Universität Hamburg
Fakultät für Mathematik,
Informatik und Naturwissenschaften

Diplomarbeit

Kontextdatenprognose auf mobilen Geräten

Matthias Meiners

matthias@meiners-online.de
Studiengang Informatik
Fachsemester 10

Erstgutachter:	Prof. Dr. Winfried Lamersdorf
Zweitgutachter:	Dr. Andreas Günter
Abgabe:	September 2009

Inhaltsverzeichnis

1 Einleitung

Der Einsatz von Computertechnik unterliegt einem ständigen Wandel. Die Zukunftsvision des *Ubiquitous Computing* zeichnet ein Bild davon, welche Rolle sie in Zukunft spielen könnte. Es sind bereits einige Bausteine zur möglichen Umsetzung der Vision vorhanden, andere befinden sich noch in der Forschung. Einer dieser Bausteine ist die *Kontextdatenprognose auf mobilen Geräten*, die durch ein in dieser Arbeit zu entwickelndes Verfahren in generischer Art und Weise angeboten werden soll. Der nachfolgende Abschnitt soll einen ersten Zugang zu den Themen Ubiquitous Computing und Kontextdatenprognose schaffen und so die Relevanz der „Kontextdatenprognose auf mobilen Geräten“ deutlich machen.

1.1. Motivation

Vor wenigen Jahrzehnten waren noch Mainframerechner die vorherrschende Form von Computern. Die Zentralität dieser Großrechner wurde bald eingeschränkt, indem man Büroarbeitsplätze mit Terminals zur interaktiven Arbeit auf den Mainframerechnern ausstattete. Die Terminals waren jedoch nicht mehr als Bedieneinheiten für die Mainframerechner. Das Prinzip von Mainframe und Terminals wurde erst mit der Einführung des Personal Computers auf breiter Front durchbrochen. Nun verfügte man über mehrere kleine statt eines großen Rechners, mit denen die Menschen autonom arbeiten konnten. Die Kehrseite der Medaille war, dass die Rechner nicht miteinander kooperieren konnten, was aber sowohl aus systemtechnischer als auch aus Anwendungssicht oft wünschenswert ist. Dieses Problem löste man, indem man Rechner vernetzt hat. So konnten Nutzer mit ihren vernetzten PCs auch viele Aufgaben erledigen, bei denen ein zentraler Mainframerechner zuvor im Vorteil war. Eine weitere Verkleinerung von Computern war zu beobachten im Zuge der zunehmenden Nutzung von *mobilen Geräten* wie Notebooks, PDAs und Mobiltelefonen, die häufig über drahtlose Netze auch mit anderen Rechnern vernetzt sind. [KH07, S. VII] [LP06, S. 35]

Mark Weiser geht in seiner Vision des *Ubiquitous Computing* davon aus, dass sich der Trend zu mehr und kleineren Computern fortsetzen wird [Wei91]. Er sieht die Zukunft darin, dass die Menschen von vielen informationstechnischen Geräten umgeben sein werden, die nicht mehr wie klassische Computer bedient werden müssen, sondern die unauffällig in den Hintergrund treten, aber doch allgegenwärtig sind und überall eine Rolle im Leben spielen [Wei91]. Um die Vision des Ubiquitous Computing herum hat sich eine eigene Forschungsrichtung entwickelt. *Context-Awareness* als ein Zweig dieser Forschungsrichtung beschäftigt sich mit Systemen, die den *Kontext* berücksichtigen, der für das Zusammenspiel zwischen Mensch und Computer eine Rolle spielt, und so die explizite Interaktion zwischen Mensch und Computer verringern [Sch07, S. 82]. Das dichte Netz an Computern im Ubiquitous Computing schafft die Möglichkeit, dass Geräte zu-

sammen überall Daten über den Kontext sammeln und sich mit dieser Vielzahl an Daten über den Kontext bewusst werden [Mat07, Vorwort].

In einer Welt mit sogenannten *smarten Geräten* des Ubiquitous Computing, die sich des Kontextes bewusst sind, kann ein Mobiltelefon z.B. die Displaybeleuchtung an die Umgebungshelligkeit als Teil des Kontextes anpassen oder anzeigen, welche Geschäfte sich in der Umgebung befinden.

Beispiel 1.1:
Ein weiteres Beispiel für Systeme mit Kontextbewusstsein findet sich im Anwendungsgebiet kontextbasierter Kooperation [Kun08], bei dem Daten über den Kontext von Geräten dazu genutzt werden, in einem kooperativen Szenario Aufgaben Geräten zuzuteilen, die aufgrund ihres Kontextes dafür geeignet erscheinen. Insbesondere macht es Sinn, dass nur solchen Geräten eine bestimmte Aufgabe zugeteilt wird, für die auch die dafür benötigten Dienste zum gegebenen Zeitpunkt lokal oder entfernt zur Verfügung stehen. Durch Prognose ergibt sich als verbessertes Entscheidungskriterium für die Aufgabenzuteilung in einer dynamischen Umgebung mit veränderlichen Dienstverfügbarkeiten, ob dem Zielgerät voraussichtlich im Zeitraum der Aufgabenausführung die benötigten Dienste zur Verfügung stehen werden.

Durch *Prognosen* über zukünftigen Kontext erhalten Geräte die Möglichkeit, vorausschauend, also proaktiv im Auftrag des Nutzers zu arbeiten [May05].

Beispiel 1.2:
Für ein Mobiltelefon, das sich des Kontextes bewusst ist, ergibt sich z.B. durch Prognose der Dauer, die der Nutzer in der Zukunft telefonieren wird, die Möglichkeit, frühzeitig auf einen drohenden Energieengpass zu reagieren. Das Mobiltelefon kann dann bei drohender Energieknappheit den Nutzer daran erinnern, den Akku rechtzeitig zu Hause zu laden, ihn frühzeitig vor drohender Energieknappheit warnen, so dass er seine Nutzung anpassen kann, oder aber in einen energiesparenden Modus wechseln, in dem z.B. die Displaybeleuchtung reduziert ist oder E-Mails erst bei Bedarf statt schon beim Eintreffen abgerufen werden. Ein realistisches Szenario besteht z.B. darin, dass ein Nutzer regelmäßig morgens das Haus verlässt und in der Mittagspause längere Telefonate führt. Das Mobiltelefon kann dann frühzeitig morgens oder am Abend des Vortags das Laden des Akkus empfehlen, wenn die verbleibende Energie voraussichtlich nicht ausreichen wird. Auf diese Weise erfolgt für Nutzer, die viel telefonieren, und in Situationen mit absehbar intensiver Nutzung eine rechtzeitige Warnung. Bei geringer Nutzung werden dagegen unnötig frühe, störende Hinweise vermieden.

Weitere Anwendungsbereiche für Kontextdatenprognose auf mobilen Geräten sind z.B. Warnungen vor Gefahrensituationen [May05, S. 32], das frühzeitige Laden von Systembibliotheken [May05, S. 32] und das vorausschauende Übertragen von Daten (z.B. aus

dem Internet oder einem großen Datenbestand in einem Firmennetz), um die Probleme langsamer und instabiler Vernetzung mobiler Geräte zu umgehen.

Diese Diplomarbeit beschäftigt sich damit, die Möglichkeit der Kontextdatenprognose für derartige Anwendungen bereitzustellen. Solche kleinen Beispiele sind es, die zusammen im Zuge des Ubiquitous Computing eine ganz neue Qualität des Computereinsatzes im Alltag ausmachen können. Beispiel 1.2 wird in dieser Arbeit wegen seiner Einfachheit immer wieder aufgegriffen werden und die Arbeit als Leitbeispiel begleiten.

