbericht an den Kunden und eine Anfrage nach Informationen für den nächsten Streckenabschnitt $x + 1$ an die Logistikabteilung. Falls $x + 1$ die letzte Teilstrecke ist, wird eine Nachricht zum Start der Transportkomplettierung versendet und der Prozess ist beendet. Ansonsten wird ein neuer Handover-Prozess angestoßen und eine Nachricht an den Transporteur T_{x+1} zum Start von Teilstrecke $x+1$ abgeschickt. Dieser sendet dann eine Nachricht an den nächsten Handover-Prozess.

Falls T_x am Anfang des Handover-Prozesses eine Nachricht an FreightMixer sendet, die ein Problem signalisiert, wird die Übergabe der Steuerung an den Handover-Problem-Prozess eingeleitet. Dazu wird im Handover-Prozess zunächst ein Signal zum Start des Handover-Problem-Prozesses versandt. Im Anschluss folgt noch eine Nachricht an den Transporteur T_x mit der Aufforderung zur Abgabe eines Problemberichts. Dann endet der Handover-Prozess. Der Handover-Problem-Prozess wartet zunächst auf die Ankunft des Problemberichts. Falls es sich bei dem Problem um ein Desaster handelt, das einen Abbruch nötig macht, so startet FreightMixer eine nicht näher beschriebene Form der Kompensation als Abschluss des DLP. Falls es sich lediglich um eine Verspätung handelt, wird der Transport fortgesetzt. Hierzu wird zunächst der Kunde benachrichtigt. Es folgt eine Anfrage zur Kompensation, was je nach Grad des Problems eine nicht näher beschriebene Wiedergutmachung bewirkt (z. B. Preisnachlass). Danach wird der Prozess genau wie im Falle des normalen Handover zu Ende geführt.

Die kooperierenden Geschäftsprozesse zur Steuerung der Frachtübergabe dienen als Vorlage für den detaillierten Entwurf von zwei Interaktionsmustern der `Transport-Coordination` Capability. Der erste Schritt besteht dabei in der Definition der Interaktionen. Hierbei ist jeweils festzulegen, ob es sich aus der Perspektive der Capability um ein- oder ausgehende Nachrichten handelt und ob der Interaktionspartner Client, Provider oder Koordinator ist und der Nachrichtenaustausch demnach mit einem Demand, einem Asset oder einer Capability erfolgt. Im E-Service-Modell sind dann entsprechende Interaktionen zu spezifizieren. Als nächstes ist die Prozessstruktur mittels Transitionen abzubilden. Abbildung 6.3 und 6.4 zeigen dies für die beiden ≪`eService Capability Interaction Flow`≫-Entwürfe von FreightMixer.[15] In den Entwürfen sind Interaktionen zwischen den Prozessen von Capabilities enthalten. Es handelt sich entweder wie bei `start_handover` um eingehende Interaktionen in Form von ≪`eService Flow Takeover`≫ oder wie bei `start_next_handover_flow` um ausgehende ≪`eService Flow Handover`≫-Interaktionen. Zudem finden sich Interaktionen mit

[15] Die Entwürfe beinhalten neben der Abbildung der vorgegebenen Geschäftsprozess-Choreografie noch zusätzliche Erweiterungen zur Integration einer optionalen Versicherungsteilleistung. Dieser Aspekt wird in der nächsten Sektion erläutert.

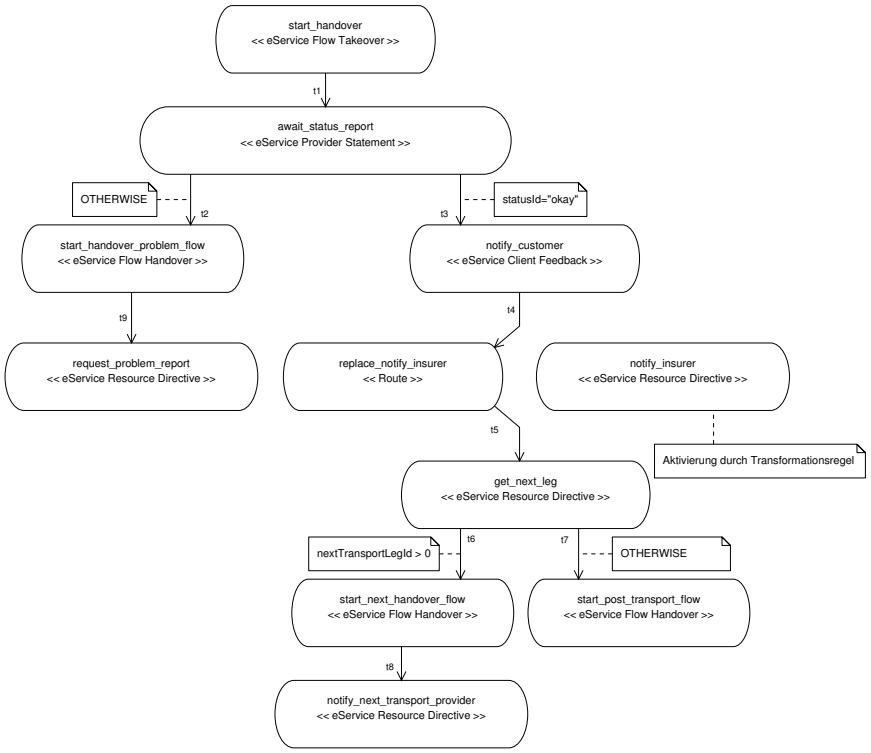

Abbildung 6.3.: ≪`eService Capability Interaction Flow`≫ für Handover

Demands und Assets. Dies sind ein- (≪`eService Provider Statement`≫) oder ausgehende Interaktionen (≪`eService Ressource Directive`≫) mit Providern und ausgehende Interaktionen (≪`eService Client Feedback`≫) mit Clients. Die Prozesslogik, z. B. bei Verzweigungen, wird an diesem Punkt zunächst informell notiert.

Der bis hierher erreichte Detaillierungsgrad des Entwurfs kann schon als komplettes plattformunabhäng. Modell angesehen werden. Ein solches Modell kann dann z. B. als Grundlage für den weiterführenden kollaborativen Entwurf dienen. Dies ist im Fall von FreightMixer jedoch nicht nötig, da hier alle Capabilities autonom entworfen werden. Der Entwurf kann daher zum plattformspezif. Modell übergehen. Hierbei sind die verschiedenen Aspekte des Modells in allen Paketen derart zu verfeinern, dass für jeden ≪`eService Capability Interaction Flow`≫ ein ≪`Web Service Orchestration Process`≫ vorliegt.

Abbildung 6.5 zeigt Erweiterungen im Zusammenhang mit dem ≪`eService Ressource Directive`≫-Element `notify_next_transport_provider`. Dieser Fall ist besonders interessant, da sich hier ein Beispiel für eine korrelierte Rolle zeigt. Der Teilnehmer in der Rolle des `TransportProvider` kann auf jeder Teilstrecke unterschiedlich sein. Zudem ist nicht a priori bekannt, wie viele Teilstrecken bei einem Transport nötig sind. Die notwendigen Rollen können daher nicht statisch festgelegt wer-

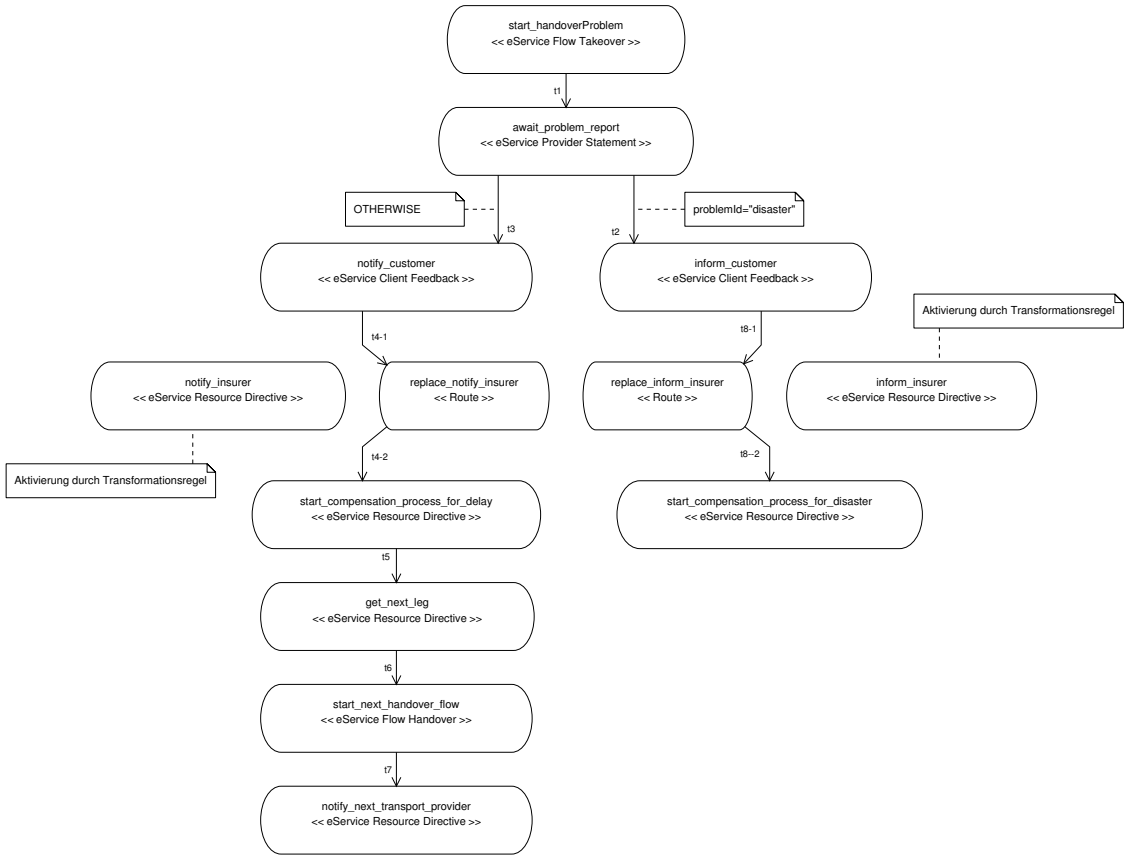

Abbildung 6.4.: Handover-Problem ≪`eService Capability Interaction Flow`≫

den. Stattdessen muss die Rolle `TransportProvider` mit der Kennung der Teilstrecke korrelieren. Dies ist zunächst bei der Definition der Rolle mittels Definition entsprechender Korrelationsinformationen festzulegen. Für den `TransportProvider` sind das die obligatorische E-Service-Instanz-Kennung (`ServiceIdType`) sowie die Kennung der Teilstrecke (`TransportLegIdType`). Als Zweites sind entsprechende Verfeinerungen der Interaktionsprozesse vorzunehmen. Generell sind dort u. a. Prozessvariablen, Transitionsbedingungen, Rolle und Kontaktpunkte für eingehende Interaktionen sowie generell eine Verfeinerung aller Interaktionen zu bestimmen. In Bezug auf die Interaktionen ist u. a. zu bestimmen, aus welchen Prozessvariablen sich die Nachricht zusammensetzt. Neben den eigentlichen Teilen der Nachricht sind am Anfang Meta-Informationen zu ergänzen. Das ist bei ausgehenden Interaktionen immer die E-Service-Instanz-Kennung. Hinzu kommt eine Konstante, die den Namen der Rolle und die Abbildung der Interaktionsparameter auf die Korrelationsinformationen der Rolle enthält (`next_transport_provider_performer_constant`). Diese Angabe wird im Performer-Ausdruck der Interaktion wiederholt. Im gezeigten Fall wird in Bezug auf die Korrelationsinformationen der `TransportProvider`-Rolle aus dem ersten Parameter

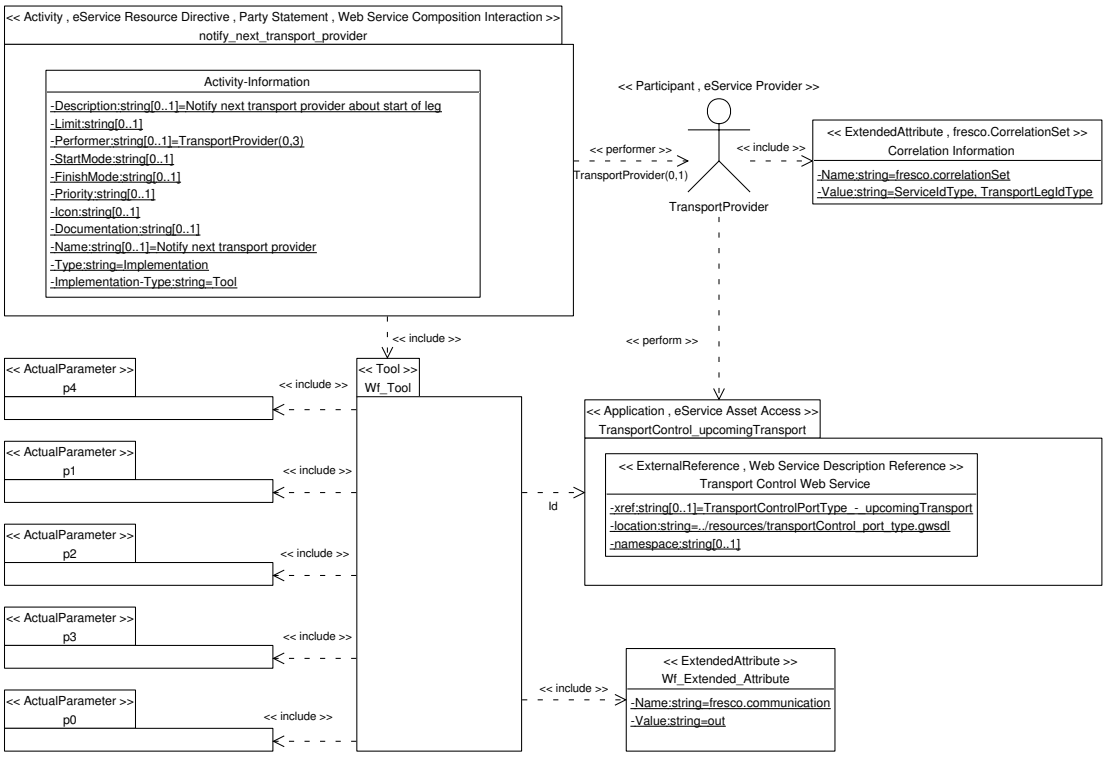

Abbildung 6.5.: PSM-Details der Interaktionen zur Anweisung von Teilstrecken

(P0) die E-Service-Instanz-Kennung (`serviceId`) und aus dem vierten Parameter (P3) die Kennung der Teilstrecke (`nextTransportLegId`) entnommen. Eine weitere wichtige Verfeinerung bezieht sich auf Kontaktpunkte. Hier müssen zum einen alle Interaktionen überhaupt mit einem Kontaktpunkt verknüpft werden, wozu das `Tool`-Element dient. Für alle Kontaktpunkte muss darüber hinaus eine Beschreibung in Form von WSDL referenziert werden. Kontaktpunkte ausgehender Interaktionen sind im Interaktionskontext vorgegeben. Dies ist für die betrachtete Interaktion der ≪`eService Asset Access`≫ `TransportControl-upcomingTransport`. Den verfeinerten Elementen des plattformunabhäng. Modell wird jeweils der Stereotyp ihrer Detailstufe zugeordnet. Die verfeinerte ≪`eService Ressource Directive`≫ wird somit in Bezug auf das WS-Interaktionsprotokoll zum ≪`Party Statement`≫ bzw. in Bezug auf das WS-Kompositionsschema zur ≪`Web Service Composition Interaction`≫.

Sind auf diese Weise alle ≪`Web Service Orchestration Process`≫-Entwürfe kompletiert worden, kann das plattformunabhäng. Modell mit dem Generator des FRESCO-Entwurfswerkzeugs in seine textuelle XPDL-Repräsentation überführt werden. Ergänzt durch die zugehörigen Schnittstellenbeschreibungen und angeordnet in der Archivstruktur ergibt sich die E-Service-Schema-Spezifikation der FreightMixer-

Dienstleistung.[16] Ein Ausschnitt dieser Spezifikation für den Handover-Interaktionsprozess ist in Anhang A beigefügt.

6.4.2. Systematische Änderung

Die E-Service-Schema-Spezifikation bildet die Basis der Transportdienstleistung von FreightMixer. Hieraus werden zunächst Angebote für E-Markets abgeleitet. Bei entsprechender Nachfrage werden nach Vorgabe des Modells E-Service-Instanzen implementiert und ausgeführt. Unmittelbar davor und danach ist es ggf. notwendig oder wünschenswert, die E-Service-Schema-Spezifikation im Zuge einer Anpassung an äußere Umstände zu ändern. Diese Umstände können sich auf Anforderungen von Kunden beziehen, die durch die E-Service-Schema-Spezifikation nicht erfüllt werden. In diesem Fall muss die Spezifikation rasch an diese Anforderungen angepasst werden, damit der Kunde möglichst schnell bedient werden kann. Eine andere Art von Umständen sind Erkenntnisse über Defizite der Spezifikation, die z. B. aus der Analyse von deren Einsatz gewonnen wurden. In diesem Fall ist es wünschenswert, eine explizite Änderungsstrategie zu formulieren. Deren Auswirkungen können umfangreich sein und sollten sich in systematischer Weise umsetzen lassen, so dass diesbezügliche Inkonsistenzen vermieden werden. Beide Fälle der Änderung werden im Folgenden für das FreightMixer-Szenario erläutert.

Bei jeder Form der regelbasierten Änderung besteht der erste Schritt in der Konzeption einer Transformationsstrategie. Hierbei sind zum einen Strukturen des Schemas zu identifizieren, die geändert werden sollen. Zum anderen ist die Art der Änderung mithilfe von Transformationsregeln zu planen. Je nach Strategie und Situation der Änderung kann die Spezifikation der angestrebten Prozessstruktur dabei auf unterschiedliche Art zwischen Schema und Regeln aufgeteilt sein. Theoretisch ist es möglich, durch Regeln komplette (Teil-)Prozesse mit Variablen, Rollen, Kontaktpunkten, Aktivitäten, Transitionen usw. zu generieren. Auf der anderen Seite kann eine Regel auch dazu genutzt werden, in einer kompletten Prozessstruktur ausschließlich Modifikationen an vorhandenen Elementen zu vollziehen. Die Wahl einer Transformationsstrategie geht eng mit der Anwendung der Transformation einher. Hierbei können grundsätzlich Schema-Variation und -Reengineering unterschieden werden. Schema-Variation dient der schnellen Umsetzung von Kundenanforderungen durch geplante Anpassung von Schemata. Hierbei werden Transformationsregeln im Voraus geplant und können bei Bedarf unmittelbar angewendet werden, um eine Variante zu realisieren. Schema-Reengineering dient der ad hoc-Änderung von Schemata entweder zur Anpassung an nicht geplante Kundenanforderungen oder zur Verbesserung von Schemata im Anschluss an deren Nutzung. Es kann festgestellt werden, dass sich die Generierung kompletter Prozessstrukturen vor allem zum Schema-Reengineering eignet. Hier ist die Transformation ein Entwurfswerkzeug

[16] Siehe hierzu Sektion 5.3.4.1, S. 413.

und erlaubt z. B. die wiederholte Spezifikation komplexer Strukturen. Beim Schema-Reengineering kommt auch die Modifikation vorhandener Prozesse zum Einsatz. Dies hilft insbesondere bei der systematischen Änderung komplexer Schemata und dämmt die Entstehung von Inkonsistenzen ein. Bei der Schema-Variation sollte hingegen überwiegend die Modifikation schon vorhandener Strukturen im Schema verwendet werden. Dadurch können vorausgeplante Strukturen z. B. mittels Modifikation des Kontrollflusses aktiviert werden. Auf diese Weise sind potenzielle Varianten Teil des Schemas und können im Kontext des VDLP geplant werden. Sie sind durch VDLP-Partner einzusehen. Trotzdem beinhaltet der aktive Teil des VDLP stets nur die notwendigen Bestandteile und ist in Bezug auf eine konkrete Variante optimiert.

Der erste Fall der Änderung im FreightMixer-Szenario ergibt sich aus der Forderung von Shoe&Shoe nach Versicherung des Transports. Diese logistische Nebenleistung erfordert verschiedene Interaktionen im Verlauf des VDLP, die in der Basisvariante der E-Service-Schema-Spezifikation nicht enthalten sind. Abbildung 6.6 zeigt die Interaktionen mit der zusätzlichen Rolle des Versicherers während der Transportkoordination als schwarze Aktivitäten. Dies beinhaltet im Wesentlichen dessen Benachrichtigung über den Verlauf des Transports. In diesem Sinne werden dem Versicherer während des Transports Statusberichte gesendet. Die wichtigste Interaktion erfolgt jedoch im Falle eines Desasters. Dann wird ein Schadensbericht gesendet, der den Schadensfall bei der Versicherung einleitet.

Die Integration der beschriebenen Interaktionen in die E-Service-Schema-Spezifikation soll nun mithilfe der regelbasierten Transformation erfolgen. Für den Geschäftsfall von Shoe&Shoe müssen dabei Regeln entworfen werden, die eine geplante Variante des Schemas für versicherte Transporte erzeugen. Diese Änderung des Basisschemas ist nicht einmalig, sondern erfolgt immer dann, wenn ein Bedarf nach dieser Variante besteht. Die Variante ist dadurch ein optionaler Teil des Schemas. Aus diesem Grund sollte sie zusammen mit dem Schema entworfen werden und dort zur Abstimmung von VDPN-Teilnehmern ersichtlich sein. Eine indirekte Spezifikation der optionalen Prozessstrukturen innerhalb der Regeln ist nicht wünschenswert. Auf der anderen Seite soll die Basisvariante des Schemas möglichst einfach bleiben. Die Integration optionaler Interaktionen in die Prozessstruktur würde zusätzliche bedingte Verzweigungen nötig machen, die nicht erwünscht sind. Dies gilt besonders dann, wenn multiple Varianten berücksichtigt werden sollen.

Die hier gewählte Lösung basiert nun auf einer kombinierten Spezifikation passiver Schema-Elemente und modifizierender Transformationsregeln. Mit passiven Schema-Elementen sind hier solche gemeint, die nicht in den Kontroll- und Datenfluss des Prozesses einbezogen sind. Diese Elemente werden bei der Generierung der BPEL-Schemata nicht berücksichtigt. Sie wirken sich daher weder auf die E-Service-Instanz-Implementierung noch -Ausführung negativ aus. Die besagten Elemente waren für den Fall von Interaktionen mit dem Versicherer schon in den Aktivitätsdiagrammen der letzten Sektion enthalten. FreightMixer plant hier zunächst die Schema-

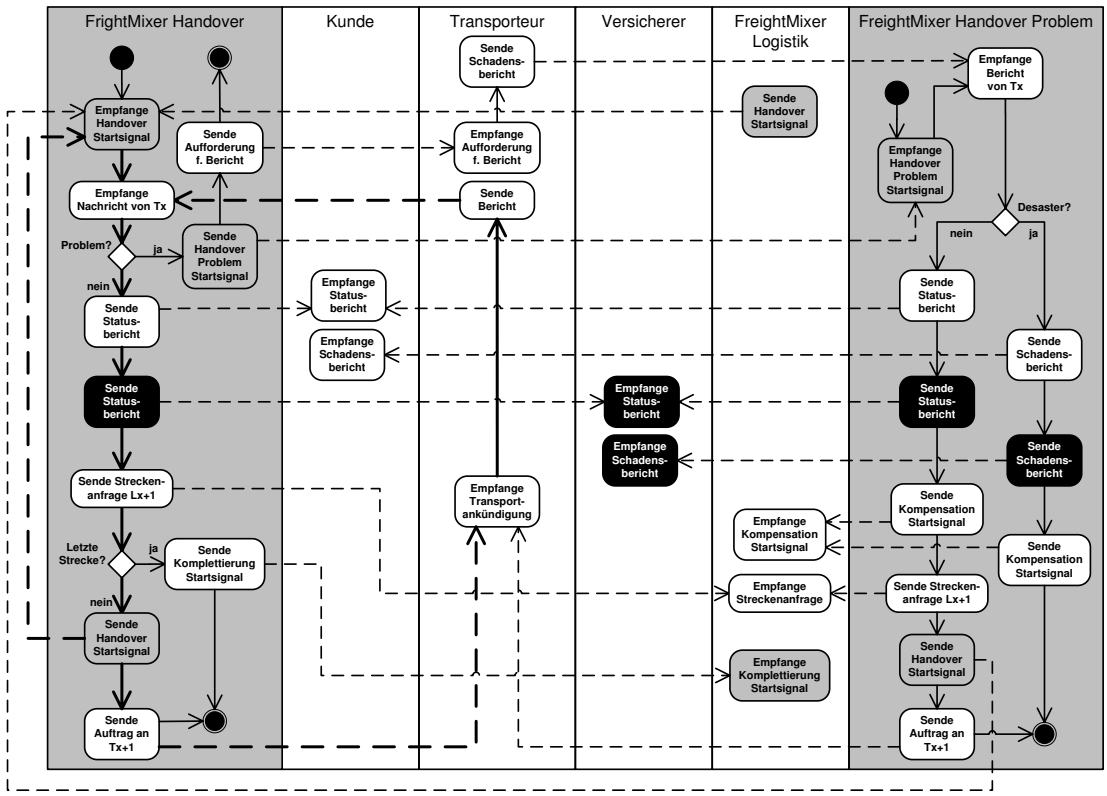

Abbildung 6.6.: Erweiterte Transportkoordination im FreightMixer-VDLP

Variante mit allen zugehörigen Aspekten im Voraus. Dann werden die optionalen Interaktionen durch eine `Route`-Aktivität als Platzhalter ersetzt. Alle Transitionen im Zusammenhang mit den optionalen Interaktionen werden mit deren Platzhaltern verbunden. Die Interaktionen und alle damit zusammenhängenden Elemente wie z. B. die `Insurer` Rolle und verschiedene Kontaktpunkte bilden dann passive Elemente. Diese können natürlich trotzdem verwendet werden, um die Variante in den Diagrammen des Modells kenntlich zu machen. Eine Möglichkeit dazu ist z. B. in den Abbildungen 6.3 und 6.4 zu sehen. Die Aktivierung der Variante erfolgt dann mittels der modifizierenden Transformationsregel in Listing 6.1.

Die Regel basiert auf einer Namenskonvention für optionale Interaktionen und deren Platzhalter. Eine Interaktion `I` wird dabei durch einen Platzhalter `replaces_I` ersetzt. Zudem wird dem Platzhalter ebenfalls die Rolle des `Insurer` als `Performer` zugeteilt. Im Suchteil der Regel werden nun Aktivitäten mit dem `Performer`-Ausdruck `Insurer` gesucht, wobei die eine das Schlüsselwort `replaces` sowie den Namen der anderen im eigenen Namen führt. Bei zwei dementsprechenden Aktivitäten handelt es sich mit Sicherheit um eine Interaktion mit dem Versicherer (`ai`) und deren Platzhalter (`ar`). In diesem Fall werden dann alle ein- (`tIn`) und ausgehenden Transitionen (`tOut`) des Platzhalters identifiziert und vermerkt. Im Ersetzungsteil werden zunächst alle

```
RULESET "integrate insurance"
  RULE "interchange insurer routes with interactions"
    MATCH_ALL
      Activity ar:
        ar.performer contains "Insurer",
        ar.name contains "replace";
      Activity ai:
        ai.performer contains "Insurer",
        ar.name contains ai.name;
      Transition{+} ti IN tIn:
        ti.To == ar;
      Transition{+} to IN tOut:
        to.From == ar;
    REPLACE_WITH
      ar;
      ai;
      FOR_EACH ti IN tIn:
        ti.To = ai;
      FOR_EACH to IN tOut:
        to.From = ai;
```

Listing 6.1: Transformationsregel zur Erweiterung des FreightMixer-VDLP

gefundenen Elemente erwähnt, um sie im Prozess zu belassen. Der Platzhalter könnte optional gelöscht werden, wirkt sich aber als passives Element nicht negativ aus. Der entscheidende Schritt besteht in der Modifikation der Transitionen. Hierbei wird die Interaktion als Ziel (`to`) bzw. als Start (`from`) aller ein- bzw. ausgehenden Transitionen bestimmt. Diese Regel wird für alle passenden Trefferkandidaten angewendet (`MATCH_ALL`). Da zudem kein Prozessselektor spezifiziert ist, erfolgt die Ersetzung für alle Prozesse im Schema.

Die Resultate der Transformation in den beiden betrachteten ≪`eService Capability Interaction Flow`≫-Prozessen des FreightMixer E-Service-Schemas sind in Abbildung 6.7 zu sehen. In beiden Teilprozessen werden alle darin enthaltenen Interaktionen mit dem Versicherer in den Kontrollfluss eingebunden. Damit entspricht das E-Service-Schema den Vorgaben des Geschäftsprozesses. Abb. 5.26 zeigt die Änderungen zwischen den Revisionen des Handover-Teilprozesses im Analysebereich des E-Service-Schema-Managers.

Ein zweiter Fall der Änderung im FreightMixer-Szenario soll das Reengineering von E-Service-Schemata verdeutlichen. Hierfür soll angenommen werden, dass bei der Analyse der Ausführung von FreightMixer-Transportdienstleistungen eine negative Auswirkung der Tracking-Funktionalität auf die Kostenstruktur ermittelt wurde. Daher soll der Versand von Statusberichten auf den Teilstrecken eingestellt werden. Dies gilt nicht nur für das Basisschema, sondern auch dessen Varianten. Problemberichte im Fehlerfall sollen jedoch weiterhin versendet werden. Abbildung 6.8 zeigt das Ziel für die Variante einer Transportdienstleistung mit Versicherung.