1.2. Zielsetzung

Ziel der Arbeit ist die Konzeption und prototypische Implementierung eines generischen Verfahrens zur *Zukunftsprognose*[1] von Kontextdaten auf mobilen Geräten. Unter *Kontextdaten* sollen Daten eines instanziierten *Kontextmodells*, das die Sicht einer Anwendung auf den Kontext widerspiegelt, verstanden werden. Die geforderte Generik des Verfahrens besteht unter Anderem in der Unterstützung möglichst vieler Arten von Kontextdaten statt einer Spezialisierung auf z.B. den Ort. Das Verfahren soll die Fähigkeiten mobiler Geräte im Rahmen des Ubiquitous Computing um die Möglichkeit der Kontextdatenprognose erweitern. Wichtig dafür ist die Abstimmung auf die spezifischen Eigenschaften mobiler Geräte. Neben der Orientierung an der Vision des Ubiquitous Computing sind bei der Konzeption auch konkrete Bedürfnisse des Nutzers zu berücksichtigen.

Die Implementierung des Verfahrens soll als eine Software genutzt werden können, die in Verbindung mit einer Anwendung auf einem mobilen Gerät läuft und für die Anwendung die Prognose von Kontextdaten übernimmt. Es wird vorausgesetzt, dass die Erhebung und eine mögliche Vorverarbeitung der Kontextdaten durch die Anwendung oder eine bestehende Systemsoftware zur Nutzung von Kontextdaten erfolgt. Die Implementierung soll im Kontext des *DEMAC*-Projektes [Kun05] durch die Prognose der Verfügbarkeit von sogenannten Dienstklassen für einzelne Geräte zur Unterstützung kontextbasierter Kooperation erprobt werden.

1.3. Vorgehensweise und Gliederung

Kapitel 2 - Problemfeld: Es erfolgt basierend auf der Literatur eine Darstellung der Themen Ubiquitous Computing, mobile Geräte, Context-Awareness und Prognose. Das Kapitel arbeitet ausgehend von dargestellten Grundlagen Kontextdatenprognose als Herausforderung heraus. Das zentrale Ziel dieses Kapitels ist die Entwicklung eines *Anforderungskataloges*.

Kapitel 3 - Bestehende Lösungsmöglichkeiten: In diesem Kapitel geht es darum, das Spektrum an Lösungsmöglichkeiten für das Problem der Prognose von Kontextdaten in der Literatur zu erschließen. Dabei werden Lösungsmöglichkeiten unterteilt in *einzelne Methoden*, die für Prognoseaufgaben im Allgemeinen in Frage kommen, und komplette, konkret ausgearbeitete, umfassende *Verfahren* für Kontextdatenprognose. Das Kapitel

[1] vgl. Definition 2.5

analysiert und bewertet die Lösungsmöglichkeiten qualitativ anhand des Anforderungskataloges und stellt Ausgangspunkte für das in Kapitel 4 zu entwickelnde Verfahren zusammen. Dabei werden bevorzugt umfassende Lösungsmöglichkeiten berücksichtigt und nach Bedarf weitere Bausteine gesammelt.

Kapitel 4 - Verfahren der strukturierten Kontextdatenprognose: Ausgehend von der Analyse bestehender Lösungsmöglichkeiten erfolgt die Ausarbeitung eines generischen Verfahrens zur Kontextdatenprognose auf mobilen Geräten, in dessen Rahmen mehrere Methoden eingesetzt werden können. Zur Realisierung der Generik erfolgt die Zusammenstellung der Methoden durch ein anwendungsspezifisches *Prognosemodell*. Im vorhergehenden Kapitel untersuchte Methoden werden als Referenzkonfiguration verwendet. Das entwickelte Verfahren koordiniert die Methoden und bildet einen Rahmen für sie in den Bereichen *Prognose*, *Lernen*, *Datenakquisition* und *Vorwissen* über den Kontext für die Prognose. Das Kapitel geht den Weg einer schrittweisen Verfeinerung vom Grundansatz über ein Basissystem zu Erweiterungen.

Kapitel 5 - Implementierung eines Prognoseframeworks: Das Basissystem des entwickelten Verfahrens wird in diesem Kapitel implementiert. Die Rolle des Verfahrens als Rahmen für Methoden wird durch eine Implementierung als Framework umgesetzt. Es erfolgt zudem eine Einrichtung der Implementierung zum Einsatz in der DEMAC-Middleware [Kun05] als eine mögliche Nutzung der Implementierung.

Kapitel 6 - Diskussion und Evaluation: In diesem Kapitel wird zum einen eine quantitative Bewertung nicht-funktionaler Eigenschaften des Verfahrens und dessen Implementierung durchgeführt. Zum anderen wird eine Bewertung anhand des in Kapitel 2 ausgearbeiteten Anforderungskatalogs vorgenommen. Es findet ein Vergleich mit den in Kapitel 3 untersuchten Verfahren statt.

Kapitel 7 - Zusammenfassung und Ausblick: Dieses Kapitel gibt einen Überblick über den Verlauf der Arbeit und deren Ergebnisse. Es werden die wesentlichen Beiträge zur Forschung herausgestellt und Anknüpfungspunkte für andere Arbeiten aufgezeigt.

Die genannten Kapitel bauen aufeinander auf. Bezüge zu vorhergehenden Abschnitten werden in dieser Arbeit zusätzlich zu den üblichen textuellen Verweisen durch anklickbare Links dargestellt, um auf dezente Art und Weise in verstärktem Umfang den Gedankengang der Arbeit nachvollziehbar zu machen und gezielt auf Textstellen innerhalb eines Abschnittes zu verweisen. Dabei referenzieren vielfach auch Verweise der Art „(vgl. Abschnitt 1.3)“ Textstellen innerhalb des genannten Abschnittes. Das Symbol „↑“ am Ende eines Satzes oder Satzteils ist ebenfalls in der elektronischen Version der Arbeit anklickbar und macht darüber hinaus deutlich, dass eine frühere Aussage aufgegriffen wird.

2 Problemfeld

In dieser Arbeit soll ein Verfahren zur Kontextdatenprognose auf mobilen Geräten entwickelt werden. Die verschiedenen Aspekte dieses Ziels bilden das Problemfeld. Es erfolgt eine Einführung in die Themen und die entsprechende Forschung. Dabei werden zunächst die beiden Themenkomplexe *Ubiquitous Computing*, *mobile Geräte* und *Context-Awareness* auf der einen Seite und *Prognose* auf der anderen Seite separat thematisiert. Der Themenkomplex der Prognose spricht das allgemeine Prognose-Problem an und zeigt Möglichkeiten, Grenzen, Arten und Herangehensweisen auf. Durch Kontext als Gegenstand, mobile Geräte als Ausführungsplattform und Ubiquitous Computing als Nutzungsszenario von Kontextdatenprognose auf mobilen Geräten ergeben sich im Vergleich zu Prognose im Allgemeinen zusätzliche Anforderungen, die zur Herausforderung der Kontextdatenprognose auf mobilen Geräten führen. Als wesentliches Ergebnis des Kapitels entsteht aus den zu berücksichtigenden Faktoren ein Anforderungskatalog, der das Gesamtproblem der Kontextdatenprognose auf mobilen Geräten konkretisiert, indem er angibt, was zur Lösung des Problems erreicht werden muss.

2.1. Ubiquitous Computing

Im Jahr 1991 veröffentlichte Mark Weiser in [Wei91] seine Vision vom Computer des 21. Jahrhunderts. Diese Vision ist heute bekannt unter dem Stichwort *Ubiquitous Computing*. Die erste der zentralen und viel zitierten Forderungen für die angestrebte Rolle von Computertechnik, auf die in diesem Abschnitt immer wieder Bezug genommen wird, ist die schon im Wort „ubiquitous“ enthaltene *Allgegenwärtigkeit* von Computertechnik [Sat01, S. 10] [Mat03, S. 3] [Sch07, S. 78]. Die zweite besteht in der Unaufdringlichkeit und Integration in die Umgebung, die nahezu zur *Unsichtbarkeit* der Computertechnik führt [Sat01, S. 10] [Mat03, S. 3] [Sch07, S. 78]. Insgesamt zielt die Vision darauf ab, dass der Mensch im Mittelpunkt steht und von der Computertechnik unterstützt wird, anstatt von ihr belastet zu werden [Mat03, S. 3]. Die Technik wird als Mittel zum Zweck gesehen [Mat03, S. 3].

Mittlerweile haben auch die Begriffe *Pervasive Computing* und *Ambient Intelligence*, die in etwa die gleiche Idee wie das Ubiquitous Computing bezeichnen, und einige weitere, ähnliche Begriffe Verbreitung gefunden. In der Literatur werden diese Begriffe teilweise als synonym angesehen (z.B. in [Sat01, S. 10] und [CK00, S. 2]). Der Begriff Pervasive Computing ist nach Mattern ein von der Industrie geschaffener Begriff [Mat03, S. 4], dem auch nachgesagt wird, dass er mehr auf die aktuelle Technik fokussiert sei [Mat03, S. 4] [Enc06, S. 12]. Wegen der Nähe der drei Begriffe Ubiquitous Computing, Pervasive Computing und Ambient Intelligence zueinander und der nur schwer möglichen Abgrenzung wird in dieser Arbeit ab hier stellvertretend für alle Begriffe die Bezeichnung Ubiquitous Computing verwendet, solange es nicht um Konzepte geht, die spezifisch

für Pervasive Computing oder Ambient Intelligence sind. Die unterschiedlichen Auffassungen vom Ubiquitous Computing eignen sich zur Strukturierung dieses Abschnittes. Dementsprechend wird begonnen mit einer eher minimalen, an aktueller Technik orientierten Sichtweise nach [HMNS03]. Diese Sichtweise geht noch nicht weit über die Forschungsgebiete der (klassischen) *verteilten Systeme* und des *Mobile Computing* hinaus, die als Vorgänger des Ubiquitous Computing angesehen werden können [Sat01, S. 10] und bereits wichtige Grundfunktionalitäten bereitstellen. Eine erweiterte Sichtweise auf das Ubiquitous Computing, die sich aus einer Reihe unterschiedlicher Beiträge aus der Literatur zusammensetzt, wird im Anschluss an die minimale Sichtweise dargestellt. Sie ist deutlich umfassender als die minimale Sichtweise. Die Darstellung der erweiterten Sichtweise wird als Folge logischer Schritte erfolgen. Das heißt, grundlegende Eigenschaften und Funktionalitäten ubiquitärer Systeme, auf denen andere aufbauen, werden zuerst präsentiert.

2.1.1. Minimale Sichtweise

Es gibt eine Reihe von *Basistechniken*, die Voraussetzung für das Ubiquitous Computing sind (dargestellt z.B. in [LP06, S. 36-44], [HMNS03], [Mat03, S. 5-17]). Gemeint sind damit in dieser Arbeit Hardware und Software, die eng mit der Hardware verbunden sind.

Es ist bis in dieses Jahrzehnt hinein ein exponentielles Wachstum der Leistungsfähigkeit von Prozessoren und in weiten Teilen auch der Kapazität von Speichern und der Datenraten von Rechnernetzen zu beobachten (*Mooresches Gesetz*). Zusätzlich wurden erhebliche qualitative technische Fortschritte gemacht wie der Ausbau *drahtloser Weitverkehrsnetze* und die Entwicklung von *Ad-Hoc-Netzwerktechniken*, mit denen Geräte spontan und ohne jegliche Infrastruktur über kurze Distanzen miteinander kommunizieren können. Aufgrund dieser Entwicklungen konnten immer kleinere und immer leistungsfähigere Geräte in großen Stückzahlen produziert werden, die über Netzwerke miteinander kommunizieren. Kleine leistungsfähige Geräte sind heutzutage z.B. eingebettete Geräte und *mobile Geräte* wie Mobiltelefone, PDAs, Navigationsgeräte und mp3-Player. Auch im Bereich der Ein- und Ausgabe ist in Zukunft kleinere, mobilere Hardware zu erwarten wie flexible Displays, elektronisches Papier und auf die Retina projizierende Brillen. [Mat03, S. 5-13]

Ausgehend von diesen Basistechniken werden in [HMNS03, S. 17-22] die folgenden vier Paradigmen postuliert.

- Dezentralisierung
- Konnektivität
- Einfachheit
- Diversifizierung

Dezentralisierung und *Konnektivität* setzen den Trend fort, der mit der Einführung des PCs begonnen hat und sich im Bereich von z.B. eingebetteten und mobilen Geräten fortsetzt. PCs schaffen den Vorteil von Autonomie und Unabhängigkeit des Nutzers. Eingebettete und mobile Geräte begleiten den Nutzer in allen Situationen und sind stets verfügbar.

Durch die Nutzung von Rechnernetzen können die Möglichkeiten der vielen Geräte erweitert werden. Das Internet, Ad-Hoc-Netze, standardisierte Schnittstellen und Konzepte wie das Semantic Web und Services ermöglichen die Kommunikation zwischen Geräten [Mat03, S. 2-3] [HMNS03, S. 21]. So wird die von Weiser geforderte Allgegenwärtigkeit in gewissem Maße erreicht, zum einen durch die Allgegenwärtigkeit der Geräte und zum anderen durch die Allgegenwärtigkeit von Informationen und Diensten, die in [HMNS03] angestrebt wird. Jeder soll seine Software auf jeder Hardware über jedes Netzwerk benutzen können [HMNS03, S. 21]. Die von Weiser geforderte Unsichtbarkeit kann zu einem gewissen Grad durch *Einfachheit* und *Diversifizierung* erreicht werden. Durch eine einfache Bedienung muss sich der Nutzer weniger mit der Technik beschäftigen und kann sich auf seine eigentlichen Aufgaben konzentrieren. Eine einfachere Bedienung wird unter Anderem durch Diversifizierung erreicht, also durch Geräte, die auf bestimmte Aufgaben spezialisiert und somit weniger komplex sind [HMNS03, S. 18-19,21-22] [Sch07, S. 79-80]. Auch neuartige Ein- und Ausgabetechniken könnten zur Einfachheit beitragen.

2.1.2. Erweiterte Sichtweise

Obige minimale Sichtweise auf das Ubiquitous Computing lässt Potential zur Erreichung von Allgegenwärtigkeit und Unsichtbarkeit ungenutzt. Die erweiterte Sichtweise ist erheblich visionärer. Die Vision lässt sich gut durch zwei Sinnbilder veranschaulichen. Beim ersten handelt es sich um das eines *(elektronischen) Assistenten* [ER06, Vorwort], der die gleiche Rolle wie ein menschlicher Assistent einnimmt [Sat01, S. 15]. Das zweite Sinnbild stammt aus der Begriffswelt der Ambient Intelligence nach Encarnação und nennt sich *intelligente Umgebung*. Eine solche Umgebung besitzt Wissen über sogenannte Smart Player sowie ihre Bedürfnisse und bedient sie entsprechend, wobei Smart Player Menschen, Tiere oder sogenannte smarte Objekte sein können [Enc06, S. 4-7]. Neben dem Begriff der intelligenten Umgebung gibt es auch den Begriff des *Smart Space*, der abzugrenzen ist von sogenannten *Smart Things*, die im Gegensatz zu Smart Spaces nicht fest installiert, sondern portabel sind [Fer07, S. 7-8]. Wegen der Ähnlichkeit wird hier der Begriff *Smart Space* stellvertretend für den Begriff der intelligenten Umgebung verwendet. Statt von Smart Objects ist von *Smart Things* oder einfach von *smarten Geräten* die Rede in dieser Arbeit. Eine weitere Idee stellt der *Personal Computing Space* dar, der aus persönlichen Geräten gebildet wird und zwischen dem Nutzer und anderen Smart Spaces oder Smart Things vermittelt [Sat01, S. 13]. Im Umfeld von Weiser wird diese Idee allerdings nicht unterstützt.