Abbildung 6.7.: Ausschnitte der erweiterten Handover (oben) und Handover-Problem (unten) Interaktionsprozesse

Das Reengineering des Schemas erfordert die Konzeption eines Transformationskonzepts ohne Einfluss auf das E-Service-Schema. Im Beispiel basiert es darauf, dass alle Interaktionen, die dem Versand von Statusberichten dienen, das Stichwort `notify` im Namen führen. Interaktionen zum Versand von Problemberichten sind mit dem Schlüsselwort `inform` ausgezeichnet. Diese können durch die Regel in Listing 6.2 geändert werden.

```
RULESET "remove notifications"
  RULE "change notifications to routes"
    MATCH_ONCE
        Activity{*} a IN notificationActivities:
        (a.name contains "notify") and (not (a.name contains "replace")),
        (a.performer contains "Customer") or (a.performer contains "Insurer");
    REPLACE_WITH
      FOR_EACH a IN notificationActivities:
        name = "activity without function",
        route = Route();
```

Listing 6.2: Transformationsregel zur Reduktion des FreightMixer-VDLP

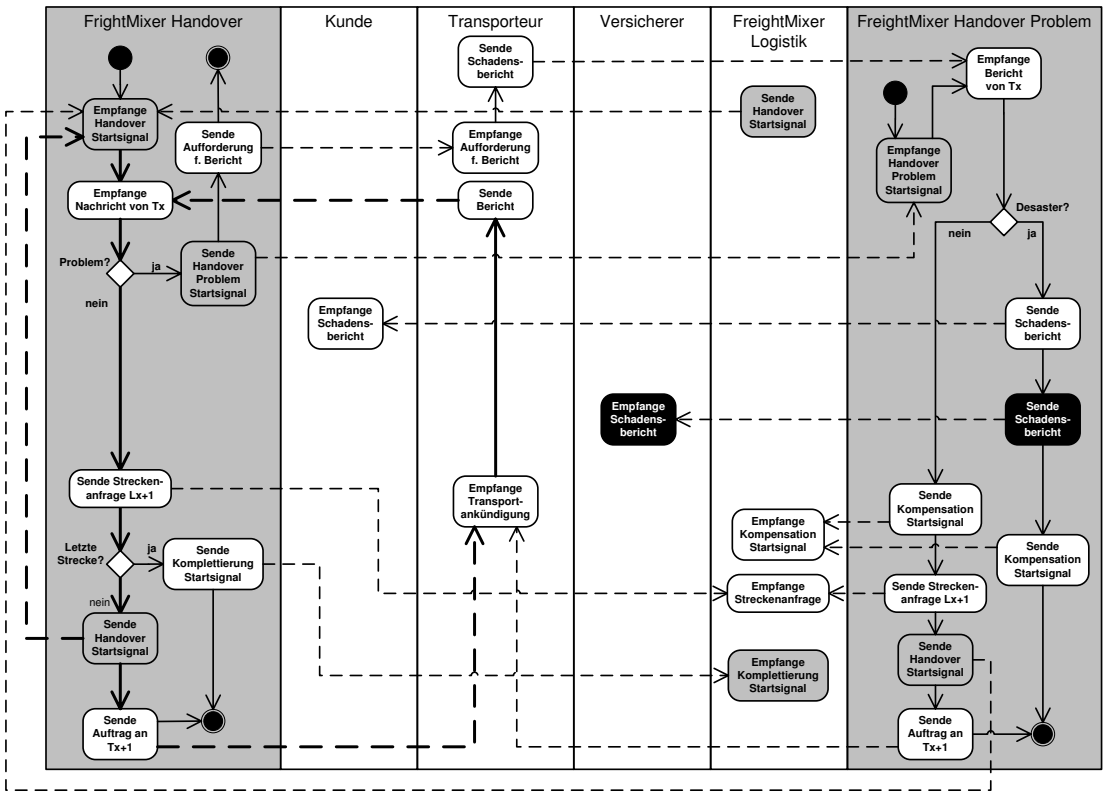

Abbildung 6.8.: Reduzierte Transportkoordination im FreightMixer-VDLP

Im Suchteil der Regel wird nach Aktivitäten gefragt, die das Schlüsselwort im Namen führen. Platzhalter sollen jedoch nicht selektiert werden. Hierzu erfolgt ein Ausschluss von deren obligatorischem Namensbestandteil `replace`. Zudem soll die Benachrichtigung nur für den Kunden und den Versicherer, nicht jedoch für den Frachtführer entfallen. Daher werden entsprechende Bedingungen für den `Performer`-Ausdruck in das Suchprädikat aufgenommen. Bei den derart beschriebenen Aktivitäten handelt es sich im Beispiel genau um die Menge aller Interaktionen, in denen ein Statusbericht an Kunde oder Versicherer versandt wird. Diese Aktivitäten werden im Ersetzungsteil aber nicht gelöscht. Dadurch würde nämlich die Prozessstruktur in Bezug auf den Kontrollfluss verändert werden. Stattdessen wird die Art der Aktivität in eine `Route` ohne Funktion geändert. Hierbei wird diese `Route` als neues Element des Prozesses erzeugt. So entfällt die Interaktion, die als `Tool`-Element beschrieben war. Die Struktur des Kontrollflusses bleibt jedoch unverändert erhalten. Die derart beschriebene Regel wird in jedem Prozess des Schemas für genau einen Trefferkandidaten angewendet, der aber in diesem Fall eindeutig ist und alle gesuchten Aktivitäten enthält. Die Resultate der nach dieser Regel durchgeführten Transformation werden in Abbildung 6.9 wiederum für die beiden betrachteten ≪`eService Capability Interaction Flow`≫-Prozesse des FreightMixer E-Service-Schemas gezeigt.

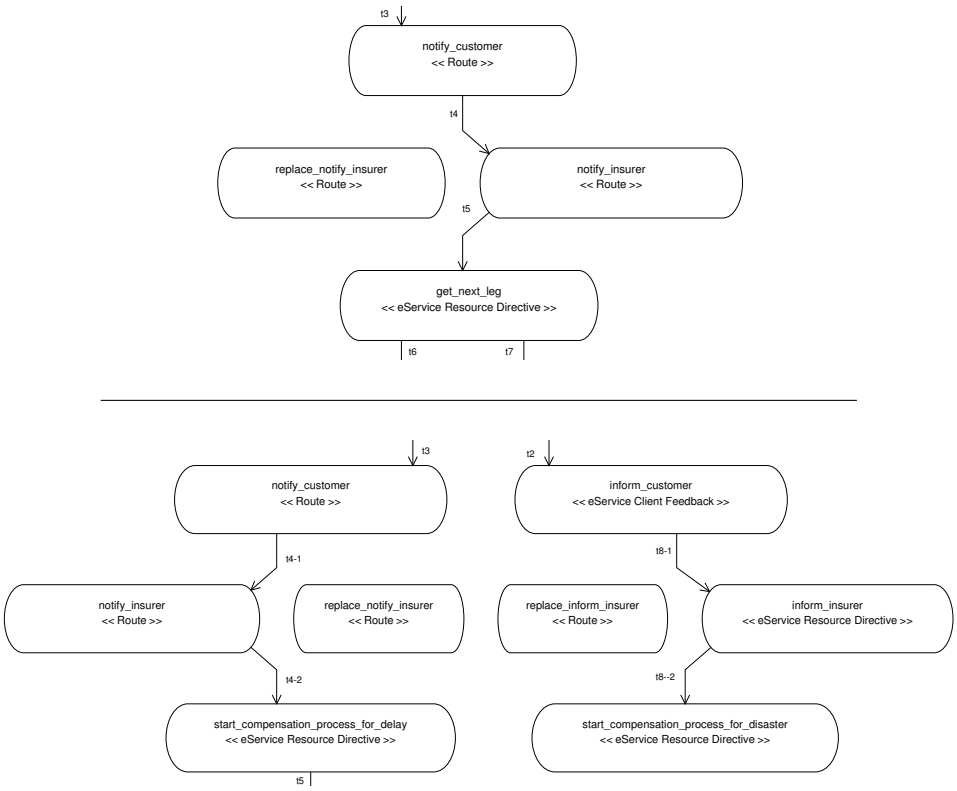

Abbildung 6.9.: Ausschnitte der reduzierten Handover- (oben) und Handover-Problem-Interaktionsprozesse (unten)

Unabhängig von der Art der Anwendung führt die Transformation eines Schemas grundsätzlich zu einer neuen Revision der E-Service-Schema-Spezifikation. Bei der Schema-Variation kann davon ausgegangen werden, dass die vorab geplante und getestete Transformation zu dem gewünschten Ergebnis führt. In diesem Fall wird die neue Revision unmittelbar in das Schema-Repository geladen und zur Implementierung und Ausführung einer E-Service-Instanz durch die Engine- und Aggregation-Manager-Plattformkomponenten verwendet. Beim Schema-Reengineering ist die Transformation hingegen ein interaktives Werkzeug. Hierbei werden situationsbedingt individuelle Regeln formuliert und sukzessive angewendet. Deren Wirkung ist nicht immer absehbar und muss mit den Hilfsmittel der Schema-Manager-Plattformkomponente validiert werden. Hierzu gehören u. a. Änderungsprotokolle und ein Mechanismus zum Revisionsvergleich. Im Falle unerwünschter Ergebnisse kann jede der sukzessiven Änderungen zurückgenommen und verbessert werden. Wenn ein zufriedenstellendes Ergebnis erreicht ist, wird dieses im Repository persistent gemacht und kann von da an zur Implementierung und Ausführung von E-Service-Instanzen verwendet werden.[17]

[17]Optimaler weise sollten die Änderungen des Reengineerings auch in das E-Service-Modell

6.4.3. Implementierung und Ausführung

Nach Anwendung der Transformationsregel zur Integration der Versicherungsteilleistung kann das E-Service-Schema verwendet werden, um die Transportdienstleistung für Shoe&Shoe zu erbringen. Dies heißt grundsätzlich, dass eine spezifische E-Service-Instanz zu implementieren und auszuführen ist. Hierzu kommt die FRESCO-TK E-Service-Management-System-Plattform zum Einsatz. Im Zusammenspiel von deren Plattformkomponenten erfolgt die Implementierung und Ausführung der E-Service-Instanz als integrierter Prozess zur individuellen Realisierung der Dienstleistung. Im Folgenden wird zunächst die Implementierung für sich betrachtet. Danach wird der integrierte Prozess zur Realisierung der Shoe&Shoe Dienstleistung erläutert.

Die Implementierung der E-Service-Instanz basiert im Wesentlichen auf der Realisierung von ESMS-Anwendungskomponenten. Die Realisierung betrifft dabei Konstruktion und Deployment. Die Anwendungskomponenten umfassen zum einen E-Service-Ressourcen für die Kontaktpunkte im PIM bzw. Web Service-Komponenten im PSM. Zum anderen gehören dazu E-Service Engines für die Capabilities im PIM bzw. Web Service-Kompositionen im PSM. Die Realisierung dieser Komponenten obliegt jeweils demjenigen Teilnehmer, dem die zugehörige Rolle des E-Service-Schemas zugeordnet wird. Die Durchführung dieser Zuordnung ist Teil des Aggregationsvorgangs der E-Service-Instanz, der weiter unten beschrieben wird. Das Ergebnis soll hier jedoch schon vorweggenommen werden. Im betrachteten Geschäftsfall übernimmt Shoe&Shoe die Rollen des `Customer` und des `WarehouseStart`. FreightMixer tritt hier als `TransportPreparationManager`, `TransportCoordinationManager`, `TransportCompletionManager` und `LogisticManager` auf. Es verbleiben noch ein Versicherer für die `Insurer`-Rolle, die Messegesellschaft als `WarehouseEnd` sowie eine noch nicht festgelegte Anzahl von Frachtführern als `TransportProvider`. Die Rollenzuteilung gibt die Verteilung der Komponenten im Rahmen eines organisationsübergreifenden Anwendungssystems zur Produktionssteuerung vor. In Abbildung 6.10 wird die Verteilung aller Komponenten der Shoe&Shoe E-Service-Instanz als UML Deployment-Diagramm gezeigt.

Das Diagramm zeigt die konkreten Teilnehmer der Dienstleistung als Knoten mit ihren zugeteilten Rollen und den damit verbundenen Anwendungskomponenten. Jede Anwendungskomponente stellt eine Grid Service-Schnittstelle zur Verfügung. Aus den ≪`eService Capability Interaction Flow`≫-Spezifikationen im E-Service-Schema geht zudem hervor, auf welche Schnittstellen die einzelnen Komponenten im Verlauf des VDLP zugreifen. Dabei bezieht jede bilaterale Interaktion immer mindestens eine E-Service Engine ein. Aus der Abbildung gehen deutlich die Konsequenzen daraus hervor, dass im betrachteten Szenario alle Capabilities autonom von FreightMixer realisiert werden. In diesem Fall sind alle E-Service Engines auf einen Knoten des

übernommen werden. Zum aktuellen Zeitpunkt unterstützt das FRESCO-TK-Entwurfswerkzeug jedoch noch keinen Import von textuellen Modellrepräsentationen, d. h. dass Änderungen am UML-Modell zurzeit manuell nachvollzogen werden müssen. Alternativ können Reengineering-Regeln als obligatorische Schema-Variation genutzt werden (Patching).

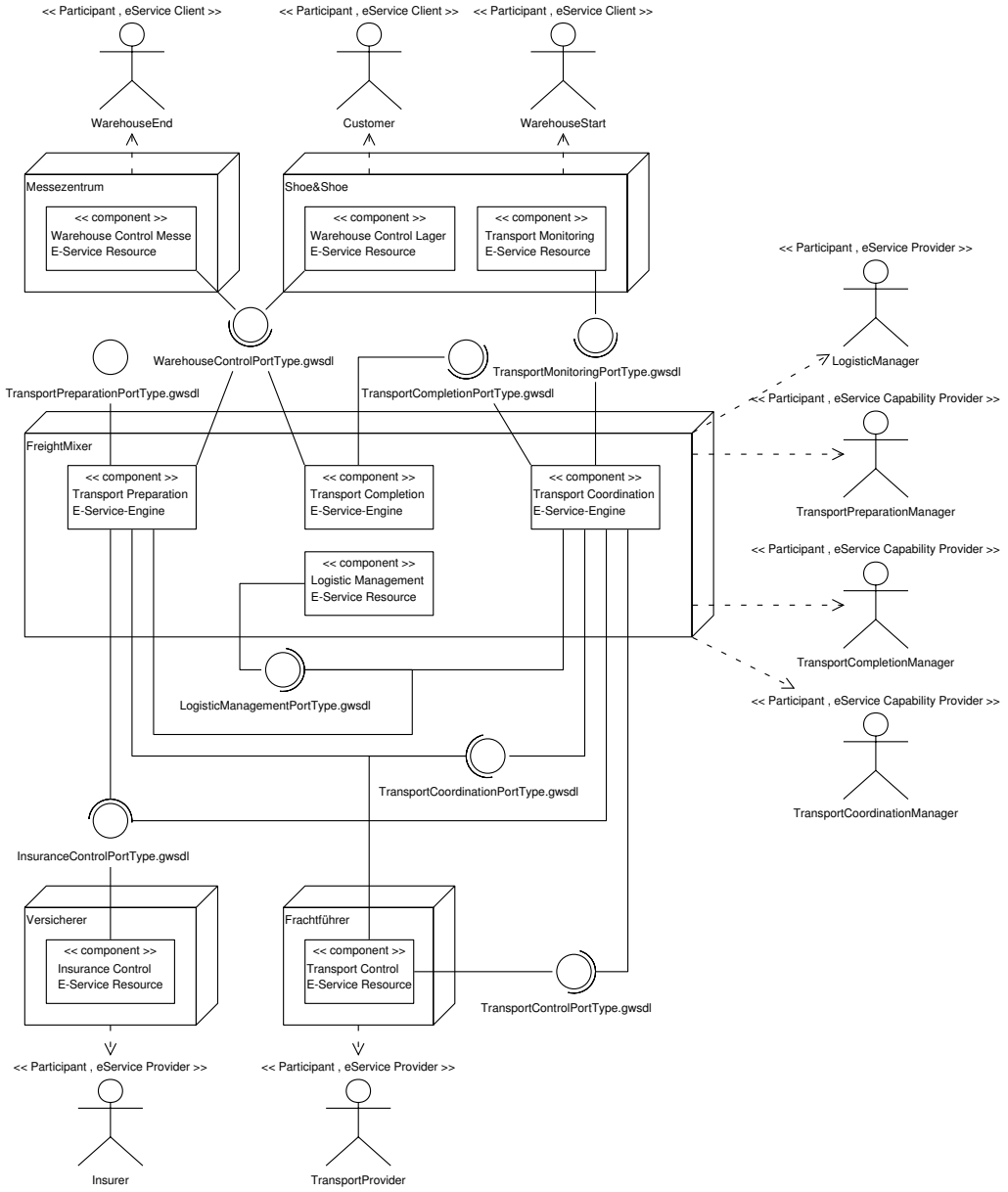

Abbildung 6.10.: Deployment von ESMS-Anwendungskomponenten für die E-Service-Instanz von Shoe&Shoe

Systems konzentriert. Zusammen mit der erwähnten Interaktionscharakteristik ergibt sich eine ausgeprägte Zentralisierung des verteilten Anwendungssystems. Technisch erlaubt das FRESCO-TK jedoch genauso gut eine Verteilung von E-Service Engines auf beliebige Knoten.

Um zu der gezeigten Systemarchitektur zu gelangen, müssen alle Teilnehmer die ihnen zugeteilten Anwendungskomponenten realisieren. Grundsätzlich können die zwei generellen Fälle von E-Service-Ressourcen- und -Engine-Realisierung unterschieden werden. E-Service-Ressourcen bilden Kontaktpunkte zu den operationalen Anwendungssystemen von Produktionseinheiten im virt. Dienstleistungsproduktionsnetzwerk. Dies können sowohl DLU im strategischen Dienstleistungsnetzwerk als auch Kunden sein. E-Service-Ressourcen sind somit in der Regel Adapter zur Integration bestehender Software-Systemlandschaften. Hierfür sind aber auch eigenständige ggf. interaktive Anwendungssysteme denkbar. Obwohl das FRESCO-TK die Konstruktion durch ein Software-Rahmenwerk unterstützt, würde diese doch in jedem Fall den zeitlichen Rahmen des Vorgangsmodells sprengen. Es muss daher vorausgesetzt werden, dass diese Komponenten bereits vorliegen. Für die beteiligten Netzwerkunternehmen ist dies unmittelbar plausibel, da dies einen Teil der koordinativen Regelungen im strategischen Netzwerk ausmacht. Wie schon weiter oben erwähnt wurde, gilt dies für Kunden nicht ohne Weiteres.[18] In Bezug auf die dort geführte Diskussion soll hier angenommen werden, dass die Schnittstellen einem fiktiven Standard entsprechen und daher auch beim Kunden schon realisiert sind.

Durch die Flexibilität des E-Service-Schemas nicht nur in Bezug auf geplante Varianten sondern auch in Bezug auf potenzielle ad hoc-Änderungen per Schema-Reengineering, kann die Existenz von E-Service Engine-Anwendungskomponenten unter keinen Umständen angenommen werden. Diese müssen für jede E-Service-Instanz individuell realisiert werden. Die Agilität des Vorgehens wird dabei im FRESCO-TK durch die Plattformkomponente des Engine-Managers gewährleistet. Dieser führt die Konstruktion und das Deployment von E-Service Engines automatisch und unter praktischen Gesichtspunkten ohne relevanten Zeitaufwand durch. Das Verfahren beruht auf der Generierung ausführbarer BPEL-Schemata aus den ≪`eService Capability Interaction Flow`≫-Spezifikationen des E-Service-Schemas. Deren Ausführung in einer BPEL-Laufzeitumgebung bildet den aktiven Kern einer E-Service Engine. Drumherum werden Adapter und Proxies generiert, die die Einbettung der BPEL-basierten Web Service-Kompositionen in die ESMS-Plattform bewirken. Für die bis hierher im Detail betrachtete `TransportCoordination` Capability, ist die automatisch konstruierte E-Service Engine in Abb. 6.11 zu sehen. Den Kern bilden zwei Komponenten, die durch BPEL-Übersetzung des `HandoverFlow` und `HandoverProblemFlow` implementiert werden.[19] Auf Basis der ebenfalls in der E-Service-Schema-Spezifikation enthaltenen Schnittstellenbeschreibungen wird ein Adapter für die `TransportCoordinationPortType`

[18] Siehe Sektion 6.3.2.1.

[19] Die BPEL-Übersetzung des `HandoverFlow` ist in Anhang A.2.8 beigefügt.

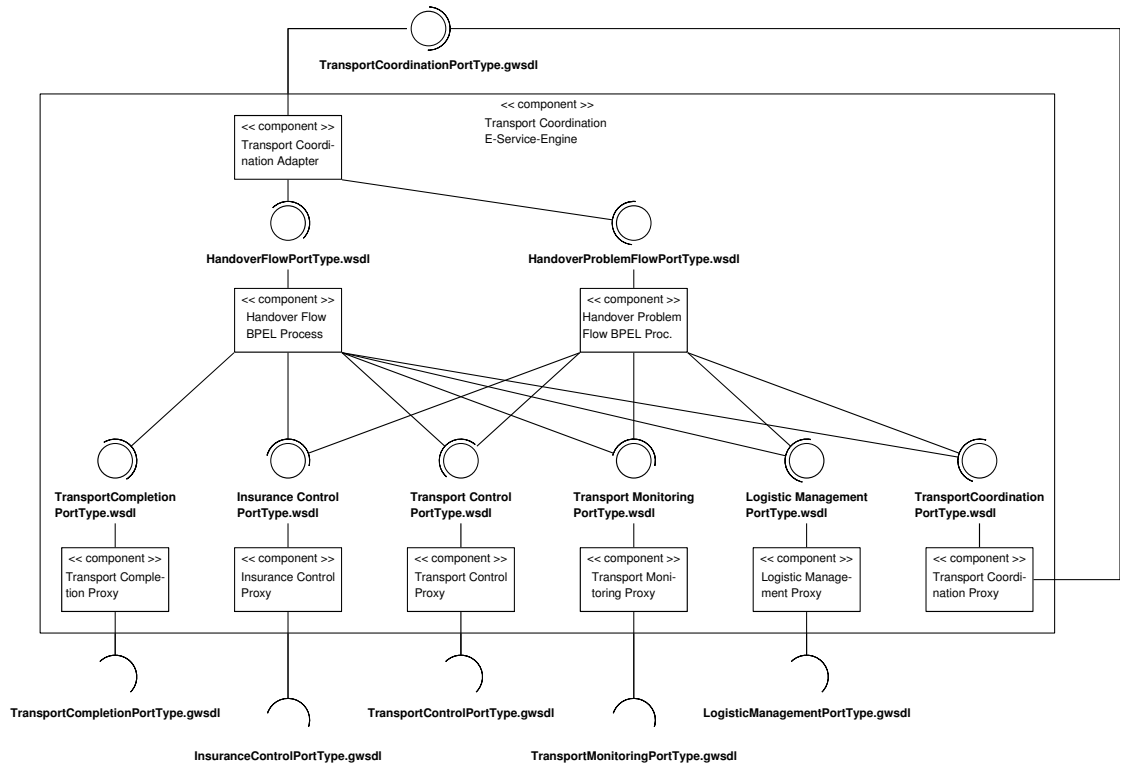

Abbildung 6.11.: Komponentendiagramm der Transport Control E-Service Engine

Grid Service-Schnittstelle generiert. Daneben wird für jede aus den Prozessen aufgerufene Grid Service-Schnittstelle ein Proxy generiert, der sich dynamisch an eine entsprechende Komponente bindet und eine Web Service-Schnittstelle bereitstellt. Im betrachteten Fall sind sechs verschiedene Proxies nötig. Schließlich erfolgt das Deployment der E-Service Engine für Adapter, BPEL-Schemata und Proxies in den jeweiligen Laufzeitumgebungen auf dem FreightMixer-Knoten.

Die beschriebene Implementierung der E-Service-Instanz bildet im FRESCO-TK einen Teilaspekt von deren Aggregation. Der Aggregationsvorgang beginnt mit der Implementierung des VDPN auf Stufe 3) des FRESCO E-Service-Management-Vorgehensmodells und begleitet alle Aspekte der Implementierung und Ausführung bis zum Ende von Stufe 4). Parallel muss FreightMixer die passenden Provider aus dem strategischen Dienstleistungsnetzwerk identifizieren. Das Szenario legt nahe, dass FreightMixer hierzu Mechanismen der E-Markets verwendet. Dies soll im Rahmen der vorliegenden Arbeit jedoch nicht thematisiert werden. Es wird lediglich vorausgesetzt, dass FreightMixer als Leader eines geschlossenen Unternehmensnetzwerks agiert und über die nötigen Mittel zur Identifikation der am besten geeigneten Logistikunternehmen verfügt. Die Aggregation umfasst dann die Registrierung der Teilnehmer, die Zuteilung der Teilnehmer zu einer Rolle im E-Service-Schema, die

Steuerung und/oder Kontrolle der Realisierung von Anwendungskomponenten sowie die Steuerung und/oder Kontrolle der Bereitstellung von Komponenteninstanzen. Bei der Durchführung dieser Aufgaben wird FreightMixer von der Aggregation-Manager-Plattformkomponente des ESMS unterstützt.

Ein wesentlicher Aspekt der Aggregation ist die Wahl einer Strategie zur Art und Abfolge der Durchführung einzelner Teilaufgaben. Diese Aggregationsstrategie muss zur produzierten Dienstleistung passen. Je nach Art der beteiligten Assets und Demands sind vor allem zwei charakteristische Fälle zu unterscheiden. Zum einen benötigen manche Aktivitäten im DLP eine bestimmte Vorlaufzeit, die mindestens einzuhalten ist. Zum anderen sind die mit einer Aktivität verbundenen Ressourcen zum Teil begrenzt und wertvoll, sollen nur für eine möglichst kurzer Dauer belegt bleiben und nicht ungenutzt blockiert werden. Andere Aspekte mit Einfluss auf die Aggregationsstrategie stehen u. a. im Zusammenhang zur Charakteristik des DLP. Hier soll oder kann die Auswahl von VDPN-Teilnehmern unter Umständen nicht sofort erfolgen. Dies kann dadurch bedingt sein, dass zu Beginn des DLP noch nicht alle zur Auswahl notwendigen Informationen vorliegen. Dahinter kann aber auch der Wunsch stehen, so lange wie möglich flexibel auf die Bewegungen des Marktes reagieren zu können.

Überlegungen zur Aggregationsstrategie sind auch im FreightMixer-Szenario notwendig. Hier sind Aktivitäten im Zusammenhang mit dem Versenden von Reports zum Tracking oder zur Fehlerbehandlung, der Administration der Versicherung, des logistischen Managements sowie der Produktionssteuerung potenziell wenig von der Art der Aggregation beeinflusst. Im Fall der physikalischen Verrichtungen des Transports ist dies jedoch möglicherweise anders. Die Lagerverwaltung benötigt ggf. einen Vorlauf zur Ausgabe oder Annahme von Waren. Dies sollte jedoch in den Entwurf des VDLP eingehen. Eine Berücksichtigung im Zuge der Aggregation ist zwar grundsätzlich möglich, dies sollte jedoch vermieden werden, da sonst planerische Aspekte der Dienstleistung logisch und physikalisch separiert werden. Aus Sicht der Aggregation stellen die Transporte der einzelnen Teilstrecken den interessantesten Aspekt dar. Genau wie die Lagerei braucht das Transportwesen potenziell einen gewissen Vorlauf, um physikalische Ressourcen in Form von Transportmitteln zu aktivieren. Dies sollte wiederum im VDLP berücksichtigt werden. Durch die flexible Zusammensetzung der Transportstrecken ergeben sich jedoch Optionen für die Auswahl und Zuweisung von Frachtführern zur korrelierten `TransportProvider`-Rolle, die nicht in die Zuständigkeit der Produktionssteuerung fallen. Eine mögliche Vorgehensweise ist die Durchführung der kompletten Routenplanung sowie die Auswahl und Zuweisung aller Frachtführer vor Beginn des VDLP. Eine andere Möglichkeit ist die Planung von Teilstrecken im Verlauf des Transports. In diesem Fall kann die Auswahl und Zuweisung von Frachtführern erst nach Planung der jeweiligen Teilstrecke erfolgen. Durch Wahl einer entsprechenden Aggregationsstrategie können beide Fälle realisiert werden. Im Folgenden wird der realistischere Fall angenommen, dass die Routenplanung im Wesentlichen zu Beginn erfolgt. In diesem Fall könnten dann z. B. drei Teilstrecken

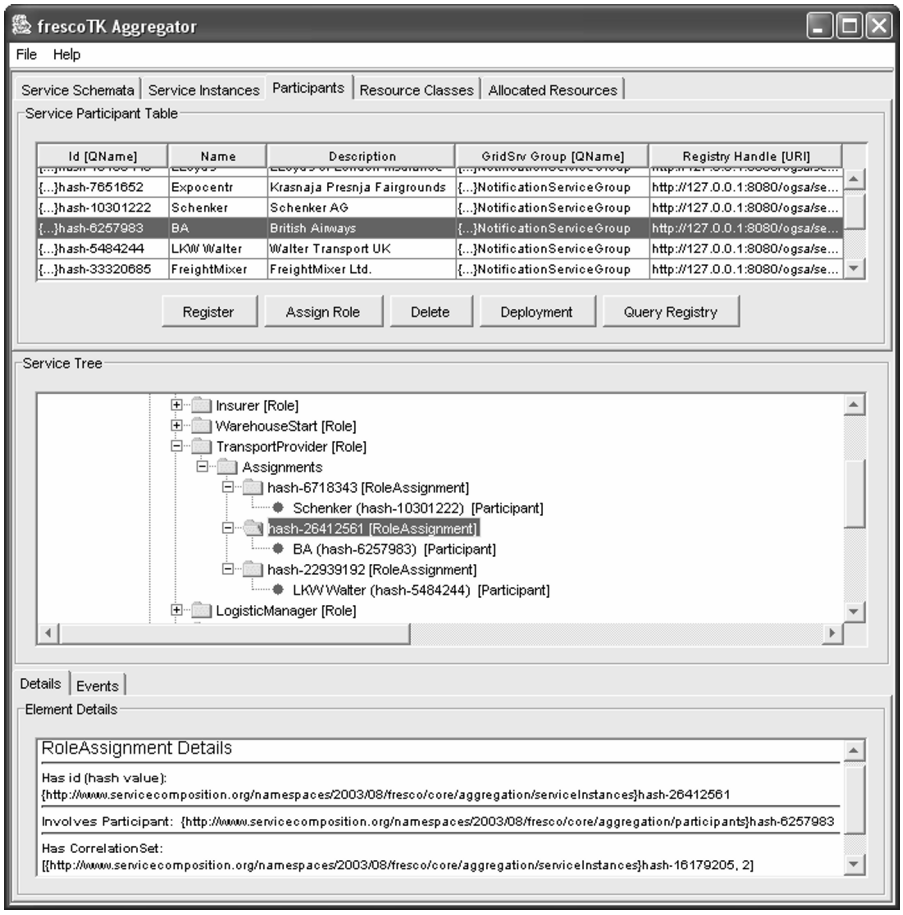

Abbildung 6.12.: Rollenzuweisungen für die Shoe&Shoe E-Service-Instanz im FRESCO-TK Aggregation-Manager

1) per LKW von Shoe&Shoe zum Flughafen in Bristol, 2) per Flugzeug von Bristol nach Moskau und 3) wieder per LKW vom Flughafen Moskau zum Messegelände vorgesehen sein. Demnach wird die `TransportProvider`-Rolle mit Korrelation zur E-Service-Instanz und Teilstrecke drei Frachtführern zugewiesen. Abbildung 6.12 zeigt, wie diese Zuweisung mithilfe der Aggregation-Manager-Plattformkomponente erfolgt. Im Detailbereich ist die Korrelation der `TransportProvider`-Rolle einer Fluggesellschaft mit der zweiten Teilstrecke zu erkennen.

Die Realisierung der Aggregationsstrategie wird in den meisten Fällen manuelle Anteile beinhalten. Von Fall zu Fall können Teilaufgaben auch automatisiert durchgeführt werden. Der Geschäftsfall im FreightMixer-Szenario lässt sich durch die standardmäßige Strategie des Aggregation-Managers unterstützen. Vor dem Start der Strategie müssen die Teilnehmer registriert und die Rollen zugewiesen werden. Es ist auch vorteilhaft, die automatisierte Realisierung der E-Service Engines schon zu Beginn durchzuführen. Wenn dann die Ausführung der Strategie gestartet wird,

fehlt nur noch die Allokation der Komponenteninstanzen. Dies führt der Aggregation-Manager bei Anfragen von Anwendungskomponenten nach Bindungsinformationen automatisch mittels OGSI-Mechanismen durch. FreightMixer kann nun die Dienstleistungsproduktion mittels Aktivierung der `TransportPreparation` Capability starten. Dies erfolgt durch Aufruf der `startTransport()`-Operation des `TransportPreparationPortType` unter Angabe der E-Service-Instanz-Kennung. Bei der Bindung an die E-Service Engine wird eine Instanz von dieser erzeugt und die darin befindliche BPEL-Laufzeitumgebung übernimmt die aktive Steuerung. Während des anschließenden Ablaufs des VDLP sammelt FreightMixer die Laufzeitinformationen der verschiedenen BPEL Engines als Protokoll über den VDLP-Verlauf. Dieser kann später für Audits oder Analysen verwendet werden.

6.5. Bewertung der FRESCO-basierten Szenarioumsetzung

Im Zuge der Bewertung soll abschließend geklärt werden, ob und inwieweit die Vision des FreightMixer-Szenarios durch das E-Service-Konzept und dessen Realisierung in FRESCO verwirklicht wurde. Zur Beantwortung dieser Frage werden zunächst die Anforderungen des Szenarios und die Eigenschaften der Lösung noch einmal zusammengefasst. Dann erfolgt ein Vergleich, der hervorheben soll, welche Aspekte gut und welche weniger gut gelöst werden. Am Ende wird hieraus eine faktische Aussage über den Grad der Anwendbarkeit des Ansatzes abgeleitet.

Das FreightMixer-Szenario gibt Voraussetzungen für einen Markt logistischer Dienstleistungen vor. Unter diesen Voraussetzungen beschreibt es die Eigenschaften einer speziellen Form der virtuellen Dienstleistungsproduktion auf Basis von E-Services. Konkrete Aussagen lassen sich für die Eigenschaften des exemplarischen Geschäftsfalls in Bezug auf die hier beschriebene Dienstleistung und deren Produktion wie folgt feststellen:

1. FreightMixer plant autonom verschiedene logistische Leistungsbündel und -prozesse auf Basis eines virt. Hub-and-Spoke-Systems grob im Voraus, wobei klar beschränkte Flexibilität in Bezug auf die Route und in Bezug auf logistische Nebenleistungen besteht.