Beispiel 2.1:
In der Praxis könnte ein Smart Space z.B. den Wunsch des Nutzers nach einem helleren Fernsehbild als Ziel annehmen. Durch die Fähigkeit der Steuerung einzelner Geräte könnte der Smart Space die Helligkeitseinstellung des Fernsehers ändern und zusätzlich bei Bedarf selbstständig einen weiteren Weg zum Ziel finden, z.B. das Dimmen der Raumbeleuchtung. [Enc06, S. 13]

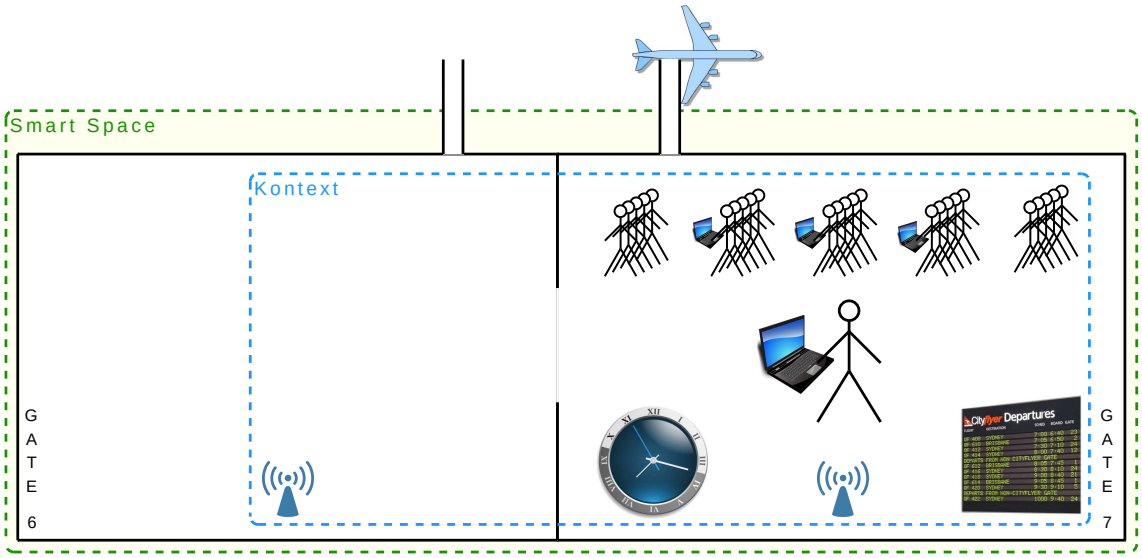

Abbildung 2.1.: grafische Darstellung von Beispiel 2.2

Beispiel 2.2:
In einem anderen Beispielszenario wartet der Nutzer auf dem Flughafen am entsprechenden Gate auf seinen Flug und möchte noch eine größere Menge Daten verschicken. Das drahtlose lokale Netz an seinem Gate ist jedoch so stark ausgelastet, dass der vollständige Versand nicht bis zum Abflug möglich ist. Da die Geräte des Nutzers smart sind, erkennen sie aber das Problem und weisen den Nutzer darauf hin, dass das Netz an einem benachbarten Gate so gering ausgelastet ist, dass er seine Daten dort voraussichtlich vollständig verschicken und wieder rechtzeitig zurück an seinem Gate sein kann. Abbildung 2.1 illustriert dieses Beispiel von Satyanarayanan. [Sat01, S. 12]

Die Frage ist nun, wie man diese Vision der erweiterten Sichtweise realisieren kann. Zunächst einmal greift die erweiterte Sichtweise gegenüber der minimalen auf zusätzliche Basistechniken zurück. Zu nennen sind an dieser Stelle RFID, verbesserte *Sensoren* (z.B. zur Positionsbestimmung), verbesserte Ausgabehardware, Sensornetze und weitere Techniken des sogenannten *Wearable Computing* und der *Augmented Reality* (beschrieben in [Mat03, S. 10-14]). Mit solchen Techniken kann eine Verbindung zwischen Realität und virtueller Realität hergestellt werden [Mat03, S. 26-28]. Wenn man reale Alltagsgegenstände mit Computerchips rechenfähig macht und sie zusätzlich mit Sensoren ausstattet, gewinnen sie die Fähigkeit, Informationen über ihre Umgebung zu erfassen und mit einem virtuellen Modell zu verknüpfen. Durch Kommunikation über Ad-Hoc-Netze und Sensornetze können solche Informationen auch ausgetauscht werden. Eine einfachere Variante im Vergleich dazu, Gegenständen die Fähigkeit zum Rechnen zu verleihen, ist das Versehen von Gegenständen mit Identifikationshardware (z.B. RFID-Tags), so dass andere, rechenfähige Gegenstände sie identifizieren und hinterlegte Informationen über sie abrufen können [Mat03, S. 26-28]. Computertechnik wird durch die Einbettung in die reale Welt noch allgegenwärtiger. Der Nutzer kann mit Computertechnik als Teil seiner

Umgebung in ähnlichen Formen interagieren wie mit seiner bisherigen Umgebung. Das trägt zur Unsichtbarkeit bei. Die Technik wird in die dem Menschen vertraute Umgebung eingebracht und tritt in den Hintergrund [Mat03, S. 4].

Basistechniken allein sind jedoch noch nicht ausreichend. Ein ubiquitäres System kann entscheidend mehr als die Summe seiner Basistechniken [LP06, S. 44] [Sat01, S. 12-13]. Notwendig ist nach Schmidt die konzeptuelle Integration der Computertechnik in den Nutzungsablauf [Sch07, S. 78]. Rein physikalische Allgegenwärtigkeit der Technik und Unsichtbarkeit durch neuartige Interaktionsformen genügen nicht, um den Menschen in den Mittelpunkt zu stellen und ihm als Mittel bei seinen Zwecken zu dienen. Die Herausforderung, der sich auch diese Arbeit stellt, ist das weitere Füllen der Lücke zwischen Basistechniken, etablierten Gebieten wie (klassischen) verteilten Systemen, Mobile Computing und der minimalen Sichtweise auf Ubiquitous Computing auf der einen Seite und der erweiterten Sichtweise und Vision auf der anderen Seite. Wichtige Stufen auf dem Weg zur Realisierung der Vision der erweiterten Sichtweise sind nach Ferscha *Context-Awareness* und *Smartness* [Fer07, S. 3].

2.1.3. Kontext

Context-Awareness bedeutet, dass ein System sich des Kontextes bewusst ist und sein Wissen über den Kontext in geeigneter Weise für den Nutzer einsetzt. *Kontext* charakterisiert die Nutzungssituation. Wenn ein System dem Nutzer als Assistent dienen können soll, muss es sich natürlich wie ein menschlicher Assistent darüber bewusst sein, was die Intentionen des Nutzers sind [Sat01, S. 13], was es selbst für Fähigkeiten hat, welche Umgebung und welche Randbedingungen zugrunde liegen, ob es sich eher um eine entspannte Situation oder eine Notlage handelt, usw. Die Fähigkeit zur Erkennung, Lokalisierung, Wahrnehmung und Vorhersage des Verhaltens von Akteuren und Objekten ist eine essentielle Voraussetzung für die Realisierung intelligenter Systeme [Fer07, S. 6].

Kontext kann weite Bereiche umfassen, von systeminternen Daten bis zu Informationen über den Nutzer und Gegenstände in der Umgebung. Ein gutes Beispiel dafür ist der Datenversand am Flughafen (Beispiel 2.2). Um dem Nutzer den Hinweis geben zu können, dass er an einem anderen Gate noch rechtzeitig alle Daten versenden kann, müssen einige Informationen bekannt sein: Größe und Wichtigkeit der zu versendenden Daten, Abflugzeit, mögliche Verspätungen des Fluges, räumliche Gegebenheiten, die erzielbaren Datenraten für die drahtlosen Netze an den Gates für den aktuellen Zeitpunkt oder besser sogar die nahe Zukunft und sicherlich noch weitere Details. Daten können von lokalen Sensoren, ganzen Sensornetzen, anderen Geräten oder auch aus verschiedenen Schichten des Gerätes selbst [Sat01, S. 12, 16] stammen. Linnhoff-Popien spricht hier auch von einer Wissenskombination [LP06, S. 44]. In der Regel sind aber überwiegend Daten aus der räumlich nahen Umgebung relevant (Prinzip der räumlichen und zeitlichen Nähe) [LP06, S. 45]. Die Daten müssen erfasst, gesammelt, aggregiert und interpretiert werden [Fer07, S. 6]. Aggregation und Interpretation dienen dazu, aus den Mengen erfasster Daten die Informationen zu gewinnen, die von Bedeutung und semantisch verwertbar sind. Eine Rolle spielen dabei auch historische Daten, die z.B. zur Prognose genutzt werden können.

Context-Awareness ist entsprechend obiger Ausführungen eine wichtige Voraussetzung für die Realisierung des Sinnbildes elektronischer Assistenz. Dabei wird das allgegenwärtige Übernehmen und Unterstützen von alltäglichen Abläufen durch Computertechnik ermöglicht und gleichzeitig die Unsichtbarkeit unterstützt. Das gelingt dadurch, dass sich Systeme durch Context-Awareness stark den Bedürfnissen und Erfordernissen des Nutzers annähern. Das System versucht, die Erwartungen des Nutzers zu erfüllen, ohne ihn über den Kontext zu befragen [Enc06, S. 12]. Stattdessen ermittelt es selbst den Kontext [Enc06, S. 12] und wertet ihn aus [LP06, S. 48]. Der Nutzer muss nicht mehr jedes zu lösende Problem selbst analysieren, Lösungsschritte entwickeln und diese in die von Computern angebotenen Operationen übersetzen. Stattdessen begibt sich die Computertechnik in die Welt des Nutzers und übernimmt diese Schritte für den Nutzer. Im Beispiel über die Helligkeit eines Fernsehbildes (Beispiel 2.1) ist es das System, das das Problem erfasst und letztendlich das Dimmen des Lichtes als Lösung einleitet. Das Ausmaß *expliziter Interaktion* zwischen Mensch und Maschine wird verringert [Sch07, S. 82]. Die Mensch-Computer-Interaktion ändert sich grundlegend.

2.1.4. Smartness

Durch Context-Awareness kann ein System sich über den Kontext bewusst werden. Das oben schon vorweggenommene Entwickeln von Lösungsschritten zu Problemen und das Übersetzen in Operationen wird jedoch nicht durch Context-Awareness geleistet und ist somit noch offen. Passend zum jeweiligen Kontext und den aktuellen Problemen müssen geeignete Aktionen gewählt werden. Dabei ist es wünschenswert, dass Systeme die Aktionen wie ein Assistent *autonom* und je nach Situation auch *proaktiv* wählen, um den Nutzer nicht zu belasten und unsichtbar im Hintergrund zu bleiben. Beim Datenversand auf dem Flughafen (Beispiel 2.2) erkennt beispielsweise das System das Problem und macht selbstständig einen Lösungsvorschlag, so dass sich der Nutzer nicht mit technischen Bedingungen wie der Auslastung des drahtlosen lokalen Netzes auseinandersetzen muss. Für proaktives, also vorausschauendes Verhalten kann auf Prognosen über den Kontext zurückgegriffen werden [Fer07, S. 7]. Im Beispiel des Datenversands auf dem Flughafen müssen z.B. die Auslastung der Netze und die Dauer der Datenübertragung prognostiziert werden. Insgesamt wird an dieser Stelle die Beziehung zum Forschungsgebiet von Agenten deutlich.

2.1.5. Problem der Kontrolle

Ein Kritikpunkt am Ubiquitous Computing liegt in dessen „totalitärem Charakter“ [Mat03, S. 35]. Wenn man Aufgaben an einen autonomen Assistenten delegiert, verliert man in der Regel die volle Kontrolle über die Mittel und Wege der Durchführung. Von einem menschenähnlichen Assistenten kann man aber verlangen, dass seine Kompetenzen vom Nutzer festgelegt werden können z.B., weil dieser einfach eine Aufgabe mit Vergnügen selbst ausführt, oder aufgrund der Wichtigkeit der Aufgabe, der Befähigung des Systems und des Nutzers sowie der Unsicherheit bei notwendigen

Entscheidungen und den daraus erwachsenden Risiken.

Wenn das System eine Aufgabe für den Nutzer durchführen soll und eine seiner Entscheidungen mit *Unsicherheit* behaftet ist, z.B. aufgrund einer unsicheren Prognose als Entscheidungsgrundlage, kann es sinnvoll sein, dass sich das System selbst Grenzen setzt und Kontakt mit dem Nutzer aufnimmt. Dies stört zwar zunächst die Unsichtbarkeit, kann aber unerwünschte Ergebnisse vermeiden, die als Fehlfunktion wahrgenommen werden könnten und so die Unsichtbarkeit in vielen Fällen mehr stören würden [Sch07, S. 78]. Ein sich dumm gebendes System weist manchmal eine höhere Benutzerfreundlichkeit und Unsichtbarkeit auf als eines, das versucht, alle Angelegenheiten selbst zu regeln und gegenüber dem Nutzer eine „unergründliche Transparenz“ zu wahren [Müh06, S. 178] [Sat01, S. 16].

Satyanarayanan erkennt, dass eine durch das System proaktiv initiierte Interaktion mit dem Nutzer auch in dem Fall sinnvoll sein kann, dass der Nutzer selbst eine Aufgabe durchführt und das System gegenüber dem Nutzer z.B. durch Prognose einen Wissensvorsprung besitzt, um den Nutzer frühzeitig vor unerwünschten Überraschungen zu bewahren [Sat01, S. 11]. In Beispiel 2.2 kann z.B. abgewendet werden, dass der Nutzer am Flughafen nicht rechtzeitig den Versand seiner Daten abschließen kann. Auch hier wird die Unsichtbarkeit der Technik durch die proaktiv initiierte Interaktion und die dadurch erreichte Vermeidung von technischen Problemen letztendlich verbessert.

Ein weiterer Aspekt des Problems der Kontrolle ergibt sich unter Usability-Gesichtspunkten. Er besteht darin, dass die Bildung von konzeptuellen Modellen durch das Ubiquitous Computing erschwert wird [Sch07, S. 85-86]. Konzeptuelle Modelle werden vom Nutzer gebildet, um das Verhalten eines Systems nachzuvollziehen und dadurch die Bedienung zu planen [Sch07, S. 85-86]. Trotz Autonomie und geringer Notwendigkeit expliziter Bedienung ubiquitärer Systeme besteht die Möglichkeit, dass der Nutzer sich wie bei einem menschlichen Assistenten einen Eindruck davon verschaffen möchte, warum das System wie handelt. *Nachvollziehbarkeit* kann das Vertrauen ins System erhöhen, eine Koordination zwischen System und Nutzer ermöglichen und den Nutzer befähigen, mögliche Fehlentscheidungen des Systems zu korrigieren. Schmidt schlägt den Einsatz von Erklärungskomponenten vor [Sch07, S. 86]. Derartige Möglichkeiten sollten genutzt werden, um Nachvollziehbarkeit herzustellen.