2. Eine Nachfrage von Leistungen führt zur Feinplanung eines individuellen Leistungsbündels in Bezug auf die Route, in Bezug auf die Nebenleistungen und in Bezug auf die Auswahl optimaler Teilnehmer im Dienstleistungsnetzwerk.

3. Die Teilnehmer des strategischen FreightMixer-Netzwerks bieten ihre Dienstleistungen als E-Services dar, die von FreightMixer zu einem E-Service des Leistungsbündels integriert werden, der eine Regelung der Koordination kooperativer Teilleistungen repräsentiert.

4. Mithilfe des E-Service steuert FreightMixer die Dienstleistungsproduktion im individuellen VDPN, was die Freigabe, Kontrolle und Sicherung der einzelnen Teilleistungen beinhaltet.

Der wesentliche Aspekt dieses Ablaufs besteht darin, dass die E-Service-Technologie FreightMixer die Regelung der Koordination kooperativer Vorgänge der Dienstleistungsproduktion autonomer Netzwerkunternehmen derart erlaubt, dass eine Individualisierung der Produktionsplanung und -steuerung in Bezug auf spezifische Anfragen unmittelbar im Zuge einzelner VDU-Lebenszyklen erfolgen kann.

In FRESCO wurde ein konzeptionelles und technisches Rahmenwerk für die allg. virtuelle Dienstleistungsproduktion auf Basis einer Virtualisierung von Dienstleistungen mittels E-Services entwickelt. In den vorangegangenen Sektionen wurde gezeigt, dass das konzeptionelle Rahmenwerk unter den Voraussetzungen des Szenarios eine Organisationsstruktur für VDPN, ein Prozessmodell für Dienstleistungen sowie ein Vorgehensmodell zur virtuellen Dienstleistungsproduktion liefert, um Logistikdienstleistungen der grundsätzlichen Art und auf die grundsätzliche Weise von FreightMixer zu produzieren. Es wurde ebenfalls gezeigt, wie diese virtuelle Dienstleistungsproduktion für den exemplarischen Geschäftsfall auf Basis des FRESCO ESMS konkret abläuft. Für diesen Ablauf kann eine Reihe von Aussagen getroffen werden:

1. FreightMixer plant logistische Leistungsbündel als E-Service-Modelle ohne Einbezug von Providern im Voraus, wobei die Flexibilität der Route auf dynamischer Rollenzuweisung basiert und eine beschränkte Anzahl von Varianten durch Schema-Variation antizipiert wird.

2. In Bezug auf die Anforderungen einer individuellen Anfrage führt FreightMixer die Transformationen der Schema-Variation durch und erstellt dadurch ein optimiertes E-Service-Schema mit den gewünschten Nebenleistungen, wobei die Route zur Laufzeit einfließt.

3. Die Teilnehmer des strategischen FreightMixer-Netzwerks bieten Kontaktpunkte zum Zugang ihre Dienstleistungen in Form von Assets und Demands an, die in Form von Schnittstellen für E-Service-Ressourcen spezifiziert sind und FreightMixer als Grundlage für E-Service-Modelle dienen.

4. Die Produktionssteuerung erfolgt mittels einer E-Service-Instanz, in die VDPN-Teilnehmer E-Service-Ressourcen einbringen und für die FreightMixer E-Service Engines generiert, die den im E-Service-Modell geplanten VDLP ausführen.

FSM und FRESCO-Entwurfswerkzeug erlauben die Erstellung von E-Service-Modellen, die die Koordination kooperativer Leistungsprozesse autonomer VDPN-Teilnehmer als VDLP regeln, sich per regelbasierter Transformation individuell anpassen lassen und im Zuge agiler Entwicklung auf der FRESCO ESMS-Plattform als organisationsübergreifendes Anwendungssystem zur Produktionssteuerung implementiert werden.

Die direkte Gegenüberstellung der einzelnen Punkte erlaubt eine Aussage über das Verhältnis von den im Szenario vorgegebenen zu den bei der Umsetzung erreichten Eigenschaften. Der erste Punkt betrifft die Planung logistischer Leistungsbündel. Dies erfolgt in der Umsetzung durch E-Service-Modelle. Eine begrenzte Anzahl von Varianten kann hier durch Modellelemente und Transformationsregeln zur Schema-Variation berücksichtigt werden. Die Flexibilität der Transportroute wird durch die Modellierung korrelierter Rollen erreicht. Es ist jedoch einzuschränken, dass die Planung von E-Service-Modellen nur einen Teilaspekt der Planung von Leistungsbündeln in Hinblick auf die Produktionssteuerung darstellt. Weitere Aspekte, z. B. in Hinblick auf die Vermarktung, müssen ergänzt werden.

Der zweite Punkt bezieht sich auf die Anpassung von Leistungsbündeln. Das Szenario verlangt hier eine individuelle Optimierung von Leistungsbündeln in Bezug auf die Anforderungen des Kunden. Hierbei ist eine Route zu wählen, Nebenleistungen einzubinden und geeignete Provider zuzuweisen. Die FRESCO-Lösung fokussiert hier vor allem die Einbindung von Nebenleistungen, da diese in Hinblick auf die Produktionssteuerung die größten Konsequenzen hat. Die Umsetzung erfolgt mittels Durchführung von Transformationen. Hierbei führt die Anwendung einer Transformationsregelmenge stets zu einem E-Service-Modell, dessen VDLP in Bezug auf eine spezifische Leistungsoption optimiert ist. Die Kombination solcher Transformationen in Hinblick auf die Anforderungen führt zu einem individuellen E-Service-Modell. Über die Anforderungen des Szenarios hinaus unterstützt das FRESCO ESMS auch ein Schema-Reengineering für nicht a priori geplante Änderungen. Bei der Individualisierung von Routen wird grundsätzlich auf die betriebswirtschaftlichen Planungsmethoden verwiesen. In Bezug auf deren Konsequenzen für die Produktionssteuerung erlaubt die FRESCO-basierte Lösung die Einbindung einer beliebigen Anzahl von Teilstrecken und Frachtführern. Der letzte Aspekt der Individualisierung ist die Auswahl optimierter Provider. Das FRESCO-Rahmenwerk erlaubt durch die zugrunde liegenden Service-orient. Konzepte und Techniken eine individuelle und ggf. dynamische Zuweisung von Rollen und Bindung von Komponenten. Darüber hinaus wird die Identifizierung von Teilnehmern nicht unterstützt. Dies wird im Szenario aber auch nicht von der E-Service-Technologie verlangt.

Der dritte Punkt legt die Rolle der E-Service-Technologie sowie die damit verbundenen Funktionen und Eigenschaften fest. Die E-Service-Technologie fungiert im Szenario primär als „Enabler“-Technik der virt. Organisation. Der E-Service-Begriff steht hier für eine dienstleistungsspezifische Integrationstechnik. Diese kombiniert Aspekte der Interaktion von Kunden und DLU mit deren Koordination im Sinne des DLP. Eine entsprechende Technik kann für alle Beteiligten vorausgesetzt werden und begründet die im Szenario geschilderten Eigenschaften der Dienstleistungserbringung. Diese Vorgabe des Szenarios bildet mehr Legitimation als Anforderung der FRESCO-basierten Umsetzung. Die FRESCO E-Service-Technik begründet die Eigenschaften der FRESCO-basierten Lösung und darf auf Grund der Ausführungen des Szenarios als vorhanden vorausgesetzt werden. Diesbezüglich setzt FRESCO die Bereitstellung von

E-Service-Ressourcen-Komponenten bei allen Teilnehmern voraus. Diese dienen als Beschreibung und technische Realisierung von Endpunkten dienstleistungsspezifischer Kommunikation aller VDPN-Produktionseinheiten, d. h. sowohl Provider als auch Kunden. Auf dieser Basis plant, modifiziert und realisiert FreightMixer virt. Dienstleistungen im Wesentlichen durch Modellierung, Implementierung und Ausführung von VDLP durch E-Service Capabilities und entsprechende Engine-Komponenten. Diese Form von E-Services geht grundsätzlich konform mit den Vorstellungen des Szenarios. Sie bringt aber auch einige zusätzliche Eigenschaften mit sich. In ihrer allgemeinen Form, in der Capabilities sowohl vom Broker als auch von den Providern geplant werden, können komplexe Prozesse von Assets in den VDLP integriert werden, was jedoch mit dem Preis einer ebenso komplexen bilateralen Abstimmung zwischen Broker und Provider einhergeht. Für die Umsetzung des FreightMixer-Szenarios wurde wegen der hohen Anforderung an die Organisation der Planung darauf verzichtet. Die Folge ist dann jedoch, dass das Interaktionsprotokoll zwischen Provider und Broker von Letzterem diktiert wird. Provider müssen sich an die Vorgabe der Planung halten bzw. müssen u. a. in Bezug auf diese Eigenschaft ausgewählt werden. Als weitere Konsequenz der FRESCO E-Service-Technik soll erwähnt werden, dass es sich dabei nicht um ein mehrstufiges Modell handelt. Die Verwendung von E-Services als Assets eines übergeordneten E-Service soll hier zwar nicht ausgeschlossen werden, ist jedoch weder geplant noch untersucht worden. Im Falle des FreightMixer-Szenarios ist dies auch nicht gefordert.

Der vierte Punkt thematisiert die Automatisierung der Produktionssteuerung. Hierbei betont das Szenario die Funktion der E-Service-Technik als Steuermechanismus im VDPN. In diesem Zusammenhang werden klassische Funktionen der Produktionssteuerung auf Netzwerkebene von Produktionsnetzwerken wie Freigabe, Kontrolle und Sicherung von Teilprozessen im Zuge des kooperativen Produktionsverlaufs angeführt. Die Realisierung berücksichtigt dies mittels der E-Service-Management-System-Plattform des technischen FRESCO-Rahmenwerks. Hierin erfolgt die Implementierung einer E-Service-Instanz als organisationsübergreifendes verteiltes Anwendungssystem zur Produktionssteuerung. Durch Methoden modellgetriebener Entwicklung ist der notwendige Zeitaufwand für das Szenario ohne praktische Relevanz. Konstruktion und Deployment der E-Service Engines im gezeigten Fall dauern Minuten, was in Anbetracht zeitlicher Toleranzen auf den Transportstrecken vernachlässigt werden kann. Die Steuerung folgt exakt dem VDLP im E-Service-Modell. Das Beispiel zeigt, wie hierdurch Transporte freigegeben, auf ihr Ergebnis hin kontrolliert und ggf. kompensiert werden können.

Als Ergebnis der Auswertung kann festgestellt werden, dass die Umsetzung der Produktionsplanung und Steuerung des exemplarischen Geschäftsfalls mit dem FRESCO-Rahmenwerk unter den beschriebenen Voraussetzungen und Einschränkungen im Rahmen der Genauigkeit des zugrunde liegenden Szenarios grundsätzlich möglich ist. In Bezug auf die Funktionen und Eigenschaften der Umsetzung in Hinblick

auf Produktionsmanagement und Steuerung werden alle Annahmen des Szenarios erfüllt. Die Umsetzung geht in den entscheidenden Punkten der Flexibilität in Bezug auf ad hoc-Anpassung und Koordination von Teilprozessen verschiedener autonomer VDPN-Teilnehmer über die Vorgaben hinaus. Es ist jedoch anzumerken, dass die Mittel des FRESCO-Rahmenwerks alleine nicht ausreichen, um alle Aspekte des Szenarios zu realisieren. Verschiedene oben diskutierte Aspekte des Netzwerk-Managements, des Dienstleistungsmarketings und der logistischen Planung gehen über die klar gezogenen inhaltlichen Grenzen der Produktionsplanung und -steuerung virt. Dienstleistungsproduktion hinaus. Gleichwohl ist die Rolle des FRESCO-Rahmenwerks als Teil des ganzheitlichen Systems einer Enterprise-Architektur schon antizipiert worden. Dessen Umsetzung gibt das Szenario allerdings nicht her.

6.6. Zusammenfassung

Die vorangegangenen Sektionen dienten zur Evaluation der Beiträge im Rahmen der vorliegenden Arbeit. Stellvertretend für die generellen Konzepte, Modelle, Methoden und Mechanismen zur Unterstützung der virtuellen Dienstleistungsproduktion durch Virtualisierung von Dienstleistungen wurde hierzu das konkrete FRESCO-Rahmenwerk betrachtet. Dessen konzeptionelle und technische Anteile wurden dazu in den Kontext des anfangs beschriebenen Szenarios gestellt und die Eigenschaften dieser Anwendung dann mit den Vorgaben verglichen.

Der erste Schritt des Vorhabens befasste sich mit der Anwendung des konzeptionellen Rahmenwerks auf die vorgegebene Situation im Szenario. Dies betraf zunächst die Organisationsstruktur. Hier wurden die Rollen des konzeptionellen VDPN-Modells im Szenario identifiziert und in Bezug auf die damit verbundenen Aufgaben als passend festgestellt. Als Nächstes konnte auch das FSMV als tragfähig verifiziert werden, um die im Szenario beschriebenen Abläufe zu unterstützen. Am Ende konnte auf Basis des FRESCO E-Service-Metamodells eine grundsätzliche Möglichkeit zur Virtualisierung der vorgegebenen logistischen Dienstleistungen aufgezeigt werden.

Nach Darstellung der konzeptionellen Einsatzfähigkeit wurde die Anwendung des FRESCO-Rahmenwerks konkretisiert. Hierzu erfolgte eine Anwendung des technischen Rahmenwerks auf die vorgegebene Situation im exemplarischen Geschäftsfall des Szenarios. Hierbei wurde zunächst die Modellierung und Spezifikation von E-Service-Modellen veranschaulicht und für den Aspekt der Frachtübergabe im Detail durchgeführt. Dann wurde die systematische Änderung von E-Service-Modellen anhand regelbasierter Transformation erläutert, in Bezug auf Schema-Variation und -Reengineering in praktischer Hinsicht systematisiert und jeweils am Beispiel des Geschäftsfalls konkret durchgeführt. Schließlich wurden die Implementierung und Ausführung einer E-Service-Instanz für den Geschäftsfall betrachtet. Dabei wurde dessen Gesamtarchitektur veranschaulicht, die generierten Komponenten analysiert und Aggregationsstrategien diskutiert.

Nach abgeschlossener Darstellung der konzeptionellen und konkreten Anwendung des FRESCO-Rahmenwerks befasste sich die letzte Sektion mit der Bewertung der dadurch erreichten Umsetzung im Vergleich zur ursprünglichen Vision des Szenarios. Hierzu wurden zum einen die essenziellen Anforderungen des Szenarios und zum anderen die wesentlichen Eigenschaften der Umsetzung identifiziert und zusammengefasst. Dann wurde gezeigt, dass die Eigenschaften die Anforderungen zumindest in Bezug auf den Aspekt der Produktionsplanung und -steuerung subsumieren.

Abschließend sollen hier die drei wesentlichen Ergebnisse der Evaluation festgehalten werden. Zum einen haben sich die FRESCO-Konzepte zumindest für die Voraussetzungen des Szenarios als passend erwiesen. Zum anderen bieten die FRESCO-Techniken eine eindeutige Unterstützung für den exemplarischen Geschäftsfall. Das FRESCO-Rahmenwerk kann somit insgesamt als Baustein für die Realisierung von VDU dienen.

7. Zusammenfassung und Ausblick

Die nachfolgenden Sektionen bilden den Abschluss der vorliegenden Arbeit. Hierzu gehört zunächst eine in Sektion 7.1 dargestellte Zusammenfassung der Ziele und Ergebnisse im Verlauf der Untersuchung. Auf Basis von deren expliziter Darstellung soll dann in Sektion 7.2 eine Bewertung abgeleitet werden. Beides zusammen bildet die Grundlage eines Ausblicks, der in Sektion 7.3 wünschenswerte Ergänzungen identifiziert und neue Perspektiven aufzeigt, die sich durch die Untersuchungen der vorliegenden Arbeit eröffnet haben.

7.1. Zusammenfassung

Motivationen und Ziele

Die vorliegende Arbeit dokumentiert eine Untersuchung von Unterstützungsmöglichkeiten der Planung und Steuerung virt. Dienstleistungsproduktion auf Basis Service-orient. Dienstleistungsvirtualisierung. Sie setzt auf dem Hintergrund jüngerer Entwicklungen auf, die in den Bereichen der Betriebswirtschaft und Organisation von Dienstleistungen sowie der Software-Technik verteilter betrieblicher Anwendungssysteme zu beobachten sind. Zum einen wurden für die Produktion von Dienstleistungen moderne Prinzipien virt. Organisation von Dienstleistungsunternehmen vorgeschlagen, die generell eine IT-Unterstützung u. a. von Planungs- und Steuerungsfunktionen des Produktionsmanagements voraussetzen. Zum anderen eröffnet die Entwicklung Service-orient. Software-Techniken neue Möglichkeiten zur Realisierung zwischenbetrieblicher Anwendungssysteme auf Basis Service-orient. Software-Architekturen. Die Zielsetzung der vorliegenden Arbeit ergab sich durch Zusammenführung von Bedarf und Fähigkeit: Es sollten Service-orient. Konzepte und Techniken entwickelt werden, mit denen die Abwicklung wesentlicher Aufgaben der Planung und Steuerung virt. Dienstleistungsproduktion unterstützt und überhaupt erst ermöglicht wird. Diese Unterstützung sollte durch Service-orient. Anwendungssysteme erreicht werden, die zur Integration der Produktionseinheiten von virt. Dienstleistungsproduktionsnetzwerken dienen. Konkret manifestierte sich das Ziel als Entwicklung eines Rahmenwerks mit konzeptionellen und technischen Anteilen.

Auf konzeptioneller Ebene sollte ein Fundament zur Anwendung Service-orient. Techniken für die Planung und Steuerung virt. Dienstleistungsproduktion erarbeitet werden, das darauf beruht, die Potenziale Service-orient. Architekturen zur Spezifikation und Automatisierung zwischenbetrieblicher Interaktionen durch Virtualisierung von Dienstleistungen in Hinblick auf den inhärenten Interaktionsaspekt des zentra-

len Dienstleistungsprozesses zu nutzen und darüber die Koordination kooperativer Vorgänge zwischen den Teilnehmern im virt. Dienstleistungsunternehmen zu regeln. Hierbei mussten konzeptionelle Anteile der Ökonomie und Software-Technik berücksichtigt und kombiniert werden. Aus ökonomischer Sicht war ein integriertes Konzept zu entwickeln, das die Virtualisierung von Dienstleistungen und das Management virt. Dienstleistungsproduktion auf Basis einer Theorie interaktionsbasierter Koordination virt. Dienstleistungsunternehmen kombiniert. Aus Software-technischer Sicht war zu überlegen, wie die zu virtualisierenden Aspekte von Dienstleistungen auf Service-orient. Architekturen abgebildet werden können. Schließlich waren Prinzipien zur Synthese zu entwickeln, um die Software-Entwicklung Service-orient. virt. Dienstleistungen mit der Planung und Steuerung virt. Dienstleistungsproduktion zu kombinieren und diese Methodik in den ganzheitlichen Kontext betrieblicher Disposition und Operation zu integrieren.

Auf technischer Ebene sollten praktische Mittel realisiert werden, um das konzeptionelle Rahmenwerk operational zu manifestieren. Als direkte Konsequenz mussten die Prinzipien für eine „Software-Entwicklung zum Produktionsmanagement“ mittels Konzeption konkreter Entwicklungsmethoden und -Mechanismen umgesetzt werden, die Planung und Steuerung virt. Dienstleistungsproduktion mit Entwurf und Implementierung Service-orient. Anwendungssysteme verbinden sowie die notwendige Flexibilität und Agilität besitzen. Hierzu sollte eine Strategie modellgetriebener Entwicklung verfolgt werden, bei der sich der Übergang von ökonomischen Produktionsmodellen über Service-orient. Prozessmodelle hin zu den Code-Artefakten ausführbarer Komponenten eines Service-orient. Anwendungssystems zu großen Teilen automatisch vollzieht. Gleichzeitig war die konzeptionelle Service-orient. Architektur für virt. Dienstleistungen mittels Entwurf einer physikalischen Architektur zu konkretisieren, die die Komponenten Service-orient. Anwendungssysteme auf das Programmiermodell einer Service-orient. Middleware abbildet und Mechanismen zur Verwaltung der dadurch repräsentierten betrieblichen Ressourcen einbezieht. Die hierzu antizipierte Strategie basierte auf dem Einsatz Grid-basierter Erweiterungen Service-orient. Basistechniken, um mittels Grid Service-Komponenten die Ressourcen virt. Dienstleistungsunternehmen zu repräsentieren und im Verlauf der Produktionssteuerung individuell zu verwalten.

Untersuchungsergebnisse und Beiträge

Die Untersuchungen der vorliegenden Arbeit zur angestrebten Entwicklung des Rahmenwerks setzten bei den ökonomischen Grundlagen an. Hier wurden zunächst die Charakteristik ökonomischer Dienstleistungen und die Determinanten der Dienstleistungsproduktion analysiert. Prozesscharakter und Interaktivität konnten dabei als wesentliche Eigenschaften erkannt werden, woraus sich alleine schon die informationstechnische Unterstützung von Dienstleistungsprozessen als entscheidender Faktor der Produktivitätssteigerung ableiten ließ. In Bezug auf diese Erkenntnisse wurden vorteil-

hafte Formen für die Organisation von Dienstleistungsunternehmen und -produktion untersucht, was insbesondere Prozess- und Netzwerkorganisation betraf. Vor allem für deren Vereinigung als virt. Organisation wurden entscheidende Vorteile vermutet und konnten durch Vorarbeiten bestätigt werden. Entsprechend wurden von nun an virt. Dienstleistungsunternehmen fokussiert und in Bezug auf das Management von virt. Dienstleistungsproduktionsnetzwerken vertieft. Insbesondere konnte hier durch ein fundiertes Konzept zur interaktionsbasierten Regelung der Koordination kooperativer Vorgänge in virt. Unternehmen eine entscheidende Verknüpfung mit der ursprünglichen Zielsetzung und insgesamt ein tragfähiges theoretisches Fundament gefunden werden. Um dies auch praktisch zu verankern, wurde zudem ein Szenario mit exemplarischem Geschäftsfall im Kontext der Güterlogistik einbezogen, dessen Basis durch eine Impact-Studie abgesichert ist und dessen Relevanz in Hinblick auf die fokussierten Aspekte virtueller Organisation und Dienstleistungsproduktion eindeutig gezeigt werden konnte.

An dieser Stelle wechselte die Untersuchung zu den technischen Grundlagen. Als Ausgangspunkt der Betrachtung wurde die betriebliche Informationsverarbeitung gewählt und insbesondere deren Integrationsaspekt herausgestellt, der im Kontext virt. Organisation wesentliche Bedeutung als Schlüsseltechnologie für die Fähigkeit zur flexiblen Kooperation von Organisationseinheiten besitzt. Diese Bedeutung wurde für VDU im Detail herausgearbeitet. Das Ergebnis war eine Anforderungsanalyse für die IV-Infrastruktur von VDU, die einen Beitrag zum Verständnis dieser IT-Anwendungsdomäne liefert. Hieraus ergaben sich drei IT-Bereiche mit spezifischer Bedeutung in Hinblick auf die Zielsetzung. Mit Enterprise-Architektur, Middleware und Workflow wurden Basistechniken zur Integration betrieblicher Modelle, Methoden und Mechanismen zur Integration von Anwendungssystem-Komponenten sowie zur Integration von und mit schematisierten Arbeitsprozessen vorgestellt, zeigten sich jedoch im VDU-Kontext als unzureichend. Daher erfolgte eine Analyse des Service Oriented Computing-Paradigmas, das insbesondere die Web Service-Abstraktion, zwischenbetriebliche Middleware-Protokolle und erfolgreiche Standardisierung beisteuert und dadurch ein deutliches Potenzial zum Einsatz im VDU aufwies. Dies wurde noch durch Techniken der Web Service-Interaktion auf Basis von Koordinationsprotokollen und Kompositionsschemata bestärkt, die in Hinblick auf die angestrebte Virtualisierung prozessbasierter und stark interaktiver Dienstleistungen optimale Voraussetzungen boten. Daher wurde auch deren Verwendung zur Prozessintegration im Detail untersucht und eine Klassifikation entsprechender Ansätze herausgearbeitet, die einen eigenständigen Beitrag allg. Service-orient. Software-Entwicklung darstellt.

Nach ausführlicher Untersuchung ökonomischer und technischer Grundlagen ging die Darstellung zur Konzeption im Sinne der Zielsetzung über. Hier wurde auf ökonomischer Ebene ein abstraktes konzeptionelles Modell für virt. Dienstleistung und für deren Produktion in virt. Dienstleistungsproduktionsnetzwerken entwickelt, das die Virtualisierung interaktiver Vorgänge und Abläufe im Dienstleistungsprozess beschreibt, auf Basis des dadurch entstandenen virt. Dienstleistungsprozesses die

Koordination kooperativer Produktionsvorgänge von VDPN-Teilnehmern regelt und diesen Regelmechanismus zur Planung und Steuerung der Dienstleistungsproduktion nutzt. Der konstruktive Beitrag dieses Modells zeigte sich sofort durch dessen exemplarische Verfeinerung. Diese führte zum konzeptionellen VDL-Modell des FRESCO-Projekts, das eine spezifische Strukturierung von VDL in Assets, Demands und Capabilities beinhaltet. Hierbei wird der VDLP in Capabilities untergliedert, die je nach Bedarf individuell zusammengesetzt werden können und damit die agile Rekonfiguration von VDPN begründen. Es folgten Untersuchungen zur Realisierung konzeptioneller VDL auf Basis der IT-Infrastruktur von VDU. Deren Ergebnis und Beitrag ist die Einführung von E-Services als Klasse integrativer Anwendungssysteme, durch die VDL zum Zweck des Produktionsmanagements in VDPN entworfen und implementiert werden, was wiederum als E-Service-Management definiert wurde. Diesbezüglich wurde der Einsatz von Service Oriented Computing-Techniken für den allgemeinen Fall hergeleitet und insbesondere für FRESCO-E-Services im Detail ausgeführt. Das Ergebnis ist ein spezifisches Service-orient. Modell für E-Services und insbesondere das FRESCO E-Service-Metamodell, mit dem auf Grundlage des generischen Workflow-Prozess-Metamodells der WfMC technikneutrale Metamodelle für Web Service-Koordinationsprotokolle und -Kompositionsschemata sowie vor allem eine Service-orient. Architektur für verteilte Anwendungssysteme zur Realisierung virt. Dienstleistungen entwickelt wurden. Als abschließender Aspekt wurde ein Konzept für das E-Service-Management als Synthese von Software-Entwicklung und Produktionsmanagement entwickelt. Hierzu erfolgte zunächst die Einordnung entsprechender Vorgehensmodelle mitsamt Modellen, Methoden und Mechanismen in das GERAM Enterprise-Architektur-Rahmenwerk, um die Möglichkeit zu deren Einbettung in den betrieblichen Gesamtkontext als fundamentale Eigenschaft zu sichern. Daraufhin wurde ein generischer Service-orient. Entwicklungslebenszyklus erarbeitet und dem Lebenszyklus virt. Unternehmen derart angepasst, dass Letzterer durch die E-Service-Entwicklung optimal unterstützt wird. Dessen Beitrag besteht in der Vorgabe des Verfahrens zur Verfeinerung in konkreten Ansätzen, was in exemplarischer Weise durch Entwicklung des FRESCO E-Service-Management-Vorgehensmodell veranschaulicht wurde. Das Ergebnis ist eine konkrete Anleitung zur Planung und Steuerung der virt. Dienstleistungsproduktion mit FRESCO-E-Services.

Die Untersuchung führte dann von der Konzeption zur technischen Realisierung. Hier wurden zunächst E-Service-Management-Systeme als Klasse von Integrationsplattformen eingeführt, die ein spezifisches E-Service-Management-Vorgehensmodell durch Methoden und Mechanismen zur Entwicklung und Ausführung einer spezifischen Form von E-Service realisieren. In der Folge wurden solche Methoden und Mechanismen als generelle Prinzipien für allgemeine E-Services sowie exemplarisch für den Fall von FRESCO E-Services entworfen und Letztere dann auch prototypisch implementiert. Ein erstes Ergebnis besteht im generellen Prinzip zur modellgetriebenen E-Service-Entwicklung, in dem u. a. notwendige Modellebenen und Transformationen identifiziert werden. Ein weiteres Ergebnis besteht im generellen Architekturprin-

zip für Service-orient. E-Service Grids auf Basis von OGSA, das Ressourcen und Engines als Komponenten einer E-Service-Implementierung festlegt und die Ressourcenverwaltung durch Grid Service Repositories und Factories einer virtuellen Ausführungsumgebung vorgibt. Das generelle Entwicklungsprinzip erfuhr eine Konkretisierung durch die FRESCO-Methodik mit grafischem Entwurf von E-Service-Modellen in UML und Generierung XPDL-basierter E-Service-Schemata, mit systematischer Änderung von E-Service-Schemata durch regelbasierte Transformation und mit Überführung von E-Service-Schemata in ein Ausführungsformat durch Generierung von BPEL-Schemata. Das generelle Architekturprinzip wurde mittels spezifischer Anwendungskomponenten in Bezug auf E-Service-Schemata und Plattformkomponenten mit Laufzeitmechanismen zur Implementierung und Aggregation von E-Service-Instanzen konkretisiert. Auf Basis dieser beiden Entwürfe wurde mit dem FRESCO-TK ein prototypisches ESMS implementiert. Dieses ergänzt COTS-Komponenten für OGSI Container und BPEL-Laufzeitumgebungen um Architekturspezifikationen und Software-Rahmenwerke für Anwendungs- und Plattformkomponenten zu einer horizontalen IT-Infrastruktur. Darauf setzen Implementierungen der FRESCO-Mechanismen als Werkzeuge und Plattformkomponenten auf, die vertikale Fachfunktionen zur Planung und Steuerung virt. Dienstleistungsproduktion bilden.

Schließlich bildete eine Evaluation der Ergebnisse den Abschluss der Untersuchung. Hierbei wurden stellvertretend für die abstrakten Anteile deren Konkretisierungen im Kontext von FRESCO herangezogen, um durch Anwendung auf das praktische Szenario deren Anwendbarkeit und Nutzen zur Planung und Steuerung virt. Dienstleistungsproduktion zu zeigen. Hier ergab sich auf konzeptioneller Ebene eine ausreichende Übereinstimmung organisatorischer Strukturen und Abläufe, so dass die Akteure des FRESCO VDPN-Modells identifiziert und in Bezug auf ihre Eignung verifiziert werden können, das FRESCO E-Service-Management-Vorgehensmodell den exemplarischen Geschäftsfall unterstützt und dessen Dienstleistung sich mittels des FSM virtualisieren lässt. Auf praktischer Ebene wurde dann die Anwendung von Methoden und Mechanismen des FRESCO-TK im Geschäftsfall demonstriert. Dadurch konnten die im Szenario geforderten Eigenschaften beim praktischen Einsatz des Rahmenwerks aufgezeigt werden. Dessen Nutzen in Bezug auf die angestrebte Unterstützung der Planung und Steuerung virt. Dienstleistungsproduktion ließ sich so plausibel machen.

7.2. Bewertung

Die nachfolgende Bewertung untersucht den Erfolg der vorliegenden Arbeit anhand verschiedener Kriterien. In erster Linie müssen dazu Ziele und Ergebnisse abgeglichen werden. Darüber hinaus sind die erkennbaren Beiträge von Interesse. Diese sind nicht nur quantitativ nach der Menge von z. B. Modellen, Methoden und Mechanismen zu bewerten. Auch Qualitäten wie die Breite der Betrachtung und die interdisziplinäre Charakteristik des Ansatzes sind zu vermerken.