2.1.6. Privatsphäre

Ein Problem beim Ubiquitous Computing ergibt sich aus dem intensiven Austausch von Daten durch Context-Awareness. Die *Privatsphäre* wird gefährdet [Mat03, S. 31-34]. Es handelt sich um einen der häufigsten und gravierendsten Kritikpunkte am Ubiquitous Computing [Mat03, S. 34] [Wei91]. Die Aufgabe des Datenschutzes wird durch das Ubiquitous Computing erheblich verkompliziert [Sat01, S. 16]. Chen fordert, dass Nutzer die Möglichkeit zur Kontrolle darüber innehaben sollten, wer Zugriff auf Daten über ihren Kontext erhält [CK00, S. 12]. Da aber anscheinend noch keine umfassenden, etablierten Richtlinien zum Datenschutz für das Ubiquitous Computing existieren und das Problem mehr den Austausch von Kontextdaten und weniger Kontextdatenprognose betrifft, wird Datenschutz in dieser Arbeit nicht berücksichtigt.

2.2. Mobile Geräte

Mobile Geräte spielen eine wichtige Rolle im Mobile Computing und bieten eine bestimmte Art von Mobilität. Dieser Abschnitt grenzt zunächst den Begriff mobiler Geräte ab und geht dann auf deren spezifische Eigenschaften ein. Es erfolgt eine Diskussion der einzelnen Eigenschaften, die zu Anforderungen für Software auf mobilen Geräten führen. Mit dem Begriff der Adaptivität wird schließlich eine etwas allgemeinere Sichtweise eingeführt, die ebenfalls die Erfordernisse mobiler Geräte thematisiert.

2.2.1. Begriffsbestimmung

Menschen sind nicht fest an einen bestimmten Ort gebunden, sondern in gewissem Maße mobil. Um eine nahtlose Integration in den Alltag und Unsichtbarkeit herzustellen, sollte das Ubiquitous Computing also diese *Mobilität* unterstützen. Das Ubiquitous Computing steht mit Mobilität in Verbindung [Kun08, S. 14]. Satyanarayanan und Imielinski et al. schätzen die Bedeutung des Mobile Computing sehr hoch ein [Sat96, S. 6] [IB94, S. 26].

Das Thema Mobilität betrifft die Beweglichkeit von Einheiten [Tur06, S. 11] im Sinne einer Änderung ihrer Position. Dabei werden mehrere Arten von Mobilität unterschieden. In Abhängigkeit davon, ob es sich um die Mobilität von physikalischen Einheiten in der körperlichen Welt oder des Codes oder Zustands eines Systems als logische Einheiten handelt, spricht man von physikalischer oder logischer Mobilität [RPM00, S. 245]. Andere Autoren unterscheiden verschiedene Arten von Mobilität danach, um welche Art von Einheit es sich handelt, die ihre Position ändert oder auf die von unterschiedlichen Positionen aus zugegriffen werden kann. Roth unterscheidet z.B. zwischen Endgeräte-, Benutzer- und Dienstmobilität [Rot02, S. 7]. Von einem mobilen System kann man zusätzlich zur Beweglichkeit fordern, dass seine Funktionen während der Bewegung genutzt werden können [Kun08, S. 15].

Diese Arbeit widmet sich gemäß ihrer Zielsetzung der Entwicklung eines Verfahrens für mobile Geräte als Art mobiler Systeme. Dementsprechend muss ein mobiles Gerät seine Position ändern können und dabei funktionsfähig bleiben. Mobile Geräte werden in der Regel im Zusammenhang mit (drahtlosen) Netzen betrachtet. Die *Endgerätemobilität* fordert im Speziellen, dass ein Gerät die Fähigkeit der Vernetzung wahrt, während es räumlich bewegt wird [Rot02, S. 7]. Daraus ergibt sich, was unter einem mobilen Gerät verstanden werden soll.

Definition 2.1 (Mobiles Gerät):
Ein *mobiles Gerät* ist ein vernetztes Gerät, das seine Position ändern kann und dabei funktionsfähig, insbesondere potentiell vernetzt bleibt.

In der Literatur ist oft die Rede von Mobile Computing, obwohl Endgerätemobilität die einzige Art von Mobilität ist, die im Text eine Rolle spielt (z.B. in [Sat96] und [IB94]). Dies hebt die Bedeutung von Endgerätemobilität und mobilen Geräten hervor. Das Forschungsthema mobiler Geräte und das des Ubiquitous Computing überschneiden sich.

Mobile Geräte können die Rolle eines Smart Things spielen oder konventionell eingesetzt werden. Im Ubiquitous Computing haben mobile Geräte als Smart Things einen wichtigen Platz, aber daneben existieren auch stationäre Smart Spaces.

Mobile Geräte sind eine gute Basistechnik für die Idee des Personal Computing Space, der einen Nutzer stets unabhängig von seinem Aufenthaltsort umgibt und zwischen anderen Geräten und dem Nutzer vermittelt. Inwieweit ein Personal Computing Space Realität werden wird, ist unklar, aber feststellen kann man schon heute, dass mobile Geräte wie Mobiltelefone und PDAs den mobilen Nutzer begleiten. Sie werden normalerweise von genau einem Nutzer verwendet. Das Szenario eines mobilen Gerätes, das fest einem Nutzer zugeordnet ist und diesen begleitet, ist wegen seiner Bedeutung heutzutage wesentlich für den Anwendungsbereich von Kontextdatenprognose auf mobilen Geräten.

2.2.2. Eigenschaften mobiler Geräte im Überblick

Nach dem Einstieg in das begriffliche Umfeld mobiler Geräte stellt sich die Frage nach deren spezifischen Eigenschaften, die sie von anderen Geräten unterscheiden und die bei der Realisierung des Verfahrens zur Kontextdatenprognose als eine Software für mobile Geräte zu berücksichtigen sind. Im Anschluss an die Feststellung solcher Eigenschaften wird es darum gehen, weitergehende Anforderungen abzuleiten, um den Eigenschaften mobiler Geräte gerecht zu werden.

Die Eigenschaften mobiler Geräte sind verbunden mit deren Technik, die mit der Zeit starke Fortschritte gemacht hat. Trotz rasanter technologischer Entwicklung in vielen Bereichen sind mobile Geräte mit einigen Einschränkungen verbunden. Satyanarayanan unterteilt diese Einschränkungen in seinem häufig zitierten Text „Fundamental Challenges in Mobile Computing“ in vier Bereiche, die er für unabhängig vom technischen Fortschritt hält [Sat96, S. 1][1]:

- **Ressourcenarmut**: Bei vorgegebenen Kosten und technologischem Niveau verfügen mobile Geräte im Vergleich zu stationären Geräten wegen Anforderungen an Größe, Gewicht und Energieverbrauch über geringere Prozessorgeschwindigkeit und Speicherkapazitäten [Sat96, S. 1].
- **Endliche Energiequellen**: Trotz technischer Fortschritte bei Akkus wird man weiterhin auf den Energieverbrauch achten müssen [Sat96, S. 1].
- **Instabile Vernetzung**: Die Qualität mobiler Verbindungen variiert stark [Sat96, S. 1]. Das Spektrum reicht von hoher Geschwindigkeit und Zuverlässigkeit bis zu niedriger Geschwindigkeit und lückenhafter Verfügbarkeit [Sat96, S. 1].
- **Höhere Risiken**: Mobile Geräte unterliegen höheren Risiken als stationäre Geräte, denn sie können gestohlen werden, verloren gehen oder beschädigt werden [Sat96, S. 1].

Der Bereich höherer Risiken ist für diese Arbeit kaum relevant und wird daher im Folgenden nicht aufgegriffen werden.

[1] Die fett gedruckten Bezeichnungen stammen nicht von Satyanarayanan.

2.2.3. Heterogenität

Eine Eigenschaft mobiler Geräte, die von Satyanarayanan nicht erwähnt wird, ist deren *Heterogenität*. Es wurde bereits auf die Diversifizierung von Geräten hingewiesen (vgl. Abschnitt 2.1.1). Abgesehen von unterschiedlichen Funktionen der Geräte variieren auch die Ressourcen und Fähigkeiten zur Vernetzung enorm. Das Spektrum reicht mindestens von Notebooks bis zu Knoten eines Sensornetzes. Die Geräte bilden auch unterschiedliche Plattformen, auf denen Software aufsetzen kann [Kun08, S. 26] [Tur06, S. 16]. Zusätzlich unterliegen die verschiedenen Arten mobiler Geräte einem ständigen Wandel, weshalb sich auch eine Klassifizierung mobiler Geräte als schwierig erweist [Rot02, S. 337]. Es existieren unterschiedliche Ansätze zur Klassifizierung von Geräten [Tur06, S. 13]. Roth unterscheidet z.B. zwischen Universalgeräten und Spezialgeräten und orthogonal dazu zwischen mobilen Standardcomputern (z.B. Notebooks), Bordcomputern (z.B. in Fahrzeugen), Handhelds (z.B. PDAs, E-Books, Digitalkameras), Wearables und Chipkarten [Rot02, S. 337-338]. Das unterschiedliche Niveau an Ressourcen und Fähigkeiten zur Vernetzung mobiler Geräte erfordert eine hohe *Skalierbarkeit*. Außerdem sollte Software für eine breite Einsetzbarkeit plattformunabhängig sein.

2.2.4. Ressourcenarmut und Energie

Die trotz technischer Fortschritte existierende Ressourcenarmut mobiler Geräte im Bereich Prozessorleistung und Speicherkapazität im Vergleich zu stationären Geräten führt dazu, dass auch die Möglichkeiten der Software auf mobilen Geräten im Vergleich zu stationären Geräten eingeschränkt werden. Die Komplexität neuer Software folgt in der Regel den Möglichkeiten der Hardware, z.B. zur Erweiterung der Funktionalität [Mat03, S. 10]. Die Erwartungen der Nutzer steigen entsprechend und können durch die Möglichkeiten mobiler Geräte im Vergleich zu stationären Geräten nicht immer erfüllt werden [Sat01, S. 13]. Softwareentwickler müssen trotz einfacherer Systemsoftware Interoperabilität mit stationären Systemen sicherstellen. Zur Ressourcenarmut kommen die Endlichkeit an Energie und die nur langsamen technischen Fortschritte bei Akkus hinzu. Das Sparen von Energie kann als Schlüsselthema angesehen werden [IB94, S. 22]. Auch Software spielt dabei durch ihre Nutzung von Ressourcen und drahtlosen Netzen eine wichtige Rolle [Mat03, S. 16] [Sat96, S. 6]. Die konsumierte Rechenzeit und die Netzwerknutzung erscheinen wegen des Energieverbrauchs eher noch kritischer als der Speicherbedarf. Der Spagat zwischen der Schonung von Ressourcen und Netzwerkkapazitäten und dem Umgang mit der Ressourcenarmut auf der einen Seite sowie der Erfüllung der Ansprüche des Nutzers auf der anderen Seite kann Software für mobile Geräte nur gelingen, wenn sie für ein bestimmtes Resultat möglichst wenig Ressourcen in Anspruch nimmt, also *effizient* arbeitet.

2.2.5. Instabile Vernetzung

Die Instabilität der Vernetzung hängt von der Art der Netze ab. Die grundsätzliche Verfügbarkeit im Bereich drahtloser Weitverkehrsnetze ist in einigen Ländern inzwischen schon recht weit fortgeschritten, in anderen weniger weit [mis08]. Die verfügbare Ge-

schwindigkeit variiert von Ort zu Ort [mis08] [mis09b] [mis09c] [mis09a]. Drahtlose lokale Netze und insbesondere Ad-Hoc-Netze stellen eine wichtige Alternative zu drahtlosen Weitverkehrsnetzen dar, was sich aus einer Reihe von Gründen ergibt wie der Unabhängigkeit von der Verfügbarkeit von Infrastruktur [Kun08, S. 22], der möglichen Kostenersparnis, der Bedeutung räumlich nahen Kontextes im Ubiquitous Computing und der Notwendigkeit einer (räumlichen) Beschränkung des Datenaustausches zur Sicherstellung der Skalierbarkeit [Sat01, S. 11-12]. Ad-Hoc-Netze führen jedoch aufgrund der geringen Reichweite und der dynamischen Netzwerktopologie zu recht hoher Instabilität der Vernetzung [Kun08, S. 23]. Deshalb ist es letztendlich notwendig, dass Software auf mobilen Geräten mit instabiler Vernetzung umgehen kann, also in dieser Hinsicht robust und fehlertolerant ist.

Wenn ein Gerät trotz unsicherer Netzwerkverbindung zu einem bestimmten Gerät mit diesem kommuniziert, kommt es beim Einsatz synchroner Kommunikation [Kun08, S. 48-49] (wie z.B. im Fall eines klassischen Remote Procedure Call) im Fall des Verlustes der Verbindung nach dem Aufruf und vor der Antwort zu einem möglicherweise lang andauernden Blockieren des Aufrufers, da dieser auf die Antwort wartet. Dieses Problem kann durch sogenannte *asynchrone Kommunikation* gelöst werden, wobei der Aufrufer dann nicht mehr durch Warten auf die Antwort blockiert ist, sondern nach dem Versand der Anfrage seine Arbeit fortsetzt. Asynchrone Kommunikation entkoppelt die beteiligten Geräte [Kun08, S. 48]. Eine etwas speziellere Anwendung asynchroner Kommunikation erfolgt im Zusammenhang mit dem Konzept des Ereignisses. Dabei dienen Nachrichten als Benachrichtigungen zwischen Geräten über Ereignisse als Ausdrucksform für wahrgenommene Zustandsänderungen [Kun08, S. 53]. *Publish/Subscribe*-Systeme ermöglichen, dass sich Interessenten für Ereignisse anmelden und über diese benachrichtigt werden können [Kun08, S. 53]. Dadurch kann vermieden werden, dass ein Gerät periodisch einen Zustand bei einem anderen Gerät abfragt (*Polling*), um sich über Änderungen zu informieren, so dass die Last auf Netzwerkebene verringert wird. Eine weitere Optimierungsmöglichkeit im Bereich der Kommunikation liegt darin, häufig abgefragte Daten (*Hotspots*) nicht on-demand, sondern per broadcast zu versenden [IB94, S. 24-25]. Der Einsatz von asynchroner Kommunikation, Publish/Subscribe-Systemen und dem Broadcast von Hotspots ist bei der Entwicklung von Software für mobile Geräte in Erwägung zu ziehen.