Bevor die Ziele der vorliegenden Arbeit in Hinblick auf deren Erfüllung überprüft werden, sollten diese zunächst selbst evaluiert werden. Ihnen liegen Annahmen zugrunde, die heute nicht als gesichertes Allgemeinwissen gelten können. Hiermit ist zum einen die Aussage gemeint, dass Dienstleistungen optimalerweise in virt. Unternehmen zu produzieren sind. Zum anderen ist dies die Annahme vorteilhafter Eigenschaften Service-orient. Techniken zur Integration und Interaktion im Kontext virt. Dienstleistungsunternehmen. Es kann jedoch festgestellt werden, dass diese Vorbedingungen in den Grundlagenteilen der vorliegenden Arbeit bestätigt wurden. Auf der einen Seite wurde bei der Aufarbeitung ökonomischer Grundlagen die Vorteilhaftigkeit virt. Dienstleistungsproduktion auf Basis konstitutiver Dienstleistungsmerkmale und fundierter Organisationstheorien eindeutig hergeleitet. Zudem konnte der Einfluss von Interaktionsprozessen gezeigt und so als Ansatzpunkt für informationstechnische Unterstützungsmechanismen des Produktionsmanagements bestätigt werden. Diese Ergebnisse mögen als Beitrag zum jungen Feld der *Service Science* eine gewisse eigenständige Bedeutung besitzen. Auf der anderen Seite lieferte die Aufarbeitung technischer Grundlagen eine strukturierte Anforderungsanalyse an die integrative IT-Infrastruktur für VDU, in die sich SOC-Techniken in Hinblick auf eine Analyse Service-orient. Prozessintegration mit eindeutiger Vorteilhaftigkeit eingliedern ließen. Beide in diesem Zusammenhang erarbeiteten Analysen können darüber hinaus wiederum als eigenständige Beiträge angesehen werden.

Vor diesem Hintergrund soll nun zunächst die Erfüllung konzeptioneller Ziele untersucht werden. Das hier angestrebte integrierte ökonomische Konzept zur Steuerung der Dienstleistungsproduktion in virt. Unternehmen auf Basis von Dienstleistungsvirtualisierung liegt sowohl in genereller als auch in FRESCO-spezifischer Form der konzeptionellen VDL- und VDPN-Modelle vor. Dem Wunsch nach einem Softwaretechnischen Konzept zur IT-Abbildung von VDL wurde mit der Einführung des allg. E-Service-Begriffs und dessen detaillierter Service-orient. Ausgestaltung durch das FRESCO E-Service-Metamodell entsprochen. Schließlich liegt die anvisierte Synthese eines Vorgehensmodells zur VDL-basierten Produktionssteuerung in Form des abstrakten Service-orient. E-Service-Entwicklungslebenszyklus sowie dessen Konkretisierung als FRESCO E-Service-Management-Vorgehensmodell vor und wurde durch Verknüpfung mit GERAM in einen fundierten Kontext gestellt. Alle konzeptionellen Ergebnisse wurden durch ihre Anwendbarkeit im Szenario bestätigt. Die konzeptionelle Zielsetzung wurde somit erfüllt.

Nachfolgend gilt die Untersuchung der Erfüllung technischer Ziele. Die hier geforderte Umsetzung des Vorgehensmodells VDL-basierter Produktionssteuerung durch einen Ansatz modellgetriebener Entwicklung erfolgte in konzeptioneller Form durch das Prinzip zur modellgetriebenen E-Service-Entwicklung und durch dessen Anwendung im Rahmen der konkreten FRESCO-Methodik sowie in konkreter Form durch horizontale Fachfunktionen des FRESCO-TK. Die nachgefragte physikalische Architektur für IT-Abbildungen virt. Dienstleistung wurde durch das generelle Architekturprinzip der E-Service Grids und dessen Konkretisierung im Entwurf der

FRESCO ESMS-Plattform konzeptionell verwirklicht und durch die horizontale IT-Infrastruktur des FRESCO-TK implementiert. Beide technischen Ergebnisse wurden in ihrer konkreten Form durch Anwendung auf den exemplarischen Geschäftsfall des Szenarios funktional verifiziert und in Bezug auf ihre grundsätzliche Qualität in Hinblick auf die Unterstützung virt. Dienstleistungsproduktion als effektiv bewertet. Die technischen Ziele sind damit ebenfalls erreicht worden.

Nachdem somit gezeigt wurde, dass alle wesentlichen Ziele der vorliegenden Arbeit grundsätzlich erreicht wurden, sollen noch einige andere Qualitäten herausgestellt werden. Hierzu zählt zum einen die grundlegend interdisziplinäre Natur der Untersuchung. Hierdurch leistet die vorliegende Arbeit einen Beitrag zur Überbrückung von Unterschieden in der Sichtweise und zur Zusammenführung von isolierten Ergebnissen in den Bereichen der Betriebswirtschaft für Dienstleistungen und der Software-Technik Service-orient. Architekturen. Die Problematik unterschiedlicher Sichtweisen wurde in den letzten Jahren z. B. und insbesondere in Bezug auf den Dienstleistungsbegriff deutlich. Wie in Sektion 3.4.1.1 angedeutet wurde, konnte in Bezug auf dessen Zusammenhang mit dem Web Service-Begriff vor allem auf der Seite der Software-Technik eine Bandbreite freier Assoziationen im Wesentlichen ohne Bezug zu ökonomischen Grundlagen beobachtet werden, wobei die häufigste Auslegung die Identität oder ein BPEL-Schema war. Dass die Abbildung von Dienstleistungen vor allem auf einzelne Web Services, aber auch auf einzelne BPEL-Schemata, als problematisch gelten muss, sollte nach den Ausführungen der vorliegenden Arbeit in Bezug auf die Struktur des interaktiven Dienstleistungsprozesses erkennbar sein. Auf der anderen Seite hebt die Untersuchung betriebswirtschaftliche und organisatorische Anwendungsfelder hervor, in denen die Kombination von Qualitäten der SOC-Techniken mit ökonomischen Theorien zu effektiven Ergebnissen führen kann und hinterlegt dies durch ihre Ergebnisse. Nicht zuletzt sollte dies auch Motivation zur Verfolgung interdisziplinärer Ansätze in verwandten Bereichen sein. In diesem Zusammenhang ist besonders die ausführliche Darstellung von Grundlagen beider Disziplinen hervorzuheben. Deren Beitrag besteht in der Einführung von Mitgliedern einer der Disziplinen in die jeweils andere.

Eine andere Qualität der vorliegenden Arbeit liegt in der Breite ihrer technischen Betrachtung. Die Verfolgung der Zielsetzung machte es notwendig, eine für wissenschaftliche Betrachtungen sonst eher ungewöhnliche und auch nicht unproblematische Vielzahl technischer Modelle, Methoden und Mechanismen über den und inklusive des gesamten Service-orient. Entwicklungslebenszyklus im direkten Zusammenhang einzubeziehen. Aus diesem Umstand resultierten vielfältige Probleme in Hinblick auf die Integration von Systemkomponenten, wie z. B. OGSI Container und BPEL-Laufzeitumgebungen, und in Hinblick auf die schiere Komplexität des ganzheitlichen Software-Systems. Diesen Problemen musste sich die Untersuchung stellen. Hierdurch wurden Ergebnisse produziert, die die alltäglichen Umstände praktischer Software-Entwicklung einbeziehen.

7.3. Ausblick

Die abschließende Sektion der vorliegenden Arbeit gibt einen Ausblick auf mögliche Richtungen der zukünftigen Forschung und Entwicklung aufbauend auf oder im Zusammenhang mit den Erkenntnissen der hier geführten Untersuchung und ihren Ergebnissen. In der vorliegenden Arbeit lag der Schwerpunkt der Betrachtungen auf Aspekten im Zusammenhang mit Interaktionsprozessen zur Regelung der Koordination kooperativer Vorgänge zwischen autonomen Kooperationspartnern. Deren Untersuchung erfolgte in einem großen Raum, der durch verschiedene Disziplinen und eine große Bandbreite verschiedener Techniken aufgespannt wurde, zwangsläufig auf relativ isolierte Weise. Infolgedessen sind Ergänzungen in Bezug auf sehr viele Aspekte denkbar. Gleichwohl macht es wenig Sinn, im Zusammenhang mit den fokussierten Ansätzen der vorliegenden Arbeit Bereiche anzusprechen, die zwar irgendwo im Raum angrenzen, in Bezug auf die fokussierte Thematik aber orthogonal verlaufen. Beispiele sind aus ökonomischer Sicht der generelle Aspekt des Netzwerkmanagements oder aus technischer Sicht der generelle Aspekt der Sicherheit, die natürlich unter Gesichtspunkten der Anwendung einer E-Service-Technologie von herausragender Bedeutung sind, deren Kombination mit der fokussierten Thematik aber aus wissenschaftlicher Sicht wenige direkte gemeinsame Fragestellungen aufwirft. Solche Aspekte werden hier daher nicht weiter erwähnt. Stattdessen soll auf einige Aspekte in unmittelbarem Zusammenhang kooperativer zwischenbetrieblicher Interaktionsprozesse verwiesen werden.

Ein solcher Aspekt mit unmittelbarer Relevanz ist die Abstimmung von Interaktionsprozessen zwischen VDPN-Teilnehmern. In der vorliegenden Arbeit wurde ein Ansatz zur Service-orient. Prozessintegration mit freier Regelung des kollaborativen Ablaufs auf Basis von Koordinationsprotokollen verfolgt. Im FRESCO E-Service-Management-Vorgehensmodell wird beschrieben, wie VDPN-Teilnehmer einzelne Interaktionsmuster des virt. Dienstleistungsprozesses als Capabilities spezifizieren und sich wechselseitig zur Vorgabe machen. Hierbei wurden jedoch keine spezifischen Methoden oder Mechanismen zur Abstimmung von Capabilities vorgegeben. Gleichwohl ist dies unter Umständen keine leichte Aufgabe und sollte unterstützt werden. Mittlerweile sind verschiedene Ansätze zur Prüfung der Kompatibilität von Web Service-Koordinationsprotokollen vorgeschlagen worden,[1] die eine potenziell sinnvolle Ergänzung des technischen Rahmenwerks darstellen. In diesem Zusammenhang stellt sich auch die Frage nach Möglichkeiten zur Evaluierung von Eigenschaften des ganzheitlichen virt. Dienstleistungsprozesses. Eine derartige Funktionalität könnte virt. Dienstleistungsproduktionsnetzwerk-Brokern beim E-Service-Entwurf auf Basis von Provider-Capabilities unter Umständen eine Hilfestellung sein, um z. B. die Lebendigkeit des Prozesses zu verifizieren. Auch hier sind einige Ansätze für Web Service-

[1] Siehe z. B. Session III in [ZOL+05].

Interaktionen, z. B. auf Basis von Petrinetzen oder LTS, vorgeschlagen worden,[2] für die jedoch das fundamentale Prozessmodell des FRESCO E-Service-Metamodells in den meisten Fällen nicht die notwendigen Voraussetzungen bietet. Gleichwohl ist ggf. über entsprechende Erweiterungen nachzudenken.

Eine Fragestellung, die sich an die Untersuchungen und Ergebnisse vor allem im konzeptionellen Teil des Rahmenwerks anschließt, bezieht sich auf Möglichkeiten zur weiteren Steigerung der Flexibilität von E-Service-Schemata. Der vorliegende Ansatz verwendet Interaktionsmuster in Form von Capabilities, aus denen ein Broker den virt. Dienstleistungsprozess im Rahmen eines E-Service-Modells entwirft. Capabilities spezifizieren dabei den Ablauf der Interaktion und regeln dadurch die Koordination kooperativer Vorgänge zwischen Teilnehmern. E-Service-Schemata könnten wesentlich an Flexibilität gewinnen, falls Capabilities auf eine höhere Abstraktionsebene angehoben würden, auf der sie ein Fragment der Kooperationslogik repräsentieren würden, ohne dessen Koordination auf Basis spezifischer Interaktionsabläufe festzulegen. Letzteres könnte dann dynamisch durch Anwendung von Transformationsregeln zur Zeit einer Anfrage (Anfang Stufe 4 des FSMV) geschehen. Die Produktion einer Dienstleistung gleichen Inhalts, könnte dann von Fall zu Fall durch gänzlich unterschiedliche VDLP gesteuert werden, mit denen vor allem individuelle qualitative Anforderungen berücksichtigt werden könnten. Dieser Ansatz bedingt u. a. die Repräsentation von Koordinationsstrategien in Form von Transformationsregeln und die Aufstellung von Entscheidungsregeln, mit denen die optimale Koordinationsstrategie aus den Anforderungen eines Geschäftsfalls abgeleitet werden kann. Die Untersuchung dieser Fragestellung liegt im zentralen Interesse des Autors der vorliegenden Arbeit und führte schon zu ersten Ergebnissen [ZLB04, ZL04b].

[2] Siehe z. B. [MPP02, FUM+06].

A. FRESCO-TK E-Service-Schema-Spezifikationsarchive

A.1. E-Service-Schema-Archive

FRESCO E-Service-Schema-Spezifikationen bestehen aus einer Menge zusammenhängender Teilspezifikationen in Form von XML-Dokumenten[1]. Dazu gehören XPDL-basierte Beschreibungen von E-Service-Shell, -Interaktionskontext und Capabilities. Zudem sind WSDL-basierte Schnittstellenbeschreibungen von E-Service-Ressourcen und -Engines in ihren Erscheinungsformen als Web und Grid Services enthalten. Zur Integration dieser Dokumente sieht das FRESCO-TK ein Archivformat vor. Dieses Format beinhaltet Konventionen in Bezug auf

- *Teilspezifikationen* eines Archivs,
- *Ordnerstruktur* zur Ablage von Teilspezifikationen im Dateisystem,
- *URI-Struktur* zur Ablage von Teilspezifikationen im Schema-Repository,
- *XML-Namensräume* der Teilspezifikationen.

Die entsprechenden Konventionen des Archivformats werden in Abb. A.1 zusammengefasst. Das Diagramm gibt in erster Linie die Ordnerstruktur des Archivs wieder. Ordner (≪*Ordner*≫) sind dunkelgrau dargestellt und jeweils mit einem korrespondierenden URI-Ort (≪*Location*≫) ergänzt. Teilspezifikationen sind hellgrau dargestellt und jeweils mit einem Namensraum (≪*Namensraum*≫) versehen. Ein entsprechendes Archiv im Dateisystem dient als Eingabe für den FRESCO-TK E-Service-Schema-Manager. Dieser führt eine Auswertung durch und lädt das Archiv in das E-Service-Schema-Repository. Dort ist es an den spezifizierten Orten durch andere Plattformkomponenten zugreifbar.

In den verbleibenden Sektionen dieses Anhangs werden verschiedene Teilspezifikationen einer exemplarischen E-Service-Schema-Spezifikation als Referenz gezeigt. Es handelt sich dabei um Auszüge aus der Schema-Spezifikation des in Kapitel 6 entwickelten FreightMixer-E-Service. Die Darstellung beschränkt sich bei den Schnittstellen und Interaktionsprozessen auf jeweils eine exemplarische Spezifikation.

[1] Siehe Sektion 5.3.4.1, S. 413.

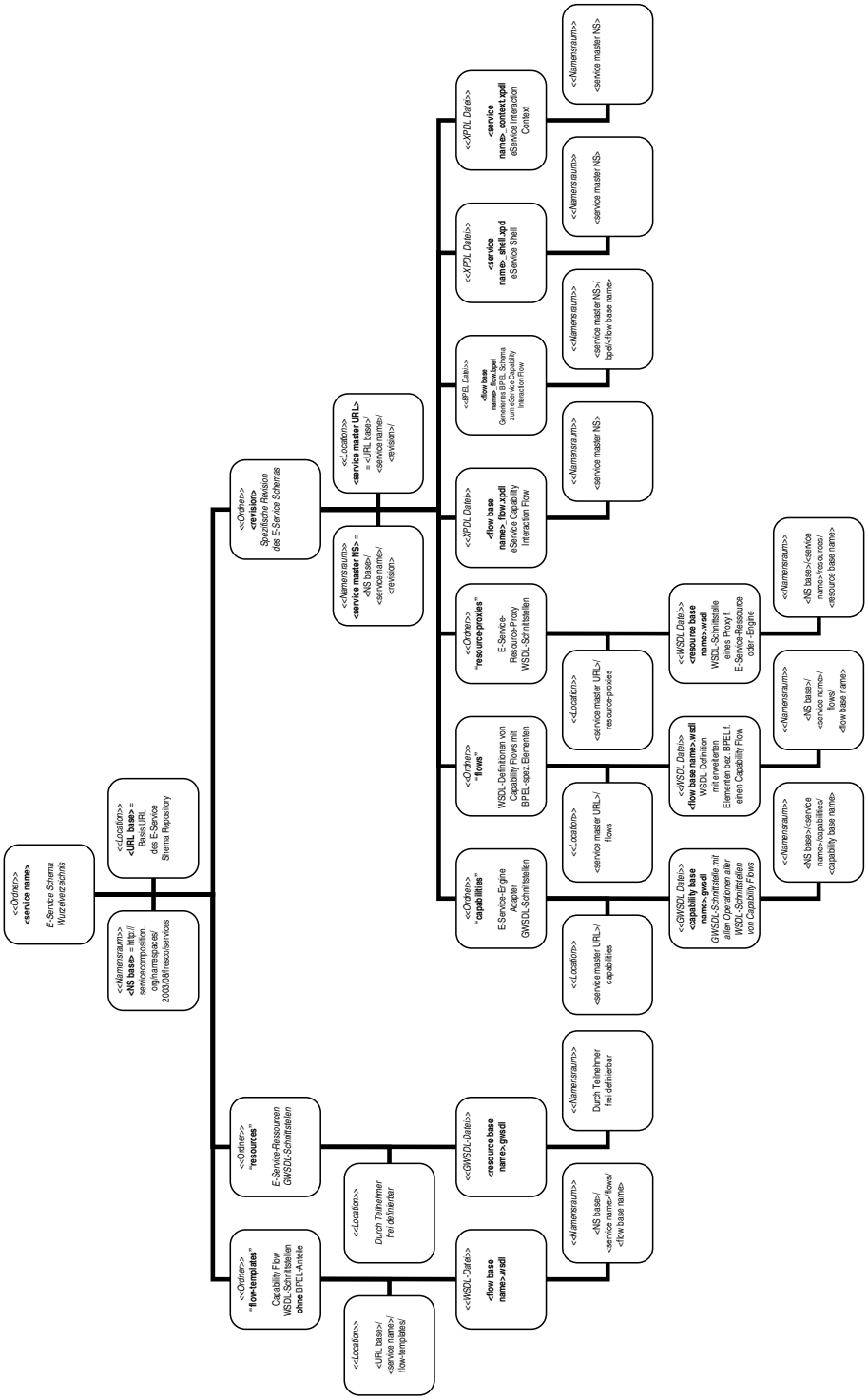

Abbildung A.1.: Aufbau von E-Service-Schema-Archiven

A.2. E-Service-Schema-Spezifikation für das Transportbeispiel

A.2.1. Flow-Template WSDL

```
<definitions name="HandoverFlow"
  targetNamespace="http://servicecomposition.org/namespaces/2003/08/fresco/
   services/Transport/flows/HandoverFlow"
  xmlns:tns="http://servicecomposition.org/namespaces/2003/08/fresco/
   services/Transport/flows/HandoverFlow"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types/>
  <message name="startInputMessage">
    <!-- proxy parameters -->
    <part name="serviceId" type="xsd:string"/>
    <!-- resource parameters -->
    <part name="transportId" type="xsd:string"/>
    <part name="transportLegId" type="xsd:int"/>
  </message>
  <message name="startOutputMessage">
    <part name="acknowledge" type="xsd:boolean"/>
  </message>
  <message name="statusReportInputMessage">
    <!-- proxy parameters -->
    <part name="serviceId" type="xsd:string"/>
    <!-- resource parameters -->
    <part name="transportId" type="xsd:string"/>
    <part name="transportLegId" type="xsd:int"/>
    <part name="statusId" type="xsd:string"/>
  </message>
  <message name="statusReportOutputMessage">
    <part name="acknowledge" type="xsd:boolean"/>
  </message>
  <portType name="HandoverFlowPortType">
    <operation name="start">
      <input message="tns:startInputMessage" name="startRequest"/>
      <output message="tns:startOutputMessage" name="startResponse"/>
    </operation>
    <operation name="statusReport">
      <input message="tns:statusReportInputMessage" name="statusReportRequest"/>
      <output message="tns:statusReportOutputMessage" name="statusReportResponse"/>
    </operation>
  </portType>
</definitions>
```

A.2.2. Resource GWSDL

```
<definitions name="TransportControl"
  targetNamespace="http://servicecomposition.org/namespaces/2003/08/fresco/
   samples/Transport/TransportControl"
  xmlns:tns="http://servicecomposition.org/namespaces/2003/08/fresco/
   samples/Transport/TransportControl"
  xmlns:fco="http://servicecomposition.org/namespaces/2003/08/fresco/core/base"
  xmlns:fcr="http://servicecomposition.org/namespaces/2003/08/fresco/core/resource"
  xmlns:gwsdl="http://www.gridforum.org/namespaces/2003/03/gridWSDLExtensions"
  xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI"
  xmlns:sd="http://www.gridforum.org/namespaces/2003/03/serviceData"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <import location="../ogsi/ogsi.gwsdl"
```

```
  namespace="http://www.gridforum.org/namespaces/2003/03/OGSI"/>
<import location="fresco.gwsdl"
  namespace="http://servicecomposition.org/namespaces/2003/08/fresco/core/base"/>
<import location="frescoResource_port_type.gwsdl"
  namespace="http://servicecomposition.org/namespaces/2003/08/fresco/core/resource"/>
<types>
  <schema
    targetNamespace=
      "http://servicecomposition.org/namespaces/2003/08/fresco/
       samples/Transport/TransportControl"
    xmlns="http://www.w3.org/2001/XMLSchema">
    <element name="upcomingTransport">
      <complexType>
        <sequence>
          <element name="transportId" type="string"/>
          <element name="transportLegId" type="int"/>
        </sequence>
      </complexType>
    </element>
    <element name="upcomingTransportResponse">
      <complexType>
        <sequence>
          <element name="acknowledge" type="boolean"/>
        </sequence>
      </complexType>
    </element>
    <element name="requestProblemReport">
      <complexType>
        <sequence>
          <element name="transportId" type="string"/>
          <element name="transportLegId" type="int"/>
        </sequence>
      </complexType>
    </element>
    <element name="requestProblemReportResponse">
      <complexType>
        <sequence>
          <element name="acknowledge" type="boolean"/>
        </sequence>
      </complexType>
    </element>
  </schema>
</types>
<message name="upcomingTransportInputMessage">
  <part element="tns:upcomingTransport" name="parameters"/>
</message>
<message name="upcomingTransportOutputMessage">
  <part element="tns:upcomingTransportResponse" name="parameters"/>
</message>
<message name="requestProblemReportInputMessage">
  <part element="tns:requestProblemReport" name="parameters"/>
</message>
<message name="requestProblemReportOutputMessage">
  <part element="tns:requestProblemReportResponse" name="parameters"/>
</message>
<gwsdl:portType extends="fcr:FrescoResource" name="TransportControlPortType">
  <operation name="upcomingTransport" parameterOrder="">
    <input message="tns:upcomingTransportInputMessage"
      name="upcomingTransportRequest"/>
    <output message="tns:upcomingTransportOutputMessage"
      name="upcomingTransportResponse"/>
    <fault name="Fault" message="ogsi:FaultMessage"/>
  </operation>
```

```
    <operation name="requestProblemReport" parameterOrder="">
      <input message="tns:requestProblemReportInputMessage"
        name="requestProblemReportRequest"/>
      <output message="tns:requestProblemReportOutputMessage"
        name="requestProblemReportResponse"/>
      <fault name="Fault" message="ogsi:FaultMessage"/>
    </operation>
  </gwsdl:portType>
</definitions>
```

A.2.3. Flow WSDL

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<definitions name="HandoverFlow"
  targetNamespace="http://.../services/Transport/rev08/flows/HandoverFlow"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
  xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI"
  xmlns:resource1="http://.../services/Transport/rev08/resources/TransportCoordination"
  xmlns:resource2="http://.../services/Transport/rev08/resources/TransportControl"
  xmlns:resource3="http://.../services/Transport/rev08/resources/InsuranceControl"
  xmlns:resource4="http://.../services/Transport/rev08/resources/TransportMonitoring"
  xmlns:resource5="http://.../services/Transport/rev08/resources/LogisticManagement"
  xmlns:resource6="http://.../services/Transport/rev08/resources/TransportCompletion"
  xmlns:slnk="http://schemas.xmlsoap.org/ws/2003/03/service-link/"
  xmlns:tns="http://.../services/Transport/rev08/flows/HandoverFlow"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <types>
  </types>
  <message name="_intMessage">
    <part name="value" type="xsd:int"/>
  </message>
  <message name="startInputMessage">
    <part name="serviceId" type="xsd:string"/>
    <part name="transportId" type="xsd:string"/>
    <part name="transportLegId" type="xsd:int"/>
  </message>
  <message name="startOutputMessage">
    <part name="acknowledge" type="xsd:boolean"/>
  </message>
  <message name="statusReportInputMessage">
    <part name="serviceId" type="xsd:string"/>
    <part name="transportId" type="xsd:string"/>
    <part name="transportLegId" type="xsd:int"/>
    <part name="statusId" type="xsd:string"/>
  </message>
  <message name="_booleanMessage">
    <part name="value" type="xsd:boolean"/>
  </message>
  <message name="statusReportOutputMessage">
    <part name="acknowledge" type="xsd:boolean"/>
  </message>
  <message name="_stringMessage">
    <part name="value" type="xsd:string"/>
  </message>
  <portType name="HandoverFlowPortType">
    <operation name="start">
      <input name="startRequest" message="tns:startInputMessage"/>
      <output name="startResponse" message="tns:startOutputMessage"/>
    </operation>
    <operation name="statusReport">
      <input name="statusReportRequest" message="tns:statusReportInputMessage"/>
```

```
    <output name="statusReportResponse"
      message="tns:statusReportOutputMessage"/>
  </operation>
</portType>
<service name="HandoverFlowService">
</service>
<slnk:serviceLinkType name="HandoverFlowPLT">
  <slnk:role name="HandoverFlowRole">
    <slnk:portType name="tns:HandoverFlowPortType"/>
  </slnk:role>
</slnk:serviceLinkType>
<slnk:serviceLinkType name="HandoverFlowPLT">
  <slnk:role name="HandoverFlowRole">
    <slnk:portType name="tns:HandoverFlowPortType"/>
  </slnk:role>
</slnk:serviceLinkType>
<slnk:serviceLinkType name="TransportCoordinationPLT">
  <slnk:role name="TransportCoordinationRole">
    <slnk:portType name="resource1:TransportCoordinationPortType"/>
  </slnk:role>
</slnk:serviceLinkType>
<slnk:serviceLinkType name="TransportControlPLT">
  <slnk:role name="TransportControlRole">
    <slnk:portType name="resource2:TransportControlPortType"/>
  </slnk:role>
</slnk:serviceLinkType>
<slnk:serviceLinkType name="InsuranceControlPLT">
  <slnk:role name="InsuranceControlRole">
    <slnk:portType name="resource3:InsuranceControlPortType"/>
  </slnk:role>
</slnk:serviceLinkType>
<slnk:serviceLinkType name="TransportMonitoringPLT">
  <slnk:role name="TransportMonitoringRole">
    <slnk:portType name="resource4:TransportMonitoringPortType"/>
  </slnk:role>
</slnk:serviceLinkType>
<slnk:serviceLinkType name="LogisticManagementPLT">
  <slnk:role name="LogisticManagementRole">
    <slnk:portType name="resource5:LogisticManagementPortType"/>
  </slnk:role>
</slnk:serviceLinkType>
<slnk:serviceLinkType name="TransportCoordinationPLT">
  <slnk:role name="TransportCoordinationRole">
    <slnk:portType name="resource1:TransportCoordinationPortType"/>
  </slnk:role>
</slnk:serviceLinkType>
<slnk:serviceLinkType name="TransportControlPLT">
  <slnk:role name="TransportControlRole">
    <slnk:portType name="resource2:TransportControlPortType"/>
  </slnk:role>
</slnk:serviceLinkType>
<slnk:serviceLinkType name="TransportCompletionPLT">
  <slnk:role name="TransportCompletionRole">
    <slnk:portType name="resource6:TransportCompletionPortType"/>
  </slnk:role>
</slnk:serviceLinkType>
<bpws:property name="ServiceIdType" type="xsd:string"/>
<bpws:property name="TransportLegIdType" type="xsd:int"/>
<bpws:propertyAlias propertyName="tns:ServiceIdType"
  messageType="tns:startInputMessage" part="serviceId"
  query="/serviceId"/>
<bpws:propertyAlias propertyName="tns:TransportLegIdType"
  messageType="tns:startInputMessage" part="transportLegId"
```

```
    query="/transportLegId"/>
  <bpws:propertyAlias propertyName="tns:ServiceIdType"
    messageType="tns:statusReportInputMessage" part="serviceId"
    query="/serviceId"/>
  <bpws:propertyAlias propertyName="tns:TransportLegIdType"
    messageType="tns:statusReportInputMessage" part="transportLegId"
    query="/transportLegId"/>
</definitions>
```

A.2.4. Capability GWSDL

```
<wsdl:definitions name="TransportCoordination" targetNamespace=
  "http://servicecomposition.org/namespaces/2003/08/fresco/
  services/Transport/rev04/capabilities/TransportCoordination"
  xmlns:tns="http://servicecomposition.org/namespaces/2003/08/fresco/
    services/Transport/rev04/capabilities/TransportCoordination"
  xmlns:fresco="http://servicecomposition.org/namespaces/2003/08/fresco/core/base"
  xmlns:fcr="http://servicecomposition.org/namespaces/2003/08/fresco/core/resource"
  xmlns:gwsdl="http://www.gridforum.org/namespaces/2003/03/gridWSDLExtensions"
  xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI"
  xmlns:sd="http://www.gridforum.org/namespaces/2003/03/serviceData"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <import location="../ogsi/ogsi.gwsdl"
    namespace="http://www.gridforum.org/namespaces/2003/03/OGSI"/>
  <import location="fresco.gwsdl"
    namespace="http://servicecomposition.org/namespaces/2003/08/fresco/core/base"/>
  <import location="frescoResource_port_type.gwsdl"
    namespace="http://servicecomposition.org/namespaces/2003/08/fresco/core/resource"/>
  <wsdl:types>
    <schema
      targetNamespace=
        "http://servicecomposition.org/namespaces/2003/08/fresco/
        services/Transport/rev04/capabilities/TransportCoordination"
      xmlns="http://www.w3.org/2001/XMLSchema">
      <element name="start">
        <complexType>
          <sequence>
            <element name="transportId" type="xsd:string"/>
            <element name="transportLegId" type="xsd:int"/>
          </sequence>
        </complexType>
      </element>
      <element name="startResponse">
        <complexType>
          <sequence>
            <element name="acknowledge" type="boolean"/>
          </sequence>
        </complexType>
      </element>
      <element name="awaitStatusReport">
        <complexType>
          <sequence>
            <element name="transportId" type="xsd:string"/>
            <element name="transportLegId" type="xsd:int"/>
            <element name="statusId" type="xsd:string"/>
          </sequence>
        </complexType>
      </element>
      <element name="awaitStatusReportResponse">
        <complexType>
```

```
        <sequence>
          <element name="acknowledge" type="boolean"/>
        </sequence>
      </complexType>
    </element>
    <element name="handleProblem">
      <complexType>
        <sequence>
          <element name="transportId" type="xsd:string"/>
          <element name="transportLegId" type="xsd:int"/>
        </sequence>
      </complexType>
    </element>
    <element name="handleProblemResponse">
      <complexType>
        <sequence>
          <element name="acknowledge" type="boolean"/>
        </sequence>
      </complexType>
    </element>
    <element name="awaitProblemReport">
      <complexType>
        <sequence>
          <element name="transportId" type="xsd:string"/>
          <element name="transportLegId" type="xsd:int"/>
          <element name="problemId" type="xsd:string"/>
        </sequence>
      </complexType>
    </element>
    <element name="awaitProblemReportResponse">
      <complexType>
        <sequence>
          <element name="acknowledge" type="boolean"/>
        </sequence>
      </complexType>
    </element>
  </schema>
</wsdl:types>
<wsdl:message name="startInputMessage">
  <wsdl:part element="tns:start" name="parameters"/>
</wsdl:message>
<wsdl:message name="startOutputMessage">
  <wsdl:part element="tns:startResponse" name="parameters"/>
</wsdl:message>
<wsdl:message name="awaitStatusReportInputMessage">
  <wsdl:part element="tns:awaitStatusReport" name="parameters"/>
</wsdl:message>
<wsdl:message name="awaitStatusReportOutputMessage">
  <wsdl:part element="tns:awaitStatusReportResponse" name="parameters"/>
</wsdl:message>
<wsdl:message name="handleProblemInputMessage">
  <wsdl:part element="tns:handleProblem" name="parameters"/>
</wsdl:message>
<wsdl:message name="handleProblemOutputMessage">
  <wsdl:part element="tns:handleProblemResponse" name="parameters"/>
</wsdl:message>
<wsdl:message name="awaitProblemReportInputMessage">
  <wsdl:part element="tns:awaitProblemReport" name="parameters"/>
</wsdl:message>
<wsdl:message name="awaitProblemReportOutputMessage">
  <wsdl:part element="tns:awaitProblemReportResponse" name="parameters"/>
</wsdl:message>
<gwsdl:portType extends="fcr:FrescoResource" name="TransportCoordinationPortType">
```

```
    <wsdl:operation name="start">
      <wsdl:input message="tns:startInputMessage"
       name="startRequest"/>
      <wsdl:output message="tns:startOutputMessage"
       name="startResponse"/>
      <fault name="Fault" message="ogsi:FaultMessage"/>
    </wsdl:operation>
    <wsdl:operation name="awaitStatusReport">
      <wsdl:input message="tns:awaitStatusReportInputMessage"
       name="awaitStatusReportRequest"/>
      <wsdl:output message="tns:awaitStatusReportOutputMessage"
       name="awaitStatusReportResponse"/>
      <fault name="Fault" message="ogsi:FaultMessage"/>
    </wsdl:operation>
    <wsdl:operation name="handleProblem">
      <wsdl:input message="tns:handleProblemInputMessage"
       name="handleProblemRequest"/>
      <wsdl:output message="tns:handleProblemOutputMessage"
       name="handleProblemResponse"/>
      <fault name="Fault" message="ogsi:FaultMessage"/>
    </wsdl:operation>
    <wsdl:operation name="awaitProblemReport">
      <wsdl:input message="tns:awaitProblemReportInputMessage"
       name="awaitProblemReportRequest"/>
      <wsdl:output message="tns:awaitProblemReportOutputMessage"
       name="awaitProblemReportResponse"/>
      <fault name="Fault" message="ogsi:FaultMessage"/>
    </wsdl:operation>
  </gwsdl:portType>
</wsdl:definitions>
```

A.2.5. Resource-Proxy WSDL

```
<wsdl:definitions
  targetNamespace="http://servicecomposition.org/namespaces/2003/08/fresco/
   services/Transport/rev04/resources/TransportControl"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:apachesoap="http://xml.apache.org/xml-soap"
  xmlns:impl="http://servicecomposition.org/namespaces/2003/08/fresco/
   services/Transport/rev04/resources/TransportControl"
  xmlns:intf="http://servicecomposition.org/namespaces/2003/08/fresco/
   services/Transport/rev04/resources/TransportControl"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <wsdl:message name="requestProblemReportRequest">
    <wsdl:part name="serviceId" type="xsd:string"/>
    <wsdl:part name="performer" type="xsd:string"/>
    <wsdl:part name="p0" type="xsd:string"/>
    <wsdl:part name="p1" type="xsd:int"/>
  </wsdl:message>
  <wsdl:message name="upcomingTransportRequest">
    <wsdl:part name="serviceId" type="xsd:string"/>
    <wsdl:part name="performer" type="xsd:string"/>
    <wsdl:part name="p0" type="xsd:string"/>
    <wsdl:part name="p1" type="xsd:int"/>
  </wsdl:message>
  <wsdl:message name="requestProblemReportResponse">
    <wsdl:part name="requestProblemReportReturn" type="xsd:boolean"/>
  </wsdl:message>
  <wsdl:message name="upcomingTransportResponse">
```

```
    <wsdl:part name="upcomingTransportReturn" type="xsd:boolean"/>
  </wsdl:message>
  <wsdl:portType name="TransportControlPortType">
    <wsdl:operation name="upcomingTransport"
     parameterOrder="serviceId performer p0 p1">
      <wsdl:input message="impl:upcomingTransportRequest"
       name="upcomingTransportRequest"/>
      <wsdl:output message="impl:upcomingTransportResponse"
       name="upcomingTransportResponse"/>
    </wsdl:operation>
    <wsdl:operation name="requestProblemReport"
     parameterOrder="serviceId performer p0 p1">
      <wsdl:input message="impl:requestProblemReportRequest"
       name="requestProblemReportRequest"/>
      <wsdl:output message="impl:requestProblemReportResponse"
       name="requestProblemReportResponse"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="transportControlServiceSoapBinding"
   type="impl:TransportControlPortType">
    <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="upcomingTransport">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="upcomingTransportRequest">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="http://servicecomposition.org/namespaces/2003/08/fresco/
             services/Transport/rev04/resources/TransportControl"
          use="encoded"/>
      </wsdl:input>
      <wsdl:output name="upcomingTransportResponse">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="http://servicecomposition.org/namespaces/2003/08/fresco/
             services/Transport/rev04/resources/TransportControl"
          use="encoded"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="requestProblemReport">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="requestProblemReportRequest">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="http://servicecomposition.org/namespaces/2003/08/fresco/
             services/Transport/rev04/resources/TransportControl"
          use="encoded"/>
      </wsdl:input>
      <wsdl:output name="requestProblemReportResponse">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="http://servicecomposition.org/namespaces/2003/08/fresco/
             services/Transport/rev04/resources/TransportControl"
          use="encoded"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="TransportControlPortTypeService">
    <wsdl:port binding="impl:transportControlServiceSoapBinding"
     name="transportControlService">
      <wsdlsoap:address
       location="http://localhost:8090/axis/services/transportControlService"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

A.2.6. eService Shell XPDL

```
<?xml version="1.0" encoding="us-ascii"?>
<!-- Convention: Package Id = <service base name>_shell -->
<Package Id="Transport_shell" Name="Transport Service"
  xmlns="http://www.wfmc.org/2002/XPDL1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.wfmc.org/2002/XPDL1.0
  http://wfmc.org/standards/docs/TC-1025_schema_10_xpdl.xsd">
  <PackageHeader>
    <XPDLVersion>1.0</XPDLVersion>
    <Vendor>FreightMixer</Vendor>
    <Created>2006-10-25 22:25:50</Created>
    <Description>E-Service sample</Description>
    <Documentation>none</Documentation>
  </PackageHeader>
  <RedefinableHeader PublicationStatus="UNDER_TEST">
    <Version>rev08</Version>
  </RedefinableHeader>
  <ConformanceClass GraphConformance="FULL_BLOCKED"/>
  <ExternalPackages>
    <!-- the transport_context package has to
     be referenced indirectly by the flow packages -->
    <ExternalPackage href="./PreTransport_flow.xpdl"/>
    <ExternalPackage href="./Handover_flow.xpdl"/>
    <ExternalPackage href="./PostTransport_flow.xpdl"/>
    <ExternalPackage href="./HandoverProblem_flow.xpdl"/>
    <ExternalPackage href="./Transport_context.xpdl"/>
  </ExternalPackages>
  <!-- TYPES ================================== -->
  <TypeDeclarations>
    <TypeDeclaration Id="ServiceIdType">
      <ExternalReference namespace="http://www.w3.org/2001/XMLSchema"
        xref="string" location="http://www.w3.org/2001/XMLSchema"/>
    </TypeDeclaration>
    <TypeDeclaration Id="TransportIdType">
      <ExternalReference namespace="http://www.w3.org/2001/XMLSchema"
        xref="string" location="http://www.w3.org/2001/XMLSchema"/>
    </TypeDeclaration>
    <TypeDeclaration Id="TransportLegIdType">
      <ExternalReference namespace="http://www.w3.org/2001/XMLSchema" xref="int"
        location="http://www.w3.org/2001/XMLSchema"/>
    </TypeDeclaration>
    <TypeDeclaration Id="statusIdType">
      <ExternalReference namespace="http://www.w3.org/2001/XMLSchema"
        xref="string" location="http://www.w3.org/2001/XMLSchema"/>
    </TypeDeclaration>
    <TypeDeclaration Id="problemIdType">
      <ExternalReference namespace="http://www.w3.org/2001/XMLSchema"
        xref="string" location="http://www.w3.org/2001/XMLSchema"/>
    </TypeDeclaration>
    <TypeDeclaration Id="AcknowledgeType">
      <ExternalReference namespace="http://www.w3.org/2001/XMLSchema"
        xref="boolean" location="http://www.w3.org/2001/XMLSchema"/>
    </TypeDeclaration>
  </TypeDeclarations>
  <WorkflowProcesses>
    <!-- Start of top-level capabilities process -->
    <!-- The top-level process of the transport service
     shell defines capabilities as subflows -->
    <!-- Convention: WorkflowProcess Id =  Capabilities -->
    <WorkflowProcess Id="Capabilities" Name="Transport Service Capabilities"
      AccessLevel="PUBLIC">
```

```
  <ProcessHeader/>
  <Activities>
   <!-- Convention: Activity Id = <capablity base name> -->
   <Activity Id="TransportPreparation"
      Name="Proclamation of transport preparation capability">
      <Implementation>
      <!-- Convention: SubFlow Id = <capablity base name> -->
      <SubFlow Id="TransportPreparation"/>
      </Implementation>
      <Performer>TransportPreparationManager</Performer>
    </Activity>
    <Activity Id="TransportCoordination"
      Name="Proclamation of transport coordination capability">
      <Implementation>
        <SubFlow Id="TransportCoordination"/>
      </Implementation>
      <Performer>TransportCoordinationManager</Performer>
    </Activity>
    <Activity Id="TransportCompletion"
      Name="Proclamation of transport completion capability">
      <Implementation>
        <SubFlow Id="TransportCompletion"/>
      </Implementation>
      <Performer>TransportCompletionManager</Performer>
    </Activity>
  </Activities>
</WorkflowProcess>
<!-- Start of capability subflows -->
<!-- The second-level capability processes each define
 the flows of a capability as subflows -->
<!-- The flow processes are specified in seperate xpdl
 documents (linked as external packages) -->
<!-- Convention: WorkflowProcess Id = <capablity base name> -->
<WorkflowProcess Id="TransportPreparation" AccessLevel="PRIVATE">
  <ProcessHeader/>
  <Activities>
    <Activity Id="TransportPreparationFlow1">
      <Implementation>
        <!-- Convention: SubFlow Id = <flow base name> -->
        <SubFlow Id="PreTransportFlow" Execution="ASYNCHR"/>
      </Implementation>
    </Activity>
  </Activities>
</WorkflowProcess>
<WorkflowProcess Id="TransportCompletion" AccessLevel="PRIVATE">
  <ProcessHeader/>
  <Activities>
    <Activity Id="TransportCompletionFlow1">
      <Implementation>
        <SubFlow Id="PostTransportFlow" Execution="ASYNCHR"/>
      </Implementation>
    </Activity>
  </Activities>
</WorkflowProcess>
<WorkflowProcess Id="TransportCoordination" AccessLevel="PRIVATE">
  <ProcessHeader/>
  <Activities>
    <Activity Id="TransportCoordinationFlow1">
      <Implementation>
        <SubFlow Id="HandoverFlow" Execution="ASYNCHR"/>
      </Implementation>
    </Activity>
    <Activity Id="TransportCoordinationFlow2">
```

```
          <Implementation>
            <SubFlow Id="HandoverProblemFlow" Execution="ASYNCHR"/>
          </Implementation>
        </Activity>
      </Activities>
    </WorkflowProcess>
  </WorkflowProcesses>
  <!-- TARGETNAMESPACE  ==================================== -->
  <ExtendedAttributes>
    <ExtendedAttribute Name="fresco.targetNamespace"
      Value="http://.../fresco/services/Transport/rev08"/>
  </ExtendedAttributes>
</Package>
```

A.2.7. eService Capability Interaction Flow XPDL

```
<?xml version="1.0" encoding="us-ascii"?>
<Package Id="Handover_flow" Name="Transport Handover Flow"
  xmlns="http://www.wfmc.org/2002/XPDL1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.wfmc.org/2002/XPDL1.0
  http://wfmc.org/standards/docs/TC-1025_schema_10_xpdl.xsd">
  <PackageHeader>
    <XPDLVersion>1.0</XPDLVersion>
    <Vendor>FreightMixer</Vendor>
    <Created>2006-10-19 22:25:50</Created>
    <Description>Process schema of handover interaction flow</Description>
  </PackageHeader>
  <RedefinableHeader PublicationStatus="UNDER_TEST">
    <Version>rev08</Version>
  </RedefinableHeader>
  <ConformanceClass GraphConformance="NON_BLOCKED"/>
  <ExternalPackages>
    <ExternalPackage href="./Transport_context.xpdl"/>
  </ExternalPackages>
  <!-- TYPES ================================== -->
  <TypeDeclarations>
    <TypeDeclaration Id="ServiceIdType">
      <ExternalReference namespace="http://www.w3.org/2001/XMLSchema"
        xref="string" location="http://www.w3.org/2001/XMLSchema"/>
    </TypeDeclaration>
    <TypeDeclaration Id="TransportIdType">
      <ExternalReference namespace="http://www.w3.org/2001/XMLSchema"
        xref="string" location="http://www.w3.org/2001/XMLSchema"/>
    </TypeDeclaration>
    <TypeDeclaration Id="TransportLegIdType">
      <ExternalReference namespace="http://www.w3.org/2001/XMLSchema" xref="int"
        location="http://www.w3.org/2001/XMLSchema"/>
    </TypeDeclaration>
    <TypeDeclaration Id="statusIdType">
      <ExternalReference namespace="http://www.w3.org/2001/XMLSchema"
        xref="string" location="http://www.w3.org/2001/XMLSchema"/>
    </TypeDeclaration>
    <TypeDeclaration Id="problemIdType">
      <ExternalReference namespace="http://www.w3.org/2001/XMLSchema"
        xref="string" location="http://www.w3.org/2001/XMLSchema"/>
    </TypeDeclaration>
    <TypeDeclaration Id="AcknowledgeType">
      <ExternalReference namespace="http://www.w3.org/2001/XMLSchema"
        xref="boolean" location="http://www.w3.org/2001/XMLSchema"/>
    </TypeDeclaration>
    <TypeDeclaration Id="BOOLEAN">
```

```
    <ExternalReference namespace="http://www.w3.org/2001/XMLSchema"
      xref="boolean" location="http://www.w3.org/2001/XMLSchema"/>
  </TypeDeclaration>
  <TypeDeclaration Id="STRING">
    <ExternalReference namespace="http://www.w3.org/2001/XMLSchema"
      xref="string" location="http://www.w3.org/2001/XMLSchema"/>
  </TypeDeclaration>
</TypeDeclarations>
<!-- INBOUND INTERFACE ================================== -->
<Applications>
  <Application Id="HandoverFlow_statusReport">
    <ExternalReference xref="HandoverFlowPortType_-_statusReport"
      location="./flows/HandoverFlow.wsdl"/>
  </Application>
  <Application Id="HandoverFlow_start">
    <ExternalReference xref="HandoverFlowPortType_-_start"
       location="./flows/HandoverFlow.wsdl"/>
  </Application>
</Applications>
<WorkflowProcesses>
  <!-- FLOW ================================== -->
  <WorkflowProcess Id="HandoverFlow" Name="HandoverFlow" AccessLevel="PUBLIC">
    <ProcessHeader/>
    <DataFields>
      <!-- VARIABLES ================================== -->
      <DataField Id="serviceId" Name="ServiceId" IsArray="FALSE">
        <DataType>
          <DeclaredType Id="ServiceIdType"/>
        </DataType>
        <Description>
          Unique service id, serving as service's correlation parameter
        </Description>
      </DataField>
      <DataField Id="transportId" Name="transportId" IsArray="FALSE">
        <DataType>
          <DeclaredType Id="TransportIdType"/>
        </DataType>
      </DataField>
      <DataField Id="acknowledge" Name="acknowledge" IsArray="FALSE">
        <DataType>
          <DeclaredType Id="AcknowledgeType"/>
        </DataType>
      </DataField>
      <DataField Id="transportLegId" Name="transportLegId" IsArray="FALSE">
        <DataType>
          <DeclaredType Id="TransportLegIdType"/>
        </DataType>
      </DataField>
      <DataField Id="nextTransportLegId" Name="transportLegId" IsArray="FALSE">
        <DataType>
          <DeclaredType Id="TransportLegIdType"/>
        </DataType>
      </DataField>
      <DataField Id="statusId" Name="statusId" IsArray="FALSE">
        <DataType>
          <DeclaredType Id="statusIdType"/>
        </DataType>
      </DataField>
      <!-- CONSTANTS ================================== -->
      <DataField Id="positive_acknowledge_constant" IsArray="FALSE">
        <DataType>
          <DeclaredType Id="BOOLEAN"/>
        </DataType>
```

```
      <InitialValue>true</InitialValue>
    </DataField>
    <!-- BINDINGS ======================================== -->
    <DataField Id="transport_coordination_manager_performer_constant"
      IsArray="FALSE">
      <DataType>
        <DeclaredType Id="STRING"/>
      </DataType>
      <InitialValue>TransportCoordinationManager(0)</InitialValue>
    </DataField>
    <DataField Id="transport_completion_manager_performer_constant"
      IsArray="FALSE">
      <DataType>
        <DeclaredType Id="STRING"/>
      </DataType>
      <InitialValue>TransportCompletionManager(0)</InitialValue>
    </DataField>
    <DataField Id="transport_customer_performer_constant" IsArray="FALSE">
      <DataType>
        <DeclaredType Id="STRING"/>
      </DataType>
      <InitialValue>Customer(0)</InitialValue>
    </DataField>
    <DataField Id="transport_insurer_performer_constant" IsArray="FALSE">
      <DataType>
        <DeclaredType Id="STRING"/>
      </DataType>
      <InitialValue>Insurer(0)</InitialValue>
    </DataField>
    <DataField Id="next_transport_provider_performer_constant"
      IsArray="FALSE">
      <DataType>
        <DeclaredType Id="STRING"/>
      </DataType>
      <InitialValue>TransportProvider(0,4)</InitialValue>
    </DataField>
    <DataField Id="transport_provider_performer_constant" IsArray="FALSE">
      <DataType>
        <DeclaredType Id="STRING"/>
      </DataType>
      <InitialValue>TransportProvider(0,4)</InitialValue>
    </DataField>
    <DataField Id="transport_completion_manager_performer_constant2"
      IsArray="FALSE">
      <DataType>
        <DeclaredType Id="STRING"/>
      </DataType>
      <InitialValue>TransportCompletionManager(0,4)</InitialValue>
    </DataField>
  </DataFields>
  <!-- WS AGGREGATOR / E-Service COORDINATOR ============= -->
  <Participants>
    <Participant Id="HandoverCoordinator">
      <ParticipantType Type="SYSTEM"/>
      <Description>
        process instance for each service instance and leg
      </Description>
      <ExtendedAttributes>
        <ExtendedAttribute name="fresco.correlationSet"
          Value="ServiceIdType,TransportLegIdType"/>
      </ExtendedAttributes>
    </Participant>
  </Participants>
```

```
<!-- INTERACTIONS ==================================== -->
<Activities>
  <Activity Id="start_handover" Name="Start handover">
    <Description>Externally invoked start activity</Description>
    <Implementation>
      <Tool Id="HandoverFlow_start" Type="PROCEDURE">
        <ActualParameters>
          <!-- internal in params -->
          <ActualParameter>in:serviceId</ActualParameter>
          <!-- resource in params -->
          <ActualParameter>in:transportId</ActualParameter>
          <ActualParameter>in:transportLegId</ActualParameter>
          <ActualParameter>out:positive_acknowledge_constant</ActualParameter>
        </ActualParameters>
        <ExtendedAttributes>
          <ExtendedAttribute Name="fresco.communication" Value="in"/>
        </ExtendedAttributes>
      </Tool>
    </Implementation>
    <Performer>HandoverCoordinator(0,2)</Performer>
  </Activity>
  <Activity Id="await_status_report" Name="Await status report">
    <Description>Coordinator waits for transport report</Description>
    <Implementation>
      <Tool Id="HandoverFlow_statusReport" Type="PROCEDURE">
        <ActualParameters>
          <ActualParameter>in:serviceId</ActualParameter>
          <ActualParameter>in:transportId</ActualParameter>
          <ActualParameter>in:transportLegId</ActualParameter>
          <ActualParameter>in:statusId</ActualParameter>
          <ActualParameter>out:positive_acknowledge_constant</ActualParameter>
        </ActualParameters>
        <ExtendedAttributes>
          <ExtendedAttribute Name="fresco.communication" Value="in"/>
        </ExtendedAttributes>
      </Tool>
    </Implementation>
    <Performer>HandoverCoordinator(0,2)</Performer>
  </Activity>
  <Activity Id="start_handover_problem_flow"
    Name="Start handover problem flow">
    <Description>
      Inter-capability transition from handover to handover-problem
    </Description>
    <Implementation>
      <Tool Id="TransportCoordination_handleProblem" Type="PROCEDURE">
        <ActualParameters>
          <ActualParameter>in:serviceId</ActualParameter>
          <ActualParameter>in:transport_coordination_manager_performer_constant
          </ActualParameter>
          <ActualParameter>in:transportId</ActualParameter>
          <ActualParameter>in:transportLegId</ActualParameter>
          <ActualParameter>out:acknowledge</ActualParameter>
        </ActualParameters>
        <ExtendedAttributes>
          <ExtendedAttribute Name="fresco.communication" Value="out"/>
        </ExtendedAttributes>
      </Tool>
    </Implementation>
    <Performer>TransportCoordinationManager(0)</Performer>
  </Activity>
  <Activity Id="request_problem_report" Name="request_problem_report">
    <Description>
```

```
    Ask the transport provider about a problem report
  </Description>
  <Implementation>
    <Tool Id="TransportControl_requestProblemReport" Type="PROCEDURE">
      <ActualParameters>
        <ActualParameter>in:serviceId</ActualParameter>
        <ActualParameter>in:transport_provider_performer_constant
        </ActualParameter>
        <ActualParameter>in:transportId</ActualParameter>
        <ActualParameter>in:nextTransportLegId</ActualParameter>
        <ActualParameter>out:acknowledge</ActualParameter>
      </ActualParameters>
      <ExtendedAttributes>
        <ExtendedAttribute Name="fresco.communication" Value="out"/>
      </ExtendedAttributes>
    </Tool>
  </Implementation>
  <Performer>TransportProvider(0,3)</Performer>
</Activity>
<Activity Id="notify_insurer" Name="notify_insurer">
  <Description>Notify insurer about transport progress</Description>
  <Implementation>
    <Tool Id="InsuranceControl_reportStatus" Type="PROCEDURE">
      <ActualParameters>
        <ActualParameter>in:serviceId</ActualParameter>
        <ActualParameter>in:transport_insurer_performer_constant
        </ActualParameter>
        <ActualParameter>in:transportId</ActualParameter>
        <ActualParameter>in:transportLegId</ActualParameter>
        <ActualParameter>in:statusId</ActualParameter>
        <ActualParameter>out:acknowledge</ActualParameter>
      </ActualParameters>
      <ExtendedAttributes>
        <ExtendedAttribute Name="fresco.communication" Value="out"/>
      </ExtendedAttributes>
    </Tool>
  </Implementation>
  <Performer>Insurer(0)</Performer>
</Activity>
<Activity Id="notify_customer" Name="notify_customer">
  <Description>Notify customer about transport progress</Description>
  <Implementation>
    <Tool Id="TransportMonitoring_completedTransport" Type="PROCEDURE">
      <ActualParameters>
        <ActualParameter>in:serviceId</ActualParameter>
        <ActualParameter>in:transport_customer_performer_constant
        </ActualParameter>
        <ActualParameter>in:transportId</ActualParameter>
        <ActualParameter>in:transportLegId</ActualParameter>
        <ActualParameter>in:statusId</ActualParameter>
        <ActualParameter>out:acknowledge</ActualParameter>
      </ActualParameters>
      <ExtendedAttributes>
        <ExtendedAttribute Name="fresco.communication" Value="out"/>
      </ExtendedAttributes>
    </Tool>
  </Implementation>
  <Performer>Customer(0)</Performer>
</Activity>
<Activity Id="get_next_leg" Name="Get next leg">
  <Description>Logistic computes next transport leg</Description>
  <Implementation>
    <Tool Id="LogisticManagement_computeNextLeg" Type="PROCEDURE">
```

```
      <ActualParameters>
        <ActualParameter>in:serviceId</ActualParameter>
        <ActualParameter>in:transport_coordination_manager_performer_constant
        </ActualParameter>
        <ActualParameter>in:transportId</ActualParameter>
        <ActualParameter>in:transportLegId</ActualParameter>
        <ActualParameter>out:nextTransportLegId</ActualParameter>
      </ActualParameters>
      <ExtendedAttributes>
        <ExtendedAttribute Name="fresco.communication" Value="out"/>
      </ExtendedAttributes>
    </Tool>
  </Implementation>
  <Performer>LogisticManager(0)</Performer>
</Activity>
<Activity Id="start_next_handover_flow" Name="start handover flow">
  <Description>
    Inter-capability transition from handover to next handover
  </Description>
  <Implementation>
    <Tool Id="TransportCoordination_start" Type="PROCEDURE">
      <ActualParameters>
        <ActualParameter>in:serviceId</ActualParameter>
        <ActualParameter>in:transport_coordination_manager_performer_constant
        </ActualParameter>
        <ActualParameter>in:transportId</ActualParameter>
        <ActualParameter>in:transportLegId</ActualParameter>
        <ActualParameter>out:acknowledge</ActualParameter>
      </ActualParameters>
      <ExtendedAttributes>
        <ExtendedAttribute Name="fresco.communication" Value="out"/>
      </ExtendedAttributes>
    </Tool>
  </Implementation>
  <Performer>TransportCoordinationManager(0)</Performer>
</Activity>
<Activity Id="notify_next_transport_provider"
  Name="Notify next transport provider">
  <Description>
    Notify the next transport provider about the start of his leg
  </Description>
  <Implementation>
    <Tool Id="TransportControl_upcomingTransport" Type="PROCEDURE">
      <ActualParameters>
        <ActualParameter>in:serviceId</ActualParameter>
        <ActualParameter>in:next_transport_provider_performer_constant
        </ActualParameter>
        <ActualParameter>in:transportId</ActualParameter>
        <ActualParameter>in:nextTransportLegId</ActualParameter>
        <ActualParameter>out:acknowledge</ActualParameter>
      </ActualParameters>
      <ExtendedAttributes>
        <ExtendedAttribute Name="fresco.communication" Value="out"/>
      </ExtendedAttributes>
    </Tool>
  </Implementation>
  <Performer>TransportProvider(0,3)</Performer>
</Activity>
<Activity Id="start_post_transport_flow"
  Name="start post transport flow">
  <Description>
    Inter-capability transition from handover to handover-problem
  </Description>
```

```
        <Implementation>
          <Tool Id="TransportCompletion_finishTransport" Type="PROCEDURE">
            <ActualParameters>
              <ActualParameter>in:serviceId</ActualParameter>
              <ActualParameter>in:transport_completion_manager_performer_constant
              </ActualParameter>
              <ActualParameter>in:transportId</ActualParameter>
              <ActualParameter>out:acknowledge</ActualParameter>
            </ActualParameters>
            <ExtendedAttributes>
              <ExtendedAttribute Name="fresco.communication" Value="out"/>
            </ExtendedAttributes>
          </Tool>
        </Implementation>
        <Performer>TransportCompletionManager(0)</Performer>
      </Activity>
      <Activity Id="replace_notify_insurer" Name="replace notify_insurer">
        <Description>Statische Aktivitaet nach Verzweigung</Description>
        <Route/>
        <Performer>Insurer</Performer>
      </Activity>
    </Activities>
    <Transitions>
      <Transition Id="t1" Name="t1"
        From="start_handover"
        to="await_status_report"/>
      <Transition Id="t2" name="t2"
        From="await_status_report"
        to="start_handover_problem_flow">
        <Condition Type="OTHERWISE"/>
      </Transition>
      <Transition Id="t3" Name="t3"
        From="await_status_report"
        To="notify_customer">
        <Condition Type="CONDITION">statusId="okay"</Condition>
      </Transition>
      <Transition Id="t4" Name="t4"
        From="notify_customer"
        To="replace_notify_insurer"/>
      <Transition Id="t5" Name="t5"
        From="replace_notify_insurer"
        To="get_next_leg"/>
      <Transition Id="t6" name="t6"
        From="get_next_leg" Name="t4"
        To="start_next_handover_flow">
        <Condition type="CONDITION">nextTransportLegId > 0</Condition>
      </Transition>
      <Transition Id="t7" Name="t7"
        From="get_next_leg"
        To="start_post_transport_flow">
        <Condition Type="OTHERWISE"/>
      </Transition>
      <Transition Id="t8" Name="t8"
        From="start_next_handover_flow"
        To="notify_next_transport_provider"/>
      <Transition Id="t9" Name="t9"
        From="start_handover_problem_flow"
        To="request_problem_report"/>
    </Transitions>
  </WorkflowProcess>
</WorkflowProcesses>
<!-- TARGETNAMESPACE ==================================== -->
<ExtendedAttributes>
```

```
    <ExtendedAttribute Name="fresco.targetNamespace"
    Value="http://.../fresco/services/Transport/rev08"/>
  </ExtendedAttributes>
</Package>
```

A.2.8. eService Capability Interaction Flow BPEL

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<process targetNamespace="http://.../services/Transport/rev08/bpel/HandoverFlow"
  name="HandoverFlow" xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
  xmlns:resource1="http://.../services/Transport/rev08/flows/HandoverFlow"
  xmlns:resource2="http://.../services/Transport/rev08/resources/TransportCoordination"
  xmlns:resource3="http://.../services/Transport/rev08/resources/TransportControl"
  xmlns:resource4="http://.../services/Transport/rev08/resources/InsuranceControl"
  xmlns:resource5="http://.../services/Transport/rev08/resources/TransportMonitoring"
  xmlns:resource6="http://.../services/Transport/rev08/resources/LogisticManagement"
  xmlns:resource7="http://.../services/Transport/rev08/resources/TransportCompletion"
  xmlns:tns="http://.../services/Transport/rev08/bpel/HandoverFlow"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <partners>
    <partner name="HandoverFlow" serviceLinkType="resource1:HandoverFlowPLT"
      myRole="HandoverFlowRole"/>
    <partner name="TransportCoordinationManager"
      serviceLinkType="resource2:TransportCoordinationPLT"
      partnerRole="TransportCoordinationRole"/>
    <partner name="TransportProvider"
      serviceLinkType="resource3:TransportControlPLT"
      partnerRole="TransportControlRole"/>
    <partner name="Insurer" serviceLinkType="resource4:InsuranceControlPLT"
      partnerRole="InsuranceControlRole"/>
    <partner name="Customer" serviceLinkType="resource5:TransportMonitoringPLT"
      partnerRole="TransportMonitoringRole"/>
    <partner name="LogisticManager"
      serviceLinkType="resource6:LogisticManagementPLT"
      partnerRole="LogisticManagementRole"/>
    <partner name="TransportCompletionManager"
      serviceLinkType="resource7:TransportCompletionPLT"
      partnerRole="TransportCompletionRole"/>
  </partners>
  <variables>
    <variable name="serviceId" messageType="resource1:_stringMessage"/>
    <variable name="transportId" messageType="resource1:_stringMessage"/>
    <variable name="acknowledge" messageType="resource1:_booleanMessage"/>
    <variable name="transportLegId" messageType="resource1:_intMessage"/>
    <variable name="nextTransportLegId" messageType="resource1:_intMessage"/>
    <variable name="statusId" messageType="resource1:_stringMessage"/>
    <variable name="positive_acknowledge_constant"
      messageType="resource1:_booleanMessage"/>
    <variable name="transport_coordination_manager_performer_constant"
      messageType="resource1:_stringMessage"/>
    <variable name="transport_completion_manager_performer_constant"
      messageType="resource1:_stringMessage"/>
    <variable name="transport_customer_performer_constant"
      messageType="resource1:_stringMessage"/>
    <variable name="transport_insurer_performer_constant"
      messageType="resource1:_stringMessage"/>
    <variable name="next_transport_provider_performer_constant"
      messageType="resource1:_stringMessage"/>
    <variable name="transport_provider_performer_constant"
      messageType="resource1:_stringMessage"/>
    <variable name="transport_completion_manager_performer_constant2"
      messageType="resource1:_stringMessage"/>
```

```
  <variable name="message1_in" messageType="resource1:startInputMessage"/>
  <variable name="message1_out" messageType="resource1:startOutputMessage"/>
  <variable name="message2_in"
    messageType="resource1:statusReportInputMessage"/>
  <variable name="message2_out"
    messageType="resource1:statusReportOutputMessage"/>
  <variable name="message3_in" messageType="resource2:handleProblemRequest"/>
  <variable name="message3_out"
    messageType="resource2:handleProblemResponse"/>
  <variable name="message4_in"
    messageType="resource3:requestProblemReportRequest"/>
  <variable name="message4_out"
    messageType="resource3:requestProblemReportResponse"/>
  <variable name="message5_in" messageType="resource4:reportStatusRequest"/>
  <variable name="message5_out" messageType="resource4:reportStatusResponse"/>
  <variable name="message6_in"
    messageType="resource5:completedTransportRequest"/>
  <variable name="message6_out"
    messageType="resource5:completedTransportResponse"/>
  <variable name="message7_in" messageType="resource6:computeNextLegRequest"/>
  <variable name="message7_out"
    messageType="resource6:computeNextLegResponse"/>
  <variable name="message8_in" messageType="resource2:startRequest"/>
  <variable name="message8_out" messageType="resource2:startResponse"/>
  <variable name="message9_in"
    messageType="resource3:upcomingTransportRequest"/>
  <variable name="message9_out"
    messageType="resource3:upcomingTransportResponse"/>
  <variable name="message10_in"
    messageType="resource7:finishTransportRequest"/>
  <variable name="message10_out"
    messageType="resource7:finishTransportResponse"/>
</variables>
<correlationSets>
  <correlationSet name="HandoverCoordinator" properties="resource1:ServiceIdType
  resource1:TransportLegIdType"/>
</correlationSets>
<flow>
  <links>
    <link name="t1"/>
    <link name="t6"/>
    <link name="t7"/>
    <link name="t8"/>
    <link name="t2"/>
    <link name="t9"/>
    <link name="t3"/>
    <link name="t4"/>
    <link name="t5"/>
  </links>
  <sequence name="start_handover">
    <source linkName="t1"/>
    <receive partner="HandoverFlow" portType="resource1:HandoverFlowPortType"
      operation="start" variable="message1_in" createInstance="yes">
      <correlations>
        <correlation set="HandoverCoordinator"
          initiate="yes"/>
      </correlations>
    </receive>
    <assign>
      <copy>
        <from expression="TransportCompletionManager(0,4)"/>
        <to variable="transport_completion_manager_performer_constant2"
          part="value"/>
```

```
      </copy>
      <copy>
        <from expression="TransportProvider(0,4)"/>
        <to variable="transport_provider_performer_constant"
          part="value"/>
      </copy>
      <copy>
        <from expression="TransportCoordinationManager(0)"/>
        <to variable="transport_coordination_manager_performer_constant"
          part="value"/>
      </copy>
      <copy>
        <from expression="Insurer(0)"/>
        <to variable="transport_insurer_performer_constant"
          part="value"/>
      </copy>
      <copy>
        <from expression="Customer(0)"/>
        <to variable="transport_customer_performer_constant"
          part="value"/>
      </copy>
      <copy>
        <from expression="true"/>
        <to variable="positive_acknowledge_constant"
          part="value"/>
      </copy>
      <copy>
        <from expression="TransportProvider(0,4)"/>
        <to variable="next_transport_provider_performer_constant"
          part="value"/>
      </copy>
      <copy>
        <from expression="TransportCompletionManager(0)"/>
        <to variable="transport_completion_manager_performer_constant"
          part="value"/>
      </copy>
    </assign>
    <assign>
      <copy>
        <from variable="message1_in" part="serviceId"/>
        <to variable="serviceId" part="value"/>
      </copy>
      <copy>
        <from variable="message1_in" part="transportId"/>
        <to variable="transportId" part="value"/>
      </copy>
      <copy>
        <from variable="message1_in" part="transportLegId"/>
        <to variable="transportLegId" part="value"/>
      </copy>
    </assign>
    <assign>
      <copy>
        <from variable="positive_acknowledge_constant"
          part="value"/>
        <to variable="message1_out" part="acknowledge"/>
      </copy>
    </assign>
    <reply partner="HandoverFlow" portType="resource1:HandoverFlowPortType"
      operation="start" variable="message1_out"/>
  </sequence>
  <sequence name="await_status_report">
    <target linkName="t1"/>
```

```
    <source linkName="t2" transitionCondition=""/>
    <source linkName="t3" transitionCondition="statusId=&quot;okay&quot;"/>
    <receive partner="HandoverFlow" portType="resource1:HandoverFlowPortType"
      operation="statusReport" variable="message2_in"
      createInstance="no">
      <correlations>
        <correlation set="HandoverCoordinator"
          initiate="yes"/>
      </correlations>
    </receive>
    <assign>
      <copy>
        <from variable="message2_in" part="serviceId"/>
        <to variable="serviceId" part="value"/>
      </copy>
      <copy>
        <from variable="message2_in" part="transportId"/>
        <to variable="transportId" part="value"/>
      </copy>
      <copy>
        <from variable="message2_in" part="transportLegId"/>
        <to variable="transportLegId" part="value"/>
      </copy>
      <copy>
        <from variable="message2_in" part="statusId"/>
        <to variable="statusId" part="value"/>
      </copy>
    </assign>
    <assign>
      <copy>
        <from variable="positive_acknowledge_constant"
          part="value"/>
        <to variable="message2_out" part="acknowledge"/>
      </copy>
    </assign>
    <reply partner="HandoverFlow" portType="resource1:HandoverFlowPortType"
      operation="statusReport" variable="message2_out"/>
  </sequence>
  <sequence name="start_handover_problem_flow">
    <target linkName="t2"/>
    <source linkName="t9"/>
    <assign>
      <copy>
        <from variable="serviceId" part="value"/>
        <to variable="message3_in" part="serviceId"/>
      </copy>
      <copy>
        <from variable="transport_coordination_manager_performer_constant"
          part="value"/>
        <to variable="message3_in" part="performer"/>
      </copy>
      <copy>
        <from variable="transportId" part="value"/>
        <to variable="message3_in" part="p0"/>
      </copy>
      <copy>
        <from variable="transportLegId" part="value"/>
        <to variable="message3_in" part="p1"/>
      </copy>
    </assign>
    <invoke partner="TransportCoordinationManager"
      portType="resource2:TransportCoordinationPortType"
      operation="handleProblem" inputVariable="message3_in"
```

```
      outputVariable="message3_out"/>
    <assign>
      <copy>
        <from variable="message3_out" part="handleProblemReturn"/>
        <to variable="acknowledge" part="value"/>
      </copy>
    </assign>
  </sequence>
  <sequence name="request_problem_report">
    <target linkName="t9"/>
    <assign>
      <copy>
        <from variable="serviceId" part="value"/>
        <to variable="message4_in" part="serviceId"/>
      </copy>
      <copy>
        <from variable="transport_provider_performer_constant"
          part="value"/>
        <to variable="message4_in" part="performer"/>
      </copy>
      <copy>
        <from variable="transportId" part="value"/>
        <to variable="message4_in" part="p0"/>
      </copy>
      <copy>
        <from variable="nextTransportLegId"
          part="value"/>
        <to variable="message4_in" part="p1"/>
      </copy>
    </assign>
    <invoke partner="TransportProvider"
      portType="resource3:TransportControlPortType"
      operation="requestProblemReport" inputVariable="message4_in"
      outputVariable="message4_out"/>
    <assign>
      <copy>
        <from variable="message4_out" part="requestProblemReportReturn"/>
        <to variable="acknowledge" part="value"/>
      </copy>
    </assign>
  </sequence>
  <sequence name="notify_insurer">
    <assign>
      <copy>
        <from variable="serviceId" part="value"/>
        <to variable="message5_in" part="serviceId"/>
      </copy>
      <copy>
        <from variable="transport_insurer_performer_constant"
          part="value"/>
        <to variable="message5_in" part="performer"/>
      </copy>
      <copy>
        <from variable="transportId" part="value"/>
        <to variable="message5_in" part="p0"/>
      </copy>
      <copy>
        <from variable="transportLegId" part="value"/>
        <to variable="message5_in" part="p1"/>
      </copy>
      <copy>
        <from variable="statusId" part="value"/>
        <to variable="message5_in" part="p2"/>
```

```
      </copy>
    </assign>
    <invoke partner="Insurer" portType="resource4:InsuranceControlPortType"
      operation="reportStatus" inputVariable="message5_in"
      outputVariable="message5_out"/>
    <assign>
      <copy>
        <from variable="message5_out" part="reportStatusReturn"/>
        <to variable="acknowledge" part="value"/>
      </copy>
    </assign>
  </sequence>
  <sequence name="notify_customer">
    <target linkName="t3"/>
    <source linkName="t4"/>
    <assign>
      <copy>
        <from variable="serviceId" part="value"/>
        <to variable="message6_in" part="serviceId"/>
      </copy>
      <copy>
        <from variable="transport_customer_performer_constant"
          part="value"/>
        <to variable="message6_in" part="performer"/>
      </copy>
      <copy>
        <from variable="transportId" part="value"/>
        <to variable="message6_in" part="p0"/>
      </copy>
      <copy>
        <from variable="transportLegId" part="value"/>
        <to variable="message6_in" part="p1"/>
      </copy>
      <copy>
        <from variable="statusId" part="value"/>
        <to variable="message6_in" part="p2"/>
      </copy>
    </assign>
    <invoke partner="Customer"
      portType="resource5:TransportMonitoringPortType"
      operation="completedTransport" inputVariable="message6_in"
      outputVariable="message6_out"/>
    <assign>
      <copy>
        <from variable="message6_out" part="completedTransportReturn"/>
        <to variable="acknowledge" part="value"/>
      </copy>
    </assign>
  </sequence>
  <sequence name="get_next_leg">
    <target linkName="t5"/>
    <source linkName="t6" transitionCondition="nextTransportLegId > 0"/>
    <source linkName="t7" transitionCondition=""/>
    <assign>
      <copy>
        <from variable="serviceId" part="value"/>
        <to variable="message7_in" part="serviceId"/>
      </copy>
      <copy>
        <from variable="transport_coordination_manager_performer_constant"
          part="value"/>
        <to variable="message7_in" part="performer"/>
      </copy>
```

```
      <copy>
        <from variable="transportId" part="value"/>
        <to variable="message7_in" part="p0"/>
      </copy>
      <copy>
        <from variable="transportLegId" part="value"/>
        <to variable="message7_in" part="p1"/>
      </copy>
    </assign>
    <invoke partner="LogisticManager"
      portType="resource6:LogisticManagementPortType"
      operation="computeNextLeg" inputVariable="message7_in"
      outputVariable="message7_out"/>
    <assign>
      <copy>
        <from variable="message7_out" part="computeNextLegReturn"/>
        <to variable="nextTransportLegId" part="value"/>
      </copy>
    </assign>
  </sequence>
  <sequence name="start_next_handover_flow">
    <target linkName="t6"/>
    <source linkName="t8"/>
    <assign>
      <copy>
        <from variable="serviceId" part="value"/>
        <to variable="message8_in" part="serviceId"/>
      </copy>
      <copy>
        <from variable="transport_coordination_manager_performer_constant"
          part="value"/>
        <to variable="message8_in" part="performer"/>
      </copy>
      <copy>
        <from variable="transportId" part="value"/>
        <to variable="message8_in" part="p0"/>
      </copy>
      <copy>
        <from variable="transportLegId" part="value"/>
        <to variable="message8_in" part="p1"/>
      </copy>
    </assign>
    <invoke partner="TransportCoordinationManager"
      portType="resource2:TransportCoordinationPortType"
      operation="start" inputVariable="message8_in"
      outputVariable="message8_out"/>
    <assign>
      <copy>
        <from variable="message8_out" part="startReturn"/>
        <to variable="acknowledge" part="value"/>
      </copy>
    </assign>
  </sequence>
  <sequence name="notify_next_transport_provider">
    <target linkName="t8"/>
    <assign>
      <copy>
        <from variable="serviceId" part="value"/>
        <to variable="message9_in" part="serviceId"/>
      </copy>
      <copy>
        <from variable="next_transport_provider_performer_constant"
          part="value"/>
```

```
          <to variable="message9_in" part="performer"/>
        </copy>
        <copy>
          <from variable="transportId" part="value"/>
          <to variable="message9_in" part="p0"/>
        </copy>
        <copy>
          <from variable="nextTransportLegId"
            part="value"/>
          <to variable="message9_in" part="p1"/>
        </copy>
      </assign>
      <invoke partner="TransportProvider"
        portType="resource3:TransportControlPortType"
        operation="upcomingTransport" inputVariable="message9_in"
        outputVariable="message9_out"/>
      <assign>
        <copy>
          <from variable="message9_out" part="upcomingTransportReturn"/>
          <to variable="acknowledge" part="value"/>
        </copy>
      </assign>
    </sequence>
    <sequence name="start_post_transport_flow">
      <target linkName="t7"/>
      <assign>
        <copy>
          <from variable="serviceId" part="value"/>
          <to variable="message10_in" part="serviceId"/>
        </copy>
        <copy>
          <from variable="transport_completion_manager_performer_constant"
            part="value"/>
          <to variable="message10_in" part="performer"/>
        </copy>
        <copy>
          <from variable="transportId" part="value"/>
          <to variable="message10_in" part="p0"/>
        </copy>
      </assign>
      <invoke partner="TransportCompletionManager"
        portType="resource7:TransportCompletionPortType"
        operation="finishTransport" inputVariable="message10_in"
        outputVariable="message10_out"/>
      <assign>
        <copy>
          <from variable="message10_out" part="finishTransportReturn"/>
          <to variable="acknowledge" part="value"/>
        </copy>
      </assign>
    </sequence>
    <empty name="replace_notify_insurer">
      <target linkName="t4"/>
      <source linkName="t5"/>
    </empty>
  </flow>
</process>
```

A.2.9. eService Interaction Context XPDL

```
<?xml version="1.0" encoding="us-ascii"?>
<!-- Convention: Package Id = <service base name>_context -->
```

```
<Package Id="Transport_context" Name="Transport Service Context"
  xmlns="http://www.wfmc.org/2002/XPDL1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.wfmc.org/2002/XPDL1.0
  http://wfmc.org/standards/docs/TC-1025_schema_10_xpdl.xsd">
  <PackageHeader>
    <XPDLVersion>1.0</XPDLVersion>
   <Vendor>FreightMixer</Vendor>
    <Created>2006-10-25 22:25:50</Created>
    <Description>E-Service Interaction Context Sample</Description>
  </PackageHeader>
  <RedefinableHeader PublicationStatus="UNDER_TEST">
   <Version>rev08</Version>
  </RedefinableHeader>
  <ConformanceClass GraphConformance="FULL_BLOCKED"/>
  <!-- TYPES ======================================== -->
  <TypeDeclarations>
    <TypeDeclaration Id="ServiceIdType">
      <ExternalReference namespace="http://www.w3.org/2001/XMLSchema"
        xref="string" location="http://www.w3.org/2001/XMLSchema"/>
    </TypeDeclaration>
    <TypeDeclaration Id="TransportIdType">
      <ExternalReference namespace="http://www.w3.org/2001/XMLSchema"
        xref="string" location="http://www.w3.org/2001/XMLSchema"/>
    </TypeDeclaration>
    <TypeDeclaration Id="TransportLegIdType">
      <ExternalReference namespace="http://www.w3.org/2001/XMLSchema" xref="int"
        location="http://www.w3.org/2001/XMLSchema"/>
    </TypeDeclaration>
    <TypeDeclaration Id="statusIdType">
      <ExternalReference namespace="http://www.w3.org/2001/XMLSchema"
        xref="string" location="http://www.w3.org/2001/XMLSchema"/>
    </TypeDeclaration>
    <TypeDeclaration Id="problemIdType">
      <ExternalReference namespace="http://www.w3.org/2001/XMLSchema"
        xref="string" location="http://www.w3.org/2001/XMLSchema"/>
    </TypeDeclaration>
    <TypeDeclaration Id="AcknowledgeType">
      <ExternalReference namespace="http://www.w3.org/2001/XMLSchema"
        xref="boolean" location="http://www.w3.org/2001/XMLSchema"/>
    </TypeDeclaration>
  </TypeDeclarations>
  <!-- ROLES ======================================== -->
  <Participants>
    <Participant Id="Insurer" Name="Insurer">
      <ParticipantType Type="ROLE"/>
      <ExtendedAttributes>
        <ExtendedAttribute Name="fresco.correlationSet" Value="ServiceIdType"/>
      </ExtendedAttributes>
    </Participant>
    <Participant Id="WarehouseStart" Name="WarehouseStart">
      <ParticipantType Type="ROLE"/>
      <ExtendedAttributes>
        <ExtendedAttribute Name="fresco.correlationSet" Value="ServiceIdType"/>
      </ExtendedAttributes>
    </Participant>
    <Participant Id="TransportCompletionManager"
      Name="TransportCompletionManager">
     <ParticipantType Type="ROLE"/>
      <ExtendedAttributes>
        <ExtendedAttribute Name="fresco.correlationSet" Value="ServiceIdType"/>
      </ExtendedAttributes>
    </Participant>
```

```
  <Participant Id="TransportPreparationManager"
    Name="TransportPreparationManager">
    <ParticipantType Type="ROLE"/>
    <ExtendedAttributes>
      <ExtendedAttribute Name="fresco.correlationSet" Value="ServiceIdType"/>
    </ExtendedAttributes>
  </Participant>
  <Participant Id="WarehouseEnd">
    <ParticipantType Type="ROLE"/>
    <ExtendedAttributes>
      <ExtendedAttribute Name="fresco.correlationSet" Value="ServiceIdType"/>
    </ExtendedAttributes>
  </Participant>
  <Participant Id="TransportCoordinationManager"
    Name="TransportCoordinationManager">
   <ParticipantType Type="ROLE"/>
    <ExtendedAttributes>
      <ExtendedAttribute Name="fresco.correlationSet" Value="ServiceIdType"/>
    </ExtendedAttributes>
  </Participant>
  <Participant Id="Customer" Name="Customer">
    <ParticipantType Type="ROLE"/>
    <ExtendedAttributes>
      <ExtendedAttribute Name="fresco.correlationSet" Value="ServiceIdType"/>
    </ExtendedAttributes>
  </Participant>
  <Participant Id="TransportProvider" Name="TransportProvider">
    <ParticipantType Type="ROLE"/>
    <ExtendedAttributes>
      <ExtendedAttribute Name="fresco.correlationSet"
      Value="ServiceIdType, TransportLegIdType"/>
    </ExtendedAttributes>
  </Participant>
  <Participant Id="LogisticManager" Name="LogisticManager">
    <ParticipantType Type="ROLE"/>
    <ExtendedAttributes>
      <ExtendedAttribute Name="fresco.correlationSet" Value="ServiceIdType"/>
    </ExtendedAttributes>
  </Participant>
</Participants>
<!-- Contacts ================================== -->
<Applications>
  <!-- Convention: Application Id = <resource base name>_<operation name> -->
  <!-- used by: PreTransport_flow -->
  <Application Id="TransportMonitoring_problemInTransport">
    <ExternalReference xref="TransportMonitoringPortType_-_problemInTransport"
      location="../resources/transportMonitoring_port_type.gwsdl"/>
  <!-- Convention: xref = <portType>_-_<operation> ,
       <portType> = <resource base name>PortType -->
  </Application>
  <Application Id="TransportControl_requestProblemReport">
    <ExternalReference xref="TransportControlPortType_-_requestProblemReport"
      location="../resources/transportControl_port_type.gwsdl"/>
  </Application>
  <Application Id="WarehouseControl_departure">
    <ExternalReference xref="WarehouseControlPortType_-_departure"
      location="../resources/warehouseControl_port_type.gwsdl"/>
  </Application>
  <Application Id="TransportCoordination_awaitStatusReport">
    <ExternalReference xref="TransportCoordinationPortType_-_statusReport"
      location="capabilities/transportCoordination.gwsdl"/>
  </Application>
  <Application Id="InsuranceControl_registerTransport">
```

```
      <ExternalReference xref="InsuranceControlPortType_-_registerTransport"
        location="../resources/insuranceControl_port_type.gwsdl"/>
    </Application>
    <Application Id="TransportCoordination_handleProblem">
      <ExternalReference xref="TransportCoordinationPortType_-_handleProblem"
        location="capabilities/transportCoordination.gwsdl"/>
    </Application>
    <Application Id="LogisticManagement_computeNextLeg">
      <ExternalReference xref="LogisticManagementPortType_-_computeNextLeg"
        location="../resources/logisticManagement_port_type.gwsdl"/>
    </Application>
    <Application Id="TransportMonitoring_completedTransport">
      <ExternalReference xref="TransportMonitoringPortType_-_completedTransport"
        location="../resources/transportMonitoring_port_type.gwsdl"/>
    </Application>
    <Application Id="InsuranceControl_reportProblem">
      <ExternalReference xref="InsuranceControlPortType_-_reportProblem"
        location="../resources/insuranceControl_port_type.gwsdl"/>
    </Application>
    <Application Id="LogisticManagement_compensateProblem">
      <ExternalReference xref="LogisticManagementPortType_-_compensateProblem"
        location="../resources/logisticManagement_port_type.gwsdl"/>
    </Application>
    <Application Id="TransportPreparation_startTransport">
      <ExternalReference xref="TransportPreparationPortType_-_startTransport"
        location="./capabilities/transportPreparation.gwsdl"/>
    </Application>
    <Application Id="LogisticManagement_doAccounting">
      <ExternalReference xref="LogisticManagementPortType_-_doAccounting"
        location="../resources/logisticManagement_port_type.gwsdl"/>
    </Application>
    <Application Id="TransportCompletion_finishTransport">
      <ExternalReference xref="TransportCompletionPortType_-_finishTransport"
        location="./capabilities/transportCompletion.gwsdl"/>
    </Application>
    <Application Id="TransportCoordination_start">
      <ExternalReference xref="TransportCoordinationPortType_-_start"
        location="capabilities/transportCoordination.gwsdl"/>
    </Application>
    <Application Id="TransportControl_upcomingTransport">
      <ExternalReference xref="TransportControlPortType_-_upcomingTransport"
        location="../resources/transportControl_port_type.gwsdl"/>
    </Application>
    <Application Id="WarehouseControl_arrival">
      <ExternalReference xref="WarehouseControlPortType_-_arrival"
        location="../resources/warehouseControl_port_type.gwsdl"/>
    </Application>
    <Application Id="InsuranceControl_reportStatus">
      <ExternalReference xref="InsuranceControlPortType_-_reportStatus"
        location="../resources/insuranceControl_port_type.gwsdl"/>
    </Application>
    <Application Id="TransportCoordination_awaitProblemReport">
      <ExternalReference xref="TransportCoordinationPortType_-_problemReport"
        location="capabilities/transportCoordination.gwsdl"/>
    </Application>
  </Applications>
  <!-- TARGETNAMESPACE  ==================================== -->
  <ExtendedAttributes>
    <ExtendedAttribute Name="freco.targetNamespace"
      Value="http://.../fresco/services/Transport/rev08"/>
  </ExtendedAttributes>
</Package>
```

Literaturverzeichnis

[Aal99] Aalst, W. M. P. d.: Process-oriented architectures for electronic commerce and interorganizational workflow. In: *Information Systems* 24 (1999), Nr. 8, S. 639–71

[Aal03a] Aalst, W. M. P. d.: Don't go with the flow: Web services composition standards exposed. Web Services – Been there done that?, Trends & Controversies. In: *IEEE Intelligent Systems* 18 (2003), Nr. 1, S. 72–76

[Aal03b] Aalst, W. M. P. d.: Patterns and XPDL: A Critical Evaluation of the XML Process Definition Language / Queensland University of Technology. 2003 (QUT Technical report, FIT-TR-2003-06). – Forschungsbericht

[Aal04] Aalst, W. M. P. d.: Pi calculus versus Petri nets: Let us eat "humble pie" rather than further inflate the "Pi hype" / Department of Technology Management, Eindhoven University of Technology. 2004. – Discussion paper

[ABC+02] Atkinson, B. ; Bellwood, T. ; Cahuzac, M. u. a.: UDDI Version 3.0.1 / The Organization for the Advancement of Structured Information Standards (OASIS). 2002 (`http://uddi.org/pubs/uddi-v3.0.1-20031014.htm`). – UDDI Spec Technical Committee Specification

[ABE+01] Aalst, W. M. P. d. ; Batthelmess, P. ; Ellis, C.A. u. a.: Proclets: a framework for lightweight interacting workflow processes. In: *International Journal of Cooperative Information Systems* 10 (2001), Nr. 4, S. 443–81

[ABH+01] Ankolekar, Anupriya ; Burstein, Mark ; Hobbs, Jerry R. u. a.: DAML-S: Semantic Markup For Web Services. In: Cruz, I. F. (Hrsg.) ; Decker, S. (Hrsg.) ; Euzenat, J. (Hrsg.) u. a.: *Proceedings of SWWS'01, The first Semantic Web Working Symposium, Stanford University, California, USA, July 30 - August 1, 2001.* 2001, S. 411–430

[ACD+03] Andrews, T. ; Curbera, F. ; Dholakia, H. u. a.: *Business Process Execution Language for Web Services, Version 1.1, 5 May 2003.* BEA, IBM, Microsoft, SAP, Siebel, `ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf` (Zugriff am 1.8.2006), 2003

[ACG+03] Andrews, T. ; Curbera, F. ; Goland, Y. u. a.: *Business Process Execution Language for Web Services, Version 1.1, 20 March 2003*. BEA, IBM, Microsoft Corporation, `http://xml.coverpages.org/BPELv11.pdf` (Zugriff am 1.8.2006), 2003

[ACK+04] Alonso, G. ; Casati, F. ; Kuno, H. u. a.: *Web Services - Concepts, Architectures and Applications*. Springer, 2004

[ADH03] Aalst, W. M. P. d. ; Dumas, M. ; Hofstede, A.H.M. ter: Web Service Composition Languages: Old Wine in New Bottles? In: Chroust, G. (Hrsg.) ; Hofer, C. (Hrsg.): *Proceeding of the 29th EUROMICRO Conference: New Waves in System Architecture*. Los Alamitos, CA : IEEE Computer Society, 2003, S. 298–305

[AE03] Ahlert, D. ; Evanschitzky, H: *Dienstleistungsnetzwerke: Management, Erfolgsfaktoren und Benchmarks im internationalen Vergleich*. Berlin : Springer, 2003

[AFH+95] Arnold, O. ; Faisst, W. ; Härtling, M. u. a.: Virtuelle Unternehmen als Unternehmenstyp der Zukunft. In: *HMD, Theorie und Praxis der Wirtschaftsinformatik* 32 (1995), Nr. 185, S. 8–23

[AFL+99] Alonso, G. ; Fiedler, U. ; Lazcano, A. u. a.: WISE: An Infrastructure for E-Commerce. In: Dadam, P. (Hrsg.) ; Reichert, M. (Hrsg.): *Workshop Informatik '99: Enterprise-wide and Cross-enterprise Workflow Management: Concepts, Systems, Applications, Paderborn, Germany, October 6, 1999* Bd. 24. 1999

[AH02] Aalst, W. M. P. d. ; Hee, K. van: *Workflow Management: Models, Methods, and Systems*. Cambridge u.a. : MIT Press, 2002 (Cooperative Information Systems)

[AHK+03] Aalst, W. M. P. d. ; Hofstede, A.H.M. ter ; Kiepuszewski, B. u. a.: Workflow Patterns. In: *Distributed and Parallel Databases* 14 (2003), Nr. 3, S. 5–51

[AK05] Adam, H. ; Kirchheim, B.: *Entwicklung eines UML-basierten Workflowmodellierungstools zur Erfassung von Dienstleistungsprozessen*, University of Hamburg, MIN-Faculty, Department of Informatics, Studienarbeit (Bachelor Thesis), 2005

[Alp96] Alpar, P.: *Kommerzielle Nutzung des Internet*. Berlin Heidelberg u.a. : Springer, 1996

[AMI93] AMICE, ESPRIT C.: *CIMOSA - Open Systems Architecture for CIM*. 2. Berlin : Springer, 1993

[AMS02] Aissi, S. ; Malu, P. ; Srinivasan, K.: E-Business Process Modelling: The Next Big Step. In: *IEEE Computer* 35 (2002), Nr. 5, S. 55–62

[Apa06] Apache: *Velocity*. Apache Software Foundation, Jakarta Project, `http://xerces.apache.org/` (Zugriff am 1.9.2006), 2006

[Ark02] Arkin, A.: Web Services Choreography Interface 1.0 / World Wide Web Consortium. 2002 (`http://www.w3.org/TR/wsci/`). – W3C Note

[AW01] Aalst, W. M. P. d. ; Weske, M.: The P2P Approach to Interorganizational Workflows. In: Dittrich, K. R. (Hrsg.) ; Geppert, A. (Hrsg.) ; Norrie, M. C. (Hrsg.): *Advanced Information Systems Engineering, 13th International Conference, CAiSE 2001, Interlaken, Switzerland, June 4-8, 2001, Proceedings*. Berlin u.a. : Springer, 2001 (LNCS 2068), S. 140–156

[BBB+02] Banerji, A. ; Bartolini, C. ; Beringer, D. u. a.: Web Services Conversation Language (WSCL) 1.0 / World Wide Web Consortium. 2002 (`http://www.w3.org/TR/wscl10/`). – W3C Note

[BBG03] Baïna, K. ; Benali, K. ; Godart, C.: Dynamic Interconnection of Heterogeneous Workflow Processes through Services. In: Meersman, R. (Hrsg.) ; Tari, Z. (Hrsg.) ; Schmidt, D. C. (Hrsg.): *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE - OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003*. Springer, 2003 (LNCS 2888), S. 444–461

[BCC+04] Box, D. ; Christensen, E. ; Curbera, F. u. a.: Web Services Addressing (WS-Addressing) / World Wide Web Consortium. 2004 (`http://www.w3.org/Submission/2004/SUBM-ws-addressing-20040810/`). – W3C Member Submission

[BDF+01] Benatallah, B. ; Dumas, M. ; Fauvet, M.C. u. a.: Self-Coordinated and Self-Traced Composite Services with Dynamic Provider Selection / University of New South Wales, School of Computer Science and Engineering. 2001 (UNSW-CSE-TR-0108). – Forschungsbericht

[BDM02] Benatallah, B. ; Dumas, M. ; Maamar, Z.: Definition and Execution of Composite Web Services: The SELF-SERV Project. In: *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* 25 (2002), Nr. 4, S. 6

[Bet01] Bettag, U.: Web Services. In: *Informatik-Spektrum* 24 (2001), Nr. Oktober, S. 302–304

[BKK03a] Bernauer, M. ; Kappel, G. ; Kramler, G.: Comparing WSDL-Based and ebXML-Based Approaches for B2B Protocol Specification. In: Orlowska, M. (Hrsg.) ; Weerawarana, S. (Hrsg.) ; Papazoglou, M. P. (Hrsg.) u. a.: *Service-Oriented Computing - ICSOC 2003, First International Conference, Trento, Italy, December 15-18, 2003, Proceedings*. Springer, 2003 (LNCS 2910), S. 225–240

[BKK+03b] Bernauer, M. ; Kramler, G. ; Kappel, G. u. a.: Specification of Interorganizational Workflows - A Comparison of Approaches. In: *Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2003), 27-30 July, 2003, Orlando, USA*. 2003, S. 30–36

[BLHL01] Berners-Lee, T. ; Hendler, J. ; Lassila, O.: The Semantic Web. In: *Scientific American* 284 (2001), Nr. 5, S. 34–43

[BM98] Bruhn, M. (Hrsg.) ; Meffert, H. (Hrsg.): *Handbuch Dienstleistungsmanagement*. Wiesbaden : Gabler, 1998

[BMB+00] Benatallah, B. ; Medjahed, B. ; Bouguettaya, A. u. a.: Composing and Maintaining Web-based Virtual Enterprises. In: *First VLDB Workshop on Technologies for E-Services, Cairo, Egypt, September 2000*. 2000, S. 155–174

[BNS03] Bernus, P. (Hrsg.) ; Nemes, L. (Hrsg.) ; Schmidt, G. (Hrsg.): *Handbook on Enterprise Architecture*. Berlin u.a. : Springer, 2003 (International Handbooks on Information Systems)

[Bod99] Bodendorf, F.: *Wirtschaftsinformatik im Dienstleistungsbereich*. Berlin u.a. : Springer, 1999

[Bod02] Bodoff, S.: *The J2EE Tutorial*. Addison Wesley, 2002

[BP04] Brandt, H. ; Plümpe, T.: *Rule-Driven Adaptation of Workflow-Based E-Services*, University of Hamburg, MIN-Faculty, Department of Informatics, Diplomarbeit (Thesis), 2004

[BPSM+04] Bray, T. ; Paoli, J. ; Sperberg-McQueen, C. u. a.: Extensible Markup Language (XML) 1.1 / World Wide Web Consortium. 2004 (`http://www.w3.org/TR/2004/REC-xml11-20040204/`). – W3C Recommendation

[Brü99] Brütsch, D.: *Virtuelle Unternehmen.* Zürich : vdf Hochschulverlag an der ETH, 1999

[BSD03] Benatallah, B. ; Sheng, Q. Z. ; Dumas, M.: The SELF-SERV Environment for Web Services Composition. In: *IEEE Internet Computing* 7 (2003), Nr. 1, S. 40–48

[BTGN+98] Betreich-Teichmann, W. ; Ganz, W. ; Neuburger, M. u. a.: Wachstumsbereiche in der Dienstleistungswirtschaft. In: Bullinger, H. J. (Hrsg.): *Dienstleistung 2000plus.* Stuttgart : Fraunhofer IRB Verlag, 1998, S. 35 ff.

[Bus01] Bussler, C.: The Role of B2B Protocols in Inter-Enterprise Process Execution. In: Casati, F. (Hrsg.) ; Georgakopoulos, D. (Hrsg.) ; Shan, M. C. (Hrsg.): *Technologies for E-Services, Second International Workshop, TES 2001, Rome, Italy, September 14-15, 2001, Proceedings.* Berlin : Springer, 2001 (LNCS 2193), S. 16–29

[Bus03] Bussler, C.: *B2B Integration.* Berlin u.a. : Springer, 2003

[BzG02] Becker, J. ; zur Muehlen, M. ; Gille, M.: Workflow Application Architectures: Classification and Characteristics of Workflow-based Information Systems. In: Fischer, Layna (Hrsg.): *Workflow Handbook 2002.* Lighthouse Point, FL, USA : Future Strategies Inc.„ 2002, S. 39–50

[CCF+05a] Cabrera, L. F. ; Copeland, G. ; Feingold, M. u. a.: Web Services Atomic Transaction (WS-AtomicTransaction) Version 1.0 / Arjuna Technologies, Ltd., BEA Systems, Hitachi, Ltd., International Business Machines Corporation, IONA Technologies, Microsoft Corporation. 2005. – Forschungsbericht

[CCF+05b] Cabrera, L. F. ; Copeland, G. ; Feingold, M. u. a.: Web Services Business Activity Framework (WS-BusinessActivity) Version 1.0 / Arjuna Technologies, Ltd., BEA Systems, Hitachi, Ltd., International Business Machines Corporation, IONA Technologies, Microsoft Corporation. 2005. – Forschungsbericht

[CCF+05c] Cabrera, L. F. ; Copeland, G. ; Feingold, M. u. a.: Web Services Coordination (WS-Coordination) Version 1.0 / Arjuna Technologies, Ltd., BEA Systems, Hitachi, Ltd., International Business Machines Corporation, IONA Technologies, Microsoft Corporation. 2005. – Forschungsbericht

[CCM+01] Christensen, E. ; Curbera, F. ; Meredith, G. u. a.: Web Services Description Language (WSDL) 1.1 / World Wide Web Consortium. 2001 (http://www.w3.org/TR/2001/NOTE-wsdl-20010315). – W3C Note

[CD99] Clark, J. ; DeRose, S.: XML Path Language (XPath) Version 1.0 / World Wide Web Consortium. 1999 (www.w3.org/TR/xpath). – W3C Recommendation

[CDK02] Coulouris, G. ; Dollimore, J. ; Kindberg, T.: *Verteilte Systeme; Konzepte und Design.* München : Pearson Studium, 2002

[CG00] Corsten, H. ; Gössinger, R.: Produktionsplanung und -steuerung in virtuellen Produktionsnetzwerken. In: Kaluza, B. (Hrsg.) ; Blecker, T. (Hrsg.): *Produktions- und Logistikmanagement in Virtuellen Unternehmen und Unternehmensnetzwerken.* Berlin : Springer, 2000, S. 249–294

[CH00] Chen, Q. ; Hsu, M.: Inter-Enterprise Collaborative Business Process Management / Hewlett-Packard Laboratories, Palo Alto, CA, USA. 2000 (HPL-2000-107). – Forschungsbericht

[CH02] Chen, Q. ; Hsu, M.: CPM Revisited - An Architecture Comparison. In: Meersmann, R. (Hrsg.) ; Tari, Z. (Hrsg.): *On the Move to Meaningful Internet Systems, 2002 - DOA/CoopIS/ODBASE 2002 Confederated International Conferences DOA, CoopIS and ODBASE 2002 Irvine, California, USA, October 30 - November 1, 2002, Proceedings.* Berlin : Springer, 2002 (LNCS 2519), S. 72–90

[CIJ+00a] Casati, F. ; Ilnicki, S. ; Jin, L. J. u. a.: Adaptive and Dynamic Service Composition in eFlow / Hewlett-Packard Laboratories, Palo Alto, CA, USA. 2000 (HPL-2000-39). – Forschungsbericht

[CIJ+00b] Casati, F. ; Ilnicki, S. ; Jin, L. J. u. a.: eFlow: a Platform for Developing and Managing Composite e-Services / Hewlett-Packard Laboratories, Palo Alto, CA, USA. 2000 (HPL-2000-36). – Forschungsbericht. – Proceedings Academia/Industry Working Conference on Research Challenges '00. Next Generation Enterprises: Virtual Organizations and Mobile/Pervasive Technologies. AIWORC'00. 27-29 April 2000 Buffalo, NY, USA [IEEE Comput. Soc.; SUNY at Buffalo; ACM SIGMOBILE; AIS; INFORMS; ISF; Student Online] English

[CL01] Child, M. ; Linketsher, N.: Trust issues for e-services and electronic marketplaces. A customer survey / Hewlett-Packard Laboratories, Bristol, UK. 2001 (HPL-01-32). – Forschungsbericht

[CM05] Camarinha-Matos, L. M.: ICT Infrastructures vor VO. In: Camarinha-Matos, L. M. (Hrsg.) ; Afsarmanesh, H. (Hrsg.) ; Ollus, M. (Hrsg.): *Virtual Organisations: Systems and Practices*. New York : Springer, 2005, S. 83–104

[CMA03] Camarinha-Matos, L. M. ; Afsarmanesh, H.: Designing the Information Technology Subsystem. In: Bernus, P. (Hrsg.) ; Nemes, L. (Hrsg.) ; Schmidt, G. (Hrsg.): *Handbook on Enterprise Architecture*. Berlin u.a. : Springer, 2003 (International Handbooks on Information Systems)

[CMAO01] Camarinha-Matos, L. M. ; Afsarmanesh, H. ; Osório, A. L.: Flexibility and safety in a web-base infrastructure for virtual enterprises. In: *International Journal of Computer Integrated Manufacturing* 14 (2001), Nr. 1, S. 66–82

[CNW01] Curbera, F. ; Nagy, W. A. ; Weerawarana, S.: Web Services: Why and How. In: Hailpern, B. (Hrsg.): *OOPSLA 2001 Workshop on Object-Oriented Web Services: Supporting the Development, Deployment and Evolution of Web Services, Monday, 15 October 2001, Tampa, Florida, USA*. IBM Research, 2001, S. 7

[Coa37] Coase, R.H.: The Nature of the Firm. In: *Economica* (1937), Nr. 4, S. 386–405

[Cor98] Corsten, H.: Ansatzpunkte für ein integratives Dienstleistungsmanagement. In: Bruhn, M. (Hrsg.) ; Meffert, H. (Hrsg.): *Handbuch Dienstleistungsmanagement*. Wiesbaden : Gabler, 1998, S. 73–92

[Cor01] Corsten, H. (Hrsg.): *Dienstleistungsmanagement*. Oldenbourg, 2001 (Lehr- und Handbücher der Betriebswirtschaftslehre)

[CS01] Casati, F. ; Shan, M. C.: Dynamic and Adaptive Composition of e-Services. In: *Information Systems* 26 (2001), Nr. 3, S. 143–63

[CSS01] Casati, F. ; Sayal, M. ; Shan, Ming C.: Developing E-Services for Composing E-Services. In: Dittrich, K. R. (Hrsg.) ; Geppert, A. (Hrsg.) ; Norrie, M. C. (Hrsg.): *Advanced Information Systems Engineering, 13th International Conference, CAiSE 2001, Interlaken, Switzerland, June 4-8, 2001, Proceedings*. Berlin u.a. : Springer, 2001 (LNCS 2068), S. 171–186

[DFB+01] Dumas, M. ; Fauvet, M.C ; Benatallah, B. u. a.: Peer-to-Peer Traced Execution of Composite Services. In: Casati, F. (Hrsg.) ; Georgakopoulos, D. (Hrsg.) ; Shan, M. C. (Hrsg.): *Technologies*

for E-Services, Second International Workshop, TES 2001, Rome, Italy, September 14-15, 2001, Proceedings. Berlin u.a. : Springer, 2001 (LNCS 2193), S. 103–117

[DHL01] Dayal, U. ; Hsu, M. ; Ladin, R.: Business Process Coordination: State of the Art, Trends, and Open Issues. In: Apers, P. M. G. (Hrsg.) ; Atzeni, P. (Hrsg.) ; Ceri, S. (Hrsg.) u. a.: *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*. Morgan Kaufmann, 2001, S. 3–13

[DM92] Davidow, W.H. ; Malone, M.S.: *The Virtual Corporation: Structuring and Revitalizing the Corporation for the 21st Century*. New York : Harper Business, 1992

[DMDJ+97] De Michelis, G. ; Dubois, E. ; Jarke, M. u. a.: Cooperative Information Systems: A Manifesto. In: Papazoglou, M. P. (Hrsg.) ; Schlageter, G. (Hrsg.): *Cooperative Information System: Trends and Directions*. Academic Press, 1997

[EST00] Eversheim, W. ; Schellberg, O. ; Terhaag, O.: Gestaltung und Betrieb von Produktionsnetzwerken. In: Kaluza, B. (Hrsg.) ; Blecker, T. (Hrsg.): *Produktions- und Logistikmanagement in Virtuellen Unternehmen und Unternehmensnetzwerken*. Berlin : Springer, 2000, S. 35–59

[Fai98] Faisst, W.: *Die Unterstützung Virtueller Unternehmen durch Informations- und Kommunikationssysteme – eine lebenszyklusorientierte Analyse*, Friedrich Alexander Universität, Erlangen Nürnberg, Dissertation, 1998

[Fis00] Fischer, J.: Nutzung des Internet im interorganisationalen Produktionsmanagement. In: Kaluza, B. (Hrsg.) ; Blecker, T. (Hrsg.): *Produktions- und Logistikmanagement in Virtuellen Unternehmen und Unternehmensnetzwerken*. Berlin : Springer, 2000, S. 421–449

[FK99] Foster, I. (Hrsg.) ; Kesselman, C. (Hrsg.): *The Grid: Blueprint for a New Computing Infrastructure*. San Francisco : Morgan Kaufmann, 1999

[FKN+02a] Foster, I. ; Kesselman, C. ; Nick, J. u. a.: Grid Services for Distributed System Integration. In: *Computer* 35 (2002), Nr. 6, S. 37–46

[FKN+02b] Foster, I. ; Kesselman, C. ; Nick, J. u. a.: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems

Integration - Version: 6/22/2002 / Open Grid Service Infrastructure WG, Global Grid Forum. 2002. – Forschungsbericht

[FKS+05] Foster, I. ; Kishimoto, H. ; Savva, A. u. a.: The Open Grid Services Architecture, Version 1.0 / Open Grid Forum. 2005 (GFD-I.030). – Informational

[FKT01] Foster, I. ; Kesselman, C. ; Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. In: *International J. Supercomputer Applications* 15 (2001), Nr. 3

[FUM+06] Foster, H. ; Uchitel, S. ; Magee, J. u. a.: LTSA-WS: a tool for model-based verification of web service compositions and choreography. In: Osterweil, L. J. (Hrsg.) ; Rombach, H. D. (Hrsg.) ; Soffa, M. L. (Hrsg.): *28th International Conference on Software Engineering (ICSE 2006), Shanghai, China, May 20-28, 2006*. ACM, 2006, S. 771–774

[FW04] Fischer, O. ; Wenzel, B.: *Prozessorientierte Dienstleistungsunterstützung – Workflowbasierte Komposition unternehmensübergreifender Geschäftsprozesse*, University of Hamburg, MIN-Faculty, Department of Informatics, Diplomarbeit (Thesis), 2004

[GAH+00] Grefen, P. ; Aberer, K. ; Hoffner, Y. u. a.: CrossFlow: cross-organizational workflow management in dynamic virtual enterprises. In: *Computer Systems Science and Engineering* 15 (2000), Nr. 5, S. 277–290

[GCP+00] Godart, C. ; Charoy, F. ; Perrin, O. u. a.: Cooperative Workflows to Coordinate Asynchronous Cooperative Applications in a Simple Way. In: Takizawa, M. (Hrsg.): *Proceedings of the Seventh International Conference on Parallel and Distributed Systems. 4-7 July 2000 Iwate, Japan*. Los Alamitos, CA, USA : IEEE Computer Society, 2000, S. 409–416

[Gen06] Gentleware: *Poseidon for UML*. Gentleware, `http://www.gentleware.com/` (Zugriff am 1.9.2006), 2006

[GGF06] GGF: *Open Grid Forum*. `http://www.ogf.org/` (Zugriff am 1.9.2006), 2006

[GHM+03] Gudgin, Martin ; Hadley, Marc ; Mendelsohn, Noah u. a.: SOAP Version 1.2 / World Wide Web Consortium. 2003 (`http://www.w3.org/TR/soap12/`). – W3C Recommendation

[GHS95] Georgakopoulos, D. ; Hornick, M. ; Sheth, A.: An Overview of Workflow Management: from Process Modeling to Workflow Automation

Infrastructure. In: *Distributed and Parallel Databases* 3 (1995), Nr. 2, S. 119–153

[GL02] Ganesarajah, D. ; Lupu, E.: Workflow-Based Composition of Web-Services: A Business Model or a Programming Paradigm? In: Williams, A. D. (Hrsg.): *Proc. 6th International Enterprise Distributed Object Computing Conference (EDOC2002), September 17-20 2002, Lausanne, Swizerland*. Los Alamos, California : IEEE Computer Society, 2002, S. 273–284

[Glo06] Globus: *Towards Open Grid Services Architecture*. Globus Alliance, `http://www.globus.org/ogsa/` (Zugriff am 1.7.2006), 2006

[GPZ03] Gryce, C. ; Piccinelli, G. ; Zirpins, C.: A Provision-Centric Model for Electronic Services. In: *12th IEEE International Workshops on Enabling Technologies (WETICE 2003), Infrastructure for Collaborative Enterprises, 9-11 June 2003, Linz, Austria*. IEEE Computer Society, 2003, S. 113–116

[GR93] Gray, J. ; Reuter, A.: *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann Publishers, 1993

[GR03] Greiner, U. ; Rahm, E.: WEBFLOW: Ein System zur flexiblen Ausführung webbasierter, kooperativer Workflows. In: Weikum, G. (Hrsg.) ; Schöning, H. (Hrsg.) ; Rahm, E. (Hrsg.): *BTW 2003, Datenbanksysteme für Business, Technologie und Web, Tagungsband der 10. BTW-Konferenz, 26.-28. Februar 2003, Leipzig* Bd. 26. GI, 2003, S. 423–432

[Gre05] Gregor, O.: *Dezentrales Discovery von Web Services auf Basis von Peer-to-Peer Netzen*, University of Hamburg, MIN-Faculty, Department of Informatics, Studienarbeit (Bachelor Thesis), 2005

[GS00] Griese, J. ; Sieber, P.: Logistik in Unternehmensnetzwerken und Virtuellen Unternehmen. In: Kaluza, B. (Hrsg.) ; Blecker, T. (Hrsg.): *Produktions- und Logistikmanagement in Virtuellen Unternehmen und Unternehmensnetzwerken*. Berlin : Springer, 2000, S. 307–319

[GS01] Griese, J. ; Sieber, P.: *Betriebliche Geschäftsprozesse : Grundlagen, Beispiele, Konzepte*. Bern : Haupt, 2001

[GSC+99] Georgakopoulos, D. ; Schuster, H. ; Cichocki, A. u. a.: Managing process and service fusion in virtual enterprises. In: *Information Systems* 24 (1999), Nr. 6, S. 429–456

[GSV+94] Gaitanides, M. ; Scholz, R. ; Vrohlings, A. u. a.: *Prozessmanagement: Konzepte, Umsetzungen und Erfahrungen des Reengineering.* München : Hanser, 1994

[GZMW01] Griffel, F. ; Zirpins, C. ; Müller-Wilken, S.: Generative Softwarekonstruktion auf Basis typisierter Komponenten. In: Killat, U. (Hrsg.) ; Lamersdorf, W. (Hrsg.): *Kommunikation in Verteilten Systemen (KiVS), 12. Fachkonferenz der Gesellschaft für Informatik (GI) Fachgruppe „Kommunikation und verteilte Systeme“ (KuVS) unter Beteiligung des VDE/ITG, Hamburg, 20.-23. Februar 2001.* Berlin u.a. : Springer, 2001, S. 325–338

[Hag99] Hagen, C.: *A Generic Kernel for Reliable Process Support*, ETH Zürich, Dissertation (PhD Thesis), 1999

[Ham05] Hammerschall, U.: *Verteilte Systeme und Anwendungen: Architekturkonzepte, Standards und Middleware-Technologien.* München u.a. : Pearson, 2005 (Pearson Studium)

[Har03] Harms, V.: Prozessgestaltung bei Dienstleistungen. In: Pepels, W. (Hrsg.): *Betriebswirtschaft der Dienstleistungen.* Herne Berlin : nwb - Verlag Neue Wirtschaftsbriefe, 2003

[Has92] Haslinger, F.: *Volkswirtschaftliche Gesamtrechnung.* München : Oldenbourg, 1992

[HB03] Hamadi, R. ; Benatallah, B.: A Petri net-based model for web service composition. In: Schewe, K.-D. (Hrsg.) ; Zhou, X. (Hrsg.): *Database Technologies 2003, Proceedings of the 14th Australasian Database Conference, ADC 2003, Adelaide, South Australia, February 2003.* Australian Computer Society, Inc., 2003, S. 191–200

[HC93] Hammer, M. ; Champy, J.: *Reengineering the Cooperation. A Manifesto for Business Revolution.* New York : Harper Business, 1993

[Hei97] Heinrich, L.J.: Grundlagen der Wirtschafttsinformatik. In: Rechenberber, P. (Hrsg.) ; Pomberger, G. (Hrsg.): *Informatik-Handbuch.* München u.a. : Hanser, 1997, S. 859–874

[Hen92] Hentschel, B.: *Dienstleistungsqualität aus Kundensicht. Vom merkmals- zum ereignisorientierten Ansatz*, Katholische Universität Eichstätt Ingolstadt, Lehrstuhl für ABWL und Dienstleistungsmanagement, Dissertation, 1992

[HK87] Hull, R. ; King, R.: Semantic Data Modeling: Survey, Applications, and Research Issues. In: *ACM Computing Surveys* 19 (1987), Nr. 3, S. 201–260

[HNK02] Hanson, J. E. ; Nandi, P. ; Kumaran, S.: Conversation Support for Business Process Integration. In: Williams, A. D. (Hrsg.): *Proc. 6th International Enterprise Distributed Object Computing Conference (EDOC2002), September 17-20 2002, Lausanne, Swizerland*. Los Alamos, California : IEEE Computer Society, 2002, S. 65–74

[Hol95] Hollingsworth, D.: The Workflow Reference Model / Workflow Management Coalition. 1995 (TC00-1003). – Forschungsbericht

[HP00] Hewlett-Packard: *FreightMixer.com, E-services for new business creation*. Diskussionspapier, Hewlett-Packard Laboratories, Bristol, UK, 2000

[HPS+00] Hayes, J. G. ; Peyrovian, E. ; Sarin, S. u. a.: Workflow Interoperability Standards for the Internet. In: *IEEE Internet Computing* 4 (2000), Nr. 3, S. 37–45

[IBM05] IBM: *IBM WebSphere Software*. International Business Machines, `http://www.ibm.com/software/websphere/` (Zugriff am 1.9.2005), 2005

[IFI03] IFIP–IFAC: GERAM: The Generalised Enterprise Reference Architecture and Methodology Version 1.6.3 (Final). In: Bernus, P. (Hrsg.) ; Nemes, L. (Hrsg.) ; Schmidt, G. (Hrsg.): *Handbook on Enterprise Architecture*. Berlin u.a. : Springer, 2003 (International Handbooks on Information Systems), S. 21–63

[Ihd91] Ihde, G. B.: *Transport, Verkehr, Logistik. Gesamtwirtschaftliche Aspekte und einzelwirtschaftliche Handhabung*. München, 1991

[IJ95] ISO/IEC-JTC1/SC21: *Basic Reference Model of Open Distributed Processing, Parts 1-4, ITU-T X.903 - ISO/IEC 10746*. International Organisation for Standardization, 1995

[Jar88] Jarillo, J.C.: On Strategic Networks. In: *Strategic Management* (1988), Nr. 9, S. 31–41

[JBS97] Jablonski, S. (Hrsg.) ; Böhm, M. (Hrsg.) ; Schulze, W. (Hrsg.): *Workflow-Management: Entwicklung von Anwendungen und Systemen; Facetten einer neuen Technologie*. Heidelberg : dpunkt, 1997 (dpunkt-Lehrbuch)

[KBR04] Kavantzas, N. ; Burdett, D. ; Ritzinger, G.: Web Services Choreography Description Language Version 1.0 / World Wide Web Consortium. 2004 (`http://www.w3.org/TR/2004/WD-ws-cdl-10-20041012/`). – W3C Working Draft

[Ker92] Kerner, H.: *Rechnernetze nach OSI.* Bonn u.a. : Addison-Wesley, 1992

[KGV00] Koetsier, M. ; Grefen, P. ; Vonk, J.: Contracts for Cross-Organizational Workflow Management. In: Bauknecht, K. (Hrsg.) ; Madria, S. K. (Hrsg.) ; Pernul, G. (Hrsg.): *Electronic Commerce and Web Technologies, First International Conference, EC-Web 2000, London, UK, September 4-6, 2000, Proceedings* Bd. 1875. Berlin : Springer, 2000, S. 110–121

[Kir95] Kirn, S.: Kooperierende intelligente Agenten in Virtuellen Organisationen. In: *HMD, Theorie und Praxis der Wirtschaftsinformatik* 32 (1995), Nr. 185, S. 24–36

[KL01] Kuno, H. ; Lemon, M.: A Lightweight Dynamic Conversation Controller for E-Services. In: *Third International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS '01).* IEEE Computer Society, 2001, S. 90–99

[KLK+01] Kuno, H. A. ; Lemon, M. ; Karp, A. H. u. a.: Conversations + Interfaces = Business Logic. In: Casati, F. (Hrsg.) ; Georgakopoulos, D. (Hrsg.) ; Shan, M.-C. (Hrsg.): *Technologies for E-Services, Second International Workshop, TES 2001, Rome, Italy, September 14-15, 2001, Proceedings.* Berlin : Springer, 2001 (LNCS 2193), S. 30–43

[Kos62] Kosiol, E.: *Organisation der Unternehmung.* Wiesbaden, 1962

[Kör92] Körner, H.: *Electronic Servicing - Wirkungsanalyse innovativer I&K Technologien im Dienstleistungsbereich.* Schulz-Kirchner Verlag, 1992 (Betriebswirtschaftliche Beiträge, Band 126)

[Kre99] Kreplin, K. D.: *Konkordanz englischer und deutscher Begriffe des Workflow Management.* Workflow Management Coalition (WfMC), `http://www.wfmc.org/standards/docs/Glossay_German.pdf` (Zugriff am 20.8.2005), 1999

[KRR97] Krystek, U. ; Redel, W. ; Reppegather, S.: *Grundzüge virtueller Organisationen: Elemente und Erfolgsfaktoren, Chancen und Risiken.* Wiesbaden : Gabler, 1997

[KRSR98] Kappel, G. ; Rausch-Schott, S. ; Retschitzegger, W.: Coordination in Workflow Management Systems - A Rule Based Approach. In: Conen, Wolfram (Hrsg.) ; Neumann, Gustaf (Hrsg.): *Coordination Technology for Collaborative Applications*. Berlin : Springer, 1998 (LNCS 1364), S. 99–120

[KS95] Krishnakumar, N. ; Sheth, A.: Managing Heterogeneous Multi-System Tasks to Support Enterprise-Wide Operations. In: *Distributed and Parallel Databases* 3 (1995), Nr. 2, S. 155–186

[KST01] Katranuschkov, P. ; Scherer, R. ; Turk, Z.: Intelligent services and tools for concurrent engineering ? An approach towards the next generation of collaboration platforms. In: *ITcon* 6 (2001), Nr. Special Issue Information and Communication Technology Advances in the European Construction Industry, S. 111–128

[Kuh03] Kuhnert, B.: Produktion von Dienstleistungen. In: Pepels, W. (Hrsg.): *Betriebswirtschaft der Dienstleistungen*. Herne Berlin : nwb - Verlag Neue Wirtschaftsbriefe, 2003

[KWD+04] Kummer, O. ; Wienberg, F. ; Duvigneau, M. u. a.: An extensible editor and simulation engine for Petri nets: Renew. In: Cortadella, J. (Hrsg.) ; Reisig, W. (Hrsg.): *25th International Conference on Application and Theory of Petri Nets (ICATPN 2004), Bologna, Italy*. Berlin u.a. : Springer, 2004 (LNCS 3099), S. 484–493

[Lan04] Langenhan, F. O.: Dienstleistungen. In: Lück, W. (Hrsg.): *Lexikon der Betriebswirtschaft*. München : Oldenbourg, 2004, S. 136 f.

[Las98] Lasch, R.: *Marktorientierte Gestaltung von Logistikprozessen*. Wiesbaden : Gabler, 1998

[LAS+00] Lazcano, A. ; Alonso, G. ; Schuldt, H. u. a.: The WISE Approach to Electronic Commerce. In: *International Journal of Computer Systems Science and Engineering* 15 (2000), Nr. 5, S. 343–355

[Lev01] Levine, P.: ebXML Business Process Specification Schema 5 Version 1.01 / OASIS ebXML Business process (ebXML BP) TC. 2001. – Technical Specification

[Ley01] Leymann, F.: Web Services Flow Language (WSFL 1.0) / International Business Machines Corporation. 2001 (`http://www.ibm.com/software/solutions/webservices/pdf/WSFL.pdf`). – Forschungsbericht

[LR00] Leymann, Frank ; Roller, Dieter: *Production Workflow: Concepts and Techniques*. Upper Saddle River (NJ), USA : Prentice Hall, 2000

[LSA+01] Lazcano, A. ; Schuldt, H. ; Alonso, G. u. a.: WISE: Process based E-Commerce. In: *IEEE Data Engineering Bulletin* 24 (2001), Nr. 1, S. 46–51

[LSD05] LSDIS, Large Scale Distributed Information Systems: *METEOR (Managing End-To-End OpeRations)*. University of Georgia, Computer Science Department, `http://lsdis.cs.uga.edu/projects/past/METEOR/` (Zugriff am 1.9.2005), 2005

[LY94] Lee, J. ; Yost, G.: *The PIF Process Interchange Format and Framework*. PIF Working Group, `http://ccs.mit.edu/pif1.html` (Zugriff am 10.9.2005), 1994

[Mal97] Maleri, R.: *Grundlagen der Dienstleistungsproduktion*. Heidelberg : Springer, 1997

[Mar02] Marinescu, D. C.: *Internet-Based Workflow Management - Toward a Semantic Web*. New York : Wiley, 2002 (Parallel and Distributed Computing)

[Mas05] Masak, D.: *Moderne Enterprise Architekturen*. Berlin u.a. : Springer, 2005 (Xpert.press)

[MBB+03] Medjahed, B. ; Benatallah, B. ; Bouguettaya, A. u. a.: Business-to-business interactions: issues and enabling technologies. In: *VLDB* 12 (2003), Nr. 1, S. 59–85

[MBK+01] Mertens, P. ; Bodendorf, F. ; König, W. u. a.: *Grundzüge der Wirtschaftsinformatik*. Berlin : Springer, 2001

[MC94] Malone, T. W. ; Crowston, K.: The Interdisciplinary Study of Coordination. In: *ACM Computing Surveys* 26 (1994), Nr. 1, S. 87–119

[Mer96] Merz, M.: *Elektronische Dienstemärkte: Modelle und Mechanismen zur Unterstützung von Handelstransaktionen in offenen verteilten Systemen*, Universität Hamburg, Fachbereich Informatik, Dissertation, 1996

[Mer97] Mertens, P.: *Lexikon der Wirtschaftsinformatik*. Berlin Heidelberg : Springer, 1997

[Mer99] Merten, U.: *Verteilte Leistungserstellung auf der Basis agentenbasierter Informationssysteme*. Lohmar : Josef Eul Verlag, 1999

[Mer00] Mertens, P.: *Integrierte Informationsverarbeitung 1, Administrations- und Dispositionssysteme in der Industrie*. Bd. 1, Administrations- und Dispositionssysteme in der Industrie. 12. Auflage. Wiesbaden : Gabler, 2000

[Mer02] Merz, M.: *E-Commerce und E-Business: Marktmodelle, Anwendungen und Technologien*. dpunkt, 2002

[MF97] Mertens, P. ; Faisst, W.: Virtuelle Unternehmen: Idee, Informationsverarbeitung, Illusion. In: Scheer, A.-W. (Hrsg.): *18. Saarbrücker Arbeitstagung für Industrie, Dienstleistung und Verwaltung*. Heidelberg : Physica-Verlag, 1997, S. 101–135

[MGE98] Mertens, P. (Hrsg.) ; Griese, J. (Hrsg.) ; Ehrenberg, D. (Hrsg.): *Virtuelle Unternehmen und Informationsverarbeitung*. Berlin : Springer, 1998

[MH04] McGuinness, D. L. ; Harmelen, F. v.: OWL Web Ontology Language Overview / World Wide Web Consortium. 2004 (`http://www.w3.org/TR/2004/REC-owl-features-20040210/`). – W3C Recommendation

[Mic05] Microsoft: *Microsoft .NET Homepage*. Microsoft Corporation, `http://www.microsoft.com/net/` (Zugriff am 1.9.2005), 2005

[Mül05] Müller, J.: *Workflow-based Integration*. Berlin Heidelberg : Springer, 2005 (Xpert.press)

[MM01] Miller, Joaquin ; Mukerji, Jishnu: Model Driven Architecture / Object Management Group (OMG). 2001 (ormsc/2001-07-01). – Forschungsbericht

[MM04] Milanovic, N. ; Malek, M.: Current Solutions for Web Service Composition. In: *IEEE Internet Computing* 8 (2004), Nr. 6, S. 51–59

[MMJL94] Merz, M. ; Müller-Jones, K. ; Lamersdorf, W.: Service Trading and Mediation in Distributed Computing Systems. In: Svobodova, L. (Hrsg.): *Proc. 14th 'International Conference on Distributed Computing Systems', Poznan, Polen*. IEEE Computer Society Press, 1994, S. 450–457

[Moh02] Mohan, C.: Dynamic E-business: Trends in Web Services. In: Buchmann, A. (Hrsg.) ; Casati, F. (Hrsg.) ; Fiege, L (Hrsg.) u. a.: *Proceedings of the Technologies for E-Services Third International Workshop, TES 2002, Hong Kong, China, August 23-24, 2002*. Berlin Heidelberg : Springer, 2002 (LNCS 2444), S. 1–5

[Mos96] Mosch, H.: *Electronic Data Interchange als moderne Form der Informations- und Kommunikationstechnik in der Informationslogistik bei Logistik-Dienstleistungsunternehmen; Analyse der Chancen und Risiken*, Universität Leipzig, Institut für Produktions- und Industrielle Informationswirtschaft, Dissertation, 1996

[MPC01] Mecella, M. ; Pernici, B. ; Craca, P.: Compatibility of e -Services in a Cooperative Multi-platform Environment. In: Casati, F. (Hrsg.) ; Georgakopoulos, D. (Hrsg.) ; Shan, M.-C. (Hrsg.): *Technologies for E-Services: Second International Workshop TES 2001, Rome, Italy, September 14-15, 2001, Proceedings*. Springer, 2001 (LNCS 2193), S. 44–57

[MPP02] Mecella, M. ; Presicce, F. P. ; Pernici, B.: Modeling E-service Orchestration through Petri Nets. In: Buchmann, A. (Hrsg.) ; Casati, F. (Hrsg.) ; Fiege, L (Hrsg.) u. a.: *Proceedings of the Technologies for E-Services Third International Workshop, TES 2002, Hong Kong, China, August 23-24, 2002*. Berlin Heidelberg : Springer, 2002 (LNCS 2444), S. 38–47

[MS84] Miles, R. ; Snow, C.: Fit, Failure, and the Hall of Fame. In: *California Management Review* 26 (1984), Nr. 3, S. 10–28

[MS86] Miles, R. ; Snow, C.: Organisations: New concepts for new forms. In: *California Management Review* (1986), Nr. 28, S. 62–72

[MS97] Müller-Stewens, G.: Auf dem Weg zur Virtualisierung der Prozessorganisation. In: Müller-Stewens, G. (Hrsg.): *Virtualisierung von Organisationen*. Stuttgart und Zürich : Schäffer-Poeschel, 1997, S. 1–21

[MTL99] Merz, M. ; Tu, T. ; Lamersdorf, W.: Electronic Commerce: Technologische und organisatorische Grundlagen. In: *Informatik-Spektrum* 22 (1999), Nr. 5, S. 328–343,

[MWW+98] Muth, P. ; Wodtke, D. ; Weissenfels, J. u. a.: From Centralized Workflow Specification to Distributed Workflow Execution. In: *Journal of Intelligent Information Systems* 2 (1998), S. 159–184

[NL97] Nierstrasz, O. ; Lumpe, M.: Komponenten, Komponentenframeworks und Gluing. In: *HMD, Theorie und Praxis der Wirtschaftsinformatik* 197 (1997), S. 8–23

[Nor34] Nordsiek, F.: *Grundlagen der Organisationslehre*. Stuttgart, 1934

[Nor03] Noran, O.: A Mapping of Individual Architecture Frameworks (GRAI, PERA, C4ISR, CIMOSA, ZACHMANN, ARIS) onto GERAM. In: Bernus, P. (Hrsg.) ; Nemes, L. (Hrsg.) ; Schmidt, G. (Hrsg.): *Handbook on Enterprise Architecture*. Berlin u.a. : Springer, 2003 (International Handbooks on Information Systems), S. 65–210

[OAS05a] OASIS: *OASIS Committees by Category: Web Services and SOA*. The Organization for the Advancement of Structured Information Standards (OASIS), `http://www.oasis-open.org/committees/tc_cat.php?cat=ws` (Zugriff am 1.9.2005), 2005

[OAS05b] OASIS: *OASIS Homepage*. The Organization for the Advancement of Structured Information Standards (OASIS), `http://www.oasis-open.org/` (Zugriff am 1.9.2005), 2005

[OAS05c] OASIS: *OASIS Web Services Business Process Execution Language (WSBPEL) TC*. The Organization for the Advancement of Structured Information Standards (OASIS), `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel` (Zugriff am 1.9.2005), 2005

[OEH02] O'Sullivan, J. ; Edmond, D. ; Hofstede, A.H.M. ter: What's in a service? Towards accurate description of non-functional service properties. In: *Distributed and Parallel Databases* 12 (2002), S. 117–133

[Oes01] Oestereich, Bernd: *Die UML-Kurzreferenz für die Praxis*. München, Wien : Oldenbourg, 2001

[OF96] Osterloh, M. ; Frost, J.: *Prozeßmanagement als Kernkompetenz : wie Sie business reengineering strategisch nutzen können*. Wiesbaden : Gabler, 1996

[Off03] Offermann, S.: *Ein referenznetzbasiertes Modell zur dynamischen Komposition von Web Services*, University of Hamburg, MIN-Faculty, Department of Informatics, Diplomarbeit (Thesis), 2003

[OMG03] OMG: Unified Modeling Language, v1.5 / Object Management Group (OMG). 2003 (formal/2003-03-01). – Forschungsbericht

[OMG04] OMG: Common Object Request Broker Architecture: Core Specification, Version 3.0.3 / Object Management Group (OMG). 2004 (04-03-12). – Forschungsbericht

[OMG05] OMG: Unified Modeling Language: Superstructure, version 2.0 / Object Management Group (OMG). 2005 (formal/05-07-04). – Forschungsbericht

[OMG06] OMG: *The Object Management Group (OMG)*. Object Management Group, `http://www.omg.org/` (Zugriff am 1.5.2006), 2006

[Ott96] Ott, M.C.: Virtuelle Unternehmensführung - Zukunftsweisender Ansatz im Wettlauf um künftige Markterfolge. In: *Office Management* 7-8 (1996), S. 14–17

[OYP03] Orriëns, B. ; Yang, J. ; Papazoglou, M. P.: A Framework for Business Rule Driven Service Composition. In: Benatallah, B. (Hrsg.) ; Shan, M. C. (Hrsg.): *Technologies for E-Services, 4th International Workshop, TES 2003, Berlin, Germany, September 8, 2003, Proceedings*. Berlin : Springer, 2003 (LNCS 2819), S. 14–27

[Par06] Parr, Terence: *ANTLR Translator Generator*. `http://www.antlr.org/` (Zugriff am 1.9.2006), 2006

[PBB+01] Preist, C. ; Byde, A. ; Bartolini, C. u. a.: Towards Agent-Based Service Composition Through Negotiation in Multiple Auctions. In: *Artificial Intelligence and the Simulation of Behaviour* 1 (2001), Nr. 1, S. 109–124

[PDVM01] Piccinelli, G. ; Di Vitantonio, G. ; Mokrushin, L.: Dynamic Service Aggregation in Electronic Marketplaces. In: *Computer Networks* 37 (2001), Nr. 2, S. 95–109

[Pel03] Peltz, C.: Web Service Orchestration and Choreography: a Look at WSCI and BPEL4WS. In: *Web Services Journal* 3 (2003), Nr. 7, S. 46–52

[PEZ+02] Piccinelli, G. ; Emmerich, W. ; Zirpins, C. u. a.: Web Service Interfaces for Inter-Organisational Business Processes: An Infrastructure for Automated Reconciliation. In: Williams, A. D. (Hrsg.): *Proc. 6th International Enterprise Distributed Object Computing Conference (EDOC2002), September 17-20 2002, Lausanne, Swizerland*. Los Alamos, California : IEEE Computer Society, 2002, S. 285–292

[PF95] Picot, A. ; Franck, E.: Prozessorganisation. Eine Bewertung der neuen Ansätze aus Sicht der Organisationslehre. In: Nippa, M. (Hrsg.) ; Picot, A. (Hrsg.): *Prozessmanagement und Reengineering*. Frankfurt, 1995, S. 13–38

[Pfo71] Pfohl, C.: *Marketing-Logistik: Gestaltung, Steuerung und Kontrolle des Warenflusses im modernen Markt*. Mainz, 1971

[Pfo97] Pfohl, C.: Informationsfluß in der Logistikkette. In: Pfohl, C. (Hrsg.): *Informationsfluß in der Logistikkette.* Berlin : Erich Schmidt Verlag, 1997, S. 1–45

[PFW03] Piccinelli, G. ; Finkelstein, A. ; Williams, S. L.: Service-Oriented Workflows: The DySCo Framework. In: *29th EUROMICRO Conference 2003, New Waves in System Architecture, 3-5 September 2003, Belek-Antalya, Turkey.* IEEE Computer Society, 2003, S. 291–297

[PG03] Papazoglou, M. P. ; Georgakopoulos, D.: Service-Oriented Computing: Introduction. In: *Communications of the ACM* 46 (2003), Nr. 10, S. 24–28

[PH06] Papazoglou, M. P. ; Heuvel, W.-J. van d.: Service-Oriented Design and Development Methodology. In: *Int. J. on Web Engineering and technology* 2 (2006), Nr. 4, S. 412–442

[PM93] Picot, A. ; Maier, M.: Information als Wettbewerbsfaktor. In: Pressmar, D. B. (Hrsg.): *Informationsmanagement.* Wiesbaden, 1993 (Schriften zur Unternehmensführung Bd. 4), S. 31–53

[PN98] Picot, A. ; Neuburger, R.: Virtuelle Organisationsformen im Dienstleistungssektor. In: Bruhn, M. (Hrsg.) ; Meffert, H. (Hrsg.): *Handbuch Dienstleistungsmanagement.* Wiesbaden : Gabler, 1998, S. 513–534

[PN00] Picot, A. ; Neuburger, M.: Grundzüge eines Produktionsmanagement in vernetzten Organisationen. In: Kaluza, B. (Hrsg.) ; Blecker, T. (Hrsg.): *Produktions- und Logistikmanagement in Virtuellen Unternehmen und Unternehmensnetzwerken.* Berlin : Springer, 2000, S. 177–188

[PRW03] Picot, A. ; Reichwald, R. ; Wigand, R. T.: *Die grenzenlose Unternehmung.* 5. Auflage. Wiesbaden : Gabler, 2003

[PT01] Pilioura, T. ; Tsalgatidou, A.: E-services: Current Technology and Open Issues. In: Casati, F. (Hrsg.) ; Georgakopoulos, D. (Hrsg.) ; Shan, M. C. (Hrsg.): *Technologies for E-Services, Second International Workshop, TES 2001, Rome, Italy, September 14-15, 2001, Proceedings.* Berlin : Springer, 2001 (LNCS 2193), S. 1–15

[Pud95] Puder, A.: A Declarative Extension of IDL-based Type Definitions within Open Distributed Environments. In: Patel, D. (Hrsg.) ; Sun, Y. (Hrsg.) ; Patel, S. (Hrsg.): *OOIS'94, 1994 International Conference on Object Oriented Information Systems, 19-21 December 1994, London, Proceedings.* Springer, 1995, S. 423–436

[PW03a] Piccinelli, G. ; Williams, S. L.: Workflow: A Language for Composing Web Services. In: Aalst, W. M. P. d. (Hrsg.) ; Hofstede, A.H.M. ter (Hrsg.) ; Weske, M. (Hrsg.): *Business Process Management, International Conference, BPM 2003, Eindhoven, The Netherlands, June 26-27, 2003, Proceedings*. Berlin : Springer, 2003 (LNCS 2678), S. 13–24

[PW03b] Pyke, J. ; Whitehead, R.: Does Better Math Lead to Better Business Processes? / Workflow Management Coalition (WfMC). 2003. – Discussion Paper

[PZL03] Piccinelli, G. ; Zirpins, C. ; Lamersdorf, W.: The FRESCO Framework: An Overview. In: *2003 Symposium on Applications and the Internet Workshops (SAINT 2003), 27-31 January 2003 - Orlando, FL, USA, Proceedings*. IEEE Computer Society, 2003, S. 120–126

[Ram03] Ramme, I.: Darstellung und Bedeutung von Dienstleistungen. In: Pepels, W. (Hrsg.): *Betriebswirtschaft der Dienstleistungen*. Herne Berlin : nwb - Verlag Neue Wirtschaftsbriefe, 2003

[Ren92] Rendez, H.: *Konzeption integrierter Logistik-Dienstleistungen*. München : Huss-Verlag, 1992

[RM00] Reinhart, G. ; Mehler, B.: Organisatorische und informationstechnische Aspekte beim Aufbau virtueller Fabriken. In: Kaluza, B. (Hrsg.) ; Blecker, T. (Hrsg.): *Produktions- und Logistikmanagement in Virtuellen Unternehmen und Unternehmensnetzwerken*. Berlin : Springer, 2000, S. 391–419

[Ros04] RosettaNet: *RosettaNet - eBusiness Standards for the Global Supply Chain*. `http://www.rosettanet.org/` (Zugriff am 28.9.2004), 2004

[Rum91] Rumbaugh, J.: Object-Oriented Modeling and Design. Prentice-Hall, 1991

[RW85] Riordan, M.H. ; Williamson, O.E.: Asset Specificity and Economic Organization. In: *Industrial Organization* (1985), Nr. 3, S. 365–378

[SBC+00] Schuster, H. ; Baker, D. ; Cichocki, A. u. a.: The Collaboration Management Infrastructure. In: *Proceedings of the 16th International Conference on Data Engineering, 28 February - 3 March, 2000, San Diego, California, USA*. Los Alamitos, CA, USA : IEEE Computer Society, 2000, S. 677–678

[SCD+02] Sayal, M. ; Casati, F. ; Dayal, U. u. a.: Integrating Workflow Management Systems with Business-to-Business Interaction Standards. In:

Agrawal, R. (Hrsg.) ; Dittrich, K. (Hrsg.) ; Ngu, A. H. H. (Hrsg.): *Proceedings 18th International Conference on Data Engineering. 26 Feb.-1 March 2002 San Jose, CA, USA*. Los Alamitos, CA, USA : IEEE Computer Society, 2002, S. 287–296

[Sch88] Schumacher, W.: *Die Entwicklung der betriebswirtschaftlichen Logistik und ihr Einfluss auf das zukünftige Leistungsbild des deutschen Speditions- und Lagereigewerbes*. Köln, 1988

[Sch94] Scholz, C.: Die virtuelle Organisation als Strukturkonzept der Zukunft / Universität des Saarlandes, Saarbrücken. 1994 (Arbeitspapier Nr. 30). – Forschungsbericht

[Sch96] Scholz, C.: Virtuelle Organisation - Konzeption und Realisation. In: *Zeitschrift für Organisation* (1996), Nr. 4, S. 204–210

[Sch98] Scheer, A. W.: *Wirtschaftsinformatik : Referenzmodelle für industrielle Geschäftsprozesse*. Berlin : Springer, 1998

[Sch00] Schwarze, J.: *Einführung in die Wirtschaftsinformatik*. Herne Berlin : NWB, 2000

[Sch02a] Schlimmer, J. C.: Web Services Description Requirements / World Wide Web Consortium, Web Services Description Working Group. 2002 (`http://www.w3.org/TR/2002/WD-ws-desc-reqs-20021028`). – W3C Working Draft

[Sch02b] Schönsleben, P.: *Integriertes Logistikmanagement*. Berlin : Springer, 2002

[Sch03] Schröder, C. S.: *Aufbau hierarchiearmer Produktionsnetzwerke - Technologiestrategische Option und organisatorische Gestaltungsaufgabe*, Technische Universität Berlin, Fakultät für Verkehrs- und Maschinensysteme, Dissertation, 2003

[Sch06] Schmidt, D. C.: Model-Driven Engineering. In: *IEEE Computer* 39 (2006), Nr. 2, S. 25–31

[SDN06] SDN: *Java SE at a Glance*. Sun Developer Network (SDN), `http://java.sun.com/javase/index.jsp` (Zugriff am 1.9.2006), 2006

[SG03] Sandholm, T. ; Gawor, J.: Globus Toolkit 3 Core – A Grid Service Container Framework / Globus Alliance. 2003. – Forschungsbericht

[SG05] Sahai, Akhil ; Graupner, Sven: *Web Services in the Enterprise; Concepts, Standards, Solution and Management*. New York, Boston, Dordretch, London, Moscow : Springer Science+Business Media, 2005 (Network and Systems Management)

[SGC+00] Schuster, H. ; Georgakopoulos, D. ; Cichocki, A. u. a.: Modeling and Composing Service-Based and Reference Process-Based Multi-enterprise Processes. In: Wangler, B. (Hrsg.) ; Bergman, L. (Hrsg.): *Advanced Information Systems Engineering, 12th International Conference CAiSE 2000, Stockholm, Sweden, June 5-9, 2000, Proceedings* Bd. 1789. Berlin : Springer, 2000, S. 247–263

[SGT+00] Schlenoff, C. ; Gruninger, M. ; Tissot, F. u. a.: The Process Specification Language (PSL) Overview and Version 1.0 Specification / National Institute of Standards and Technology. 2000 (NISTIR 6459). – Forschungsbericht

[SGW01] Shegalov, G. ; Gillmann, M. ; Weikum, G.: XML-enabled workflow management for e-services across heterogeneous platforms. In: *VLDB Journal* 10 (2001), Nr. 1, S. 91–103

[SH01] Stahlknecht, P. ; Hasenkamp, U.: *Einführung in die Wirtschaftsinformatik*. Heidelberg, 2001

[Sie99a] Sieber, P.: Die Internet-Unterstützung Virtueller Unternehmen. In: Sydow, J. (Hrsg.): *Management von Netzwerkorganisationen: Beiträge aus der Managementforschung*. Wiesbaden : Gabler, 1999, S. 179–213

[Sie99b] Siebert, Holger: Ökonomische Analyse von Unternehmensnetzwerken. In: Sydow, J. (Hrsg.): *Management von Netzwerkorganisationen: Beiträge aus der Managementforschung*. Wiesbaden : Gabler, 1999, S. 7–28

[Sie03] Siegel, J.: *Introduction to OMG's Unified Modeling Language (UML)*. Object Management Group (OMG), `http://www.omg.org/gettingstarted/what_is_uml.htm` (Zugriff am 1.6.2005), 2003

[Sin97] Sinz, E. J.: Architektur von Informationssystemen. In: Rechenberber, P. (Hrsg.) ; Pomberger, G. (Hrsg.): *Informatik-Handbuch*. München u.a. : Hanser, 1997, S. 875–887

[SK00] Specht, D. ; Kahmann, J.: Regelung kooperativer Tätigkeiten im virtuellen Unternehmen. In: Albach, H. (Hrsg.) ; Specht, D. (Hrsg.) ; Wildemann, H. (Hrsg.): *Virtuelle Unternehmen*. Wiesbaden : Gabler, 2000 (ZfB Ergänzungsheft 2/2000), S. 55–74

[SMC92] Snow, C. ; Miles, R. ; Coleman, H.J.: Managing 21st century network organizations. In: *Organizational Dynamics* (1992), Nr. 3, S. 5–19

[Sta87] Stabenau, H.: Die Übernahme von Logistik-Servicefunktionen durch die Spedition. In: *Erfolgspotential Logistikkette, 4. Logistik-Dialog*. Köln : TÜV Rheinland, 1987, S. 136

[SW94] Sydow, J. ; Windeler, A.: Über Netzwerke, virtuelle Integration und Interorganisationsbeziehungen. In: Sydow, J. (Hrsg.) ; Windeler, A. (Hrsg.): *Management Interorganisationaler Beziehungen - Vertrauen, Kontrolle und Informationstechnik*. Opladen, 1994, S. 1–21

[SW99] Sydow, J. ; Windeler, A.: *Steuerung von Netzwerken - Konzepte und Praktiken*. Opladen, 1999

[SW01] Sangiorgi, D. ; Walker, D.: *The PI-calculus: a theory of mobile processes*. Cambridge u.a. : Cambridge Univ. Press, 2001

[Syd92] Sydow, J.: *Strategische Netzwerke: Evolution und Organisation*. Wiesbaden : Gabler, 1992

[Syd99] Sydow, J.: Management von Netzwerkorganisationen - Zum Stand der Forschung. In: Sydow, J. (Hrsg.): *Management von Netzwerkorganisationen: Beiträge aus der Managementforschung*. Wiesbaden : Gabler, 1999, S. 279–314

[SZ99] Schulte-Zurhausen, M.: *Organisation*. München : Vahlen, 1999

[Tan86] Tanenbaum, A.S.: *Computer Networks*. Third Edition. Upper Saddle River, NJ, USA : Prentice-Hall, 1986

[Tay11] Taylor, F.W.: *The Principles of Scientific Management*. Eric Eldred, `http://melbecon.unimelb.edu.au/het/taylor/sciman.htm` (Zugriff am 1.12.2004), 1911

[TCF+03] Tuecke, S. ; Czajkowski, K. ; Foster, I. u. a.: Open Grid Service Infrastructure (OGSI) Version 1.0 / Open Grid Forum. 2003 (GFD-R-P.15). – Proposed Recommendation

[TE02] Tut, M. T. ; Edmond, D.: The Use of Patterns in Service Composition. In: Bussler, C. (Hrsg.) ; Hull, R. (Hrsg.) ; McIlraith, S. A. (Hrsg.) u. a.: *Web Services, E-Business, and the Semantic Web, CAiSE 2002 International Workshop, WES 2002, Toronto, Canada, May 27-28, 2002, Revised Papers*. Berlin : Springer, 2002 (LNCS 2512), S. 28–40

[Tha01] Thatte, S.: *XLANG – Web Services for Business Process Design*. Microsoft Inc., `http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm` (Zugriff am 26.1.2002), 2001

[TP02] Tsalgatidou, A. ; Pilioura, T.: An Overview of Standards and Related Technology in Web Services. In: *Distributed and Parallel Databases* 12 (2002), Nr. 2-3, S. 135–62

[TR01] Tiwana, A. ; Ramesh, B.: e-Services: Problems, Opportunities, and Digital Platforms. In: Sprague, R. H. (Hrsg.): *Proceedings of the 34th Annual Hawaii International Conference on System Sciences. 2001*. Los Alamitos, CA, USA : IEEE Computer Society, 2001, S. 1–8

[TS03] Tanenbaum, A.S. ; Steen, M. van: *Verteilte Systeme: Grundlagen und Paradigmen*. München u.a. : Pearson, 2003 (Pearson Studium)

[UN/93] UN/EDIFACT: EDIFACT (Electronic Data Interchange for Administration, Commerce and Transport) Syntax Rules, EDIFACT Data Element Directory / International Organisation for Standardization. 1993 (ISO 9735, ISO 7372). – Forschungsbericht

[UO04] UN/CEFACT ; OASIS: *ebXML - Enabling a Global Electronic Market*. `http://www.ebxml.org` (Zugriff am 26.9.2004), 2004

[VH96] Venkatraman, N. ; Henderson, C.: The Architecture of Virtual Organizing: Leveraging Three Independant Vectors / Systems Research Center, Boston University School of Management. 1996. – Discussion Paper

[W3C02a] W3C: *Web Service Activity*. World Wide Web Consortium, `http://www.w3.org/2002/ws/` (Zugriff am 20.6.2004), 2002

[W3C02b] W3C: *Web Service Description Working Group*. World Wide Web Consortium, `http://www.w3.org/2002/ws/desc/` (Zugriff am 26.1.2002), 2002

[W3C02c] W3C: *Web Services Architecture Working Group*. World Wide Web Consortium, `http://www.w3.org/2002/ws/arch/` (Zugriff am 26.1.2002), 2002

[W3C02d] W3C: *World Wide Web Consortium - Leading the Web to Its Full Potential*. World Wide Web Consortium, `http://www.w3c.org/` (Zugriff am 26.9.2004), 2002

[WAD+03] Wohed, P. ; Aalst, W. M. P. d. ; Dumas, M. u. a.: Analysis of Web Services Composition Languages: The Case of BPEL4WS. In: Song, I.Y. (Hrsg.) ; Liddle, S.W. (Hrsg.) ; Ling, T.W. (Hrsg.) u. a.: *Conceptual Modeling - ER 2003, 22nd International Conference on Conceptual Modeling, Chicago, IL, USA, October 13-16, 2003, Proceedings*. Springer, 2003 (LNCS 2813), S. 200–215

[Wal98] Wall, F.: Terminology of IS Architectures / Universität Witten/Herdecke, Fakultät für Wirtschaftswissenschaften, Lehrstuhl für Controlling und Informationsmanagement. 1998 (Heft Nr. 50). – Diskussionspapier

[Wen97] Wende, I.: Normen und Spezifikationen. In: Rechenberber, P. (Hrsg.) ; Pomberger, G. (Hrsg.): *Informatik-Handbuch*. München u.a. : Hanser, 1997, S. 875–887

[WfM99] WfMC: Terminology & Glossary / Workflow Management Coalition (WfMC). 1999 (WFMC-TC-1011). – WfMC Specification

[WfM00] WfMC: Workflow Standard – Interoperability Wf-XML Binding / Workflow Management Coalition (WfMC). 2000 (WFMC-TC-1023). – WfMC Specification

[WfM02] WfMC: Workflow Process Definition Interface – XML Process Definition Language 1.0 Final Draft / Workflow Management Coalition (WfMC). 2002 (WFMC-TC-1025). – WfMC Specification

[WfM04] WfMC: *Workflow Management Coalition*. `http://www.wfmc.org` (Zugriff am 20.9.2004), 2004

[WGR+00] Weissenfels, J. ; Gillmann, M. ; Roth, O. u. a.: The Mentorlite Prototype: A Light-Weight Workflow Management System. In: *Proceedings of the 16th International Conference on Data Engineering, 28 February - 3 March, 2000, San Diego, California, USA*. IEEE Computer Society, 2000, S. 685–686

[WHB01] Weitzel, T. ; Harder, T. ; Buxmann, P.: *Electronic Business und EDI mit XML*. Heidelberg : dpunkt.verlag, 2001 (xml.bibliothek)

[WI05] WS-I: *Welcome to the WS-I Organization's web site*. Web Service Interoperability Organization (WS-I), `http://www.ws-i.org/` (Zugriff am 20.12.2005), 2005

[WN98] Winand, U. (Hrsg.) ; Nathusius, K. (Hrsg.): *Unternehmungsnetzwerke und virtuelle Organisationen*. Stuttgart : Schäffer-Poeschel, 1998

[YHP02] Yang, J. ; Heuvel, W.-J. van d. ; Papazoglou, M. P.: Tackling the Challenges of Service Composition in e-Marketplaces. In: *Proceedings of the 12th International Workshop on Research Issues on Data Engineering: Engineering E-Commerce/E-Business Systems (RIDE-2EC 2002), San Jose, CA, USA*. 2002

[YHU02] Yano, K. ; Hara, H. ; Uehara, S.: Collaboration Management Framework for Integrating B-to-B and Internal Processes. In: Williams, A. D. (Hrsg.): *Proc. 6th Int. Enterprise Distributed Object Computing Conference (EDOC2002), Sep. 17-20 2002, Lausanne, Swizerland*. Los Alamos, California : IEEE Computer Society, 2002, S. 75–83

[You89] Yourdon, E.: *Modern Structured Analysis*. Prentice-Hall, 1989

[YP02] Yang, J. ; Papazoglou, M. P.: Web Component: A Substrate for Web Service Reuse and Composition. In: *Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAiSE'02), Toronto, Canada*. 2002, S. 21–36

[YP04] Yang, Jian ; Papazoglou, Mike. P.: Service Components for Managing Service Composition Lifecycle. In: *Information Systems* 29 (2004), Nr. 2, S. 97–125

[Zac87] Zachmann, J. A.: A Framework for Information Systems Architecture. In: *IBM Systems Journal* 26 (1987), Nr. 3

[ZBL03] Zirpins, C. ; Baier, T. ; Lamersdorf, W.: *A Blueprint of Service Engineering*. First European Workshop on Object Orientation and Web Service (EOOWS 2003), Darmstadt, Germany, 21. Juni 2003, `http://www.cs.ucl.ac.uk/staff/g.piccinelli/eoows/documents/paper-zirpins.pdf` (Zugriff am 1.10.2004), 2003

[ZDN+01] Zaev, Z. ; Damschen, G. ; Norton, F. u. a.: *Professional XML Web Services*. Wrox Press, 2001

[ZL04a] Zirpins, C. ; Lamersdorf, W.: Dienstorientierte Kooperationsmuster in servicebasierten Grids. In: *Praxis der Informationsverarbeitung und Kommunikation* 27 (2004), Nr. 3, S. 152–158

[ZL04b] Zirpins, C. ; Lamersdorf, W.: Service Co-operaation Patterns and their Customised Coordination. In: *Second European Workshop on Object Orientation and Web Service (EOOWS), Oslo, Norway, 14 June, 2004*. 2004

[ZLB04] Zirpins, C. ; Lamersdorf, W. ; Baier, T.: Flexible Coordination of Service Interaction Patterns. In: Aiello, M. (Hrsg.) ; Aoyama, M. (Hrsg.) ; Curbera, F. (Hrsg.) u. a.: *Proceedings 2nd International Conference on Service Oriented Computing (ICSOC04), November 15-18, 2004, NY, USA*. New York : ACM Press, 2004, S. 49–56

[ZLP04] Zirpins, C. ; Lamersdorf, W. ; Piccinelli, G.: A Service Oriented Approach to Interorganisational Cooperation. In: Mendes, M. (Hrsg.) ; Suomi, R. (Hrsg.) ; Passos, C. (Hrsg.): *Proceedings IFIP International Conference on Digital Communities in a Networked Society: eCommerce, eBusiness, and eGovernment (I3E) 2003*. Boston Dordrecht London : Kluwer Academic Publishers, 2004, S. 307–318

[ZOL+05] Zirpins, Christian (Hrsg.) ; Ortiz, Guadalupe (Hrsg.) ; Lamerdorf, Winfried (Hrsg.) u. a.: *Engineering Service Compositions: First International Workshop, WESC'05, Amsterdam, The Netherlands, December 2005, Proceedings*. Yorktown Heights : IBM Research Devision, 2005 (IBM Research Report RC23821)

[ZP04] Zirpins, C. ; Piccinelli, G.: Evolution of Service Processes by Rule Based Transformation. In: Lamersdorf, W. (Hrsg.) ; Tschammer, V. (Hrsg.) ; Amarger, S. (Hrsg.): *Proceedings IFIP International Conference on Digital Communities in a Networked Society: eCommerce, eBusiness, and eGovernment (I3E) 2004*. Boston Dordrecht London : Kluwer Academic Publishers, 2004, S. 287–305

[ZPL+04] Zirpins, C. ; Piccinelli, G. ; Lamersdorf, W. u. a.: Object Orientation and Web Services. In: Malenfant, J. (Hrsg.) ; Ostvold, B. M. (Hrsg.): *Object-Oriented Technology. ECOOP 2004 Workshop Reader, ECOOP 2004 Workshop, Oslo, Norway, June 14-18, 2004, Final Reports*. Heidelberg : Springer, 2004 (LNCS 3344), S. 1–9

[ZW97] Zelkowitz, M.V. ; Wallace, D.R.: Experimental Validation in Software Engineering. In: *Information and Software Technology* 39 (1997), S. 735–743

[ZW98] Zelkowitz, M.V. ; Wallace, D.R.: Experimental Models for Validating Technology. In: *IEEE Computer* 31 (1998), Nr. 5, S. 23–31

[ZW00] Zäpfel, G. ; Wasner, M.: Planung und Optimierung von virtuellen Hub-and-Spoke-Transportnetzwerken kooperativer Logistikdienstleister im Sammelgutverkehr. In: Albach, H. (Hrsg.) ; Specht, D. (Hrsg.) ; Wildemann, H. (Hrsg.): *Virtuelle Unternehmen*. Wiesbaden : Gabler, 2000 (ZfB Ergänzungsheft 2/2000), S. 243–260