2.2.6. Adaptivität

Ein weiterer Aspekt der Vernetzung im Zusammenhang mit der Beweglichkeit liegt im häufigen Wechsel der Umgebung [IB94, S. 20], in der sich somit immer wieder andere Geräte befinden. Am meisten Änderungen der Umgebung durch Bewegung sind zu erwarten, wenn sich ausschließlich bewegliche Geräte in der Umgebung befinden, also kein Smart Space. Im Kontext des Mobile Computing würde man von einem Ad-Hoc-System im Gegensatz zu einem Infrastruktursystem sprechen [Kun08, S. 20]. Die Bewegung anderer Geräte oder das häufig auftretende Ausschalten dieser [IB94, S. 20] erscheint als Verlust des Kontakts zu ihnen. Aus Perspektive des Netzwerkes ist ein Knoten aus dem Netz verschwunden. Bei der Kommunikation mobiler Geräte untereinander

ohne eine fest installierte Infrastruktur können Ad-Hoc-Netzwerktechniken eingesetzt werden, um mit den Veränderungen auf Netzwerkebene umzugehen [Kun08, S. 22-23], z.B. beim Routing. Durch Bewegungen verändert sich das Gesamtsystem und die Geräte müssen sich anpassen, damit das System weiterhin funktioniert. Neben den wechselnden Netzwerkbedingungen werden durch Positionswechsel auch unterschiedliche Daten durch Sensoren erfasst. Im Ubiquitous Computing kann man die Netzwerkbedingungen und die von Sensoren erfassten Daten zusammen auch als Kontext betrachten, der sich bei Positionsänderung eines mobilen smarten Gerätes oder anderer Geräte in der Umgebung ändert. Ein smartes Gerät, das über Context-Awareness verfügt, muss sich des wechselnden Kontextes bewusst werden und sich entsprechend verhalten. Was im Allgemeinen für das Ubiquitous Computing gilt, hat hier auch im Speziellen für das Mobile Computing seine Bedeutung. Mobile Geräte müssen sich in gewissem Rahmen dynamisch an Veränderungen anpassen [Sat96, S. 1], die aus Bewegungen resultieren. Sie müssen *adaptiv* sein [Sat96, S. 1].

Der Begriff der Adaptivität spielt in [Sat96] eine zentrale Rolle und wird als Schlüssel zur Mobilität bezeichnet [Sat96, S. 6]. Satyanarayanan sieht als Instrument der Adaptivität die Zuteilung von Funktionen zu Geräten [Sat96, S. 2]. Bei Imielinski et al. ist in ähnlichem Zusammenhang die Rede von Konfigurationsmanagement [IB94, S. 23]. Anzubietende Funktionen müssen nicht grundsätzlich lokal realisiert werden, sondern können auf andere Geräte verlagert werden [Mat03, S. 16]. Eine Verlagerung von Funktionen schont lokale Ressourcen, belastet jedoch Netzwerke, schafft Abhängigkeiten von diesen und von entfernten Geräten und schränkt somit die Autonomie ein [Sat96, S. 1]. Zu berücksichtigen sind daher insbesondere die Stabilität von Netzwerkverbindungen, die Verfügbarkeit von Ressourcen und Energie sowie der entstehende Bedarf nach Ressourcen und Energie. Nicht sinnvoll wäre z.B. das Verschicken einer sehr großen Datenmenge, um dann entfernt wenig aufwändige Berechnungen darauf ausführen zu lassen.

Das Übertragen von Funktionen wird z.B. beim Konzept kontextbasierter Kooperation angewendet [Kun08, S. 137-139]. Hier besteht oft die Motivation darin, dass ein Gerät gar nicht über die Fähigkeiten dazu verfügt, allein eine bestimmte Funktion anzubieten [Kun08, S. 138]. Die Entscheidung für oder gegen die Übertragung von Funktionen ist einerseits Gegenstand der Adaptivität mobiler Geräte, andererseits kann oder muss sie manchmal auch schon zur Entwicklungszeit gefällt werden.

2.3. Context-Awareness

Das Thema der Context-Awareness ist zu Beginn der Neunzigerjahre aufgekommen [CK00, S. 1]. In der Literatur werden auch Begriffe wie Kontextadaptivität [LP06, S. 45] und Kontextsensitivität [Fer07, S. 6] verwendet. In Abschnitt 2.1.3 wurde bereits ein grundlegendes Verständnis vermittelt. Context-Awareness spielt insbesondere für die Vision des Ubiquitous Computing eine wichtige Rolle. Dieser Abschnitt gibt einen genaueren Einblick in den Forschungsbereich der Context-Awareness und setzt dabei primär die Anwendung im Zusammenhang mit dem Ubiquitous Computing und mobilen Geräten voraus.

2.3.1. Implikationen vorhergehender Abschnitte

Die vorhergehenden Abschnitte haben sich mit dem Ubiquitous Computing und mobilen Geräten beschäftigt. Ein Ziel dabei besteht darin, dass sich Anwendungen der Mobilität bewusst werden [CK00, S. 2]. Das steht im Gegensatz zu der Forderung nach Transparenz, die im Bereich verteilter Systeme erhoben wird, also der Forderung nach Unsichtbarkeit von Verteilungsaspekten für die Anwendung. Diese Forderung ist im Bezug auf manche neue Anwendungen wie z.B. Navigationssysteme, die Verteilungsaspekte als Kontext berücksichtigen, nur noch bedingt haltbar, weil Verteilungsaspekte dann Gegenstand der Anwendung und keine zu verbergende Belastung sind.

Die Betrachtung speziell mobiler Geräte hat zur Folge, dass der bei mobilen Geräten häufige Fall verstärkt in den Vordergrund rückt, bei dem ein Gerät einem bestimmten Nutzer fest zugeordnet ist und diesen an jeden Ort begleitet (vgl. Abschnitt 2.2.1). In [Tur06, S. 35] ist auch von einer verstärkten Personalisierung in mobilen Anwendungen durch erhöhte Mobilität die Rede.

Die Mobilität von Geräten führt dazu, dass der Ort als Teil des Kontextes eine besondere Rolle spielt, denn er ändert sich häufig [Tur06, S. 35]. Insgesamt wird der Kontext dynamischer durch Mobilität [DA99, S. 2] und das Ubiquitous Computing [DA99, S. 11]. Das wird klar, wenn man an die Bedeutung von räumlich nahem Kontext denkt (vgl. Abschnitt 2.1.3). Durch Positionswechsel eines Nutzers mit seinem Gerät kann sich die Umgebung aus Sicht des Gerätes fast vollständig verändern. Es befinden sich nach einem Positionswechsel möglicherweise völlig andere Menschen und Gegenstände in der Umgebung. Der Nutzer könnte draußen joggen, statt wie vorher drinnen zu telefonieren. Die hohe Dynamik ist ein wichtiges Merkmal, das berücksichtigt werden muss.

2.3.2. Kontextbegriff

Nachdem einige Beziehungen zwischen Context-Awareness und Ubiquitous Computing sowie Mobile Computing aufgezeigt wurden, wird nun der Begriff des Kontextes geklärt. Als Synonyme für Kontext werden auch von manchen Autoren die Begriffe *Umgebung* und *Situation* benutzt [DA99, S. 3]. Die wohl verbreitetste Definition ist die von Dey et al., die auch in dieser Arbeit verwendet wird.

Definition 2.2 (Kontext):
„*Context* is any information that can be used to characterize the situation of an entity. An *entity* is a person, place, or object that is considered relevant to the *interaction* between a user and an application, including the user and applications themselves.“ [DA99, S. 3-4]

Dey et al. sprechen in [DA99, S. 5] auch von Attributen von Entitäten. Attribute werden daher in dieser Arbeit als ein der Definition nahe stehendes Konzept angesehen. Dey et al. motivieren die Nutzung von Kontext durch die mangelhafte Berücksichtigung von Kontext durch Computer bei traditioneller Mensch-Computer-Interaktion [DA99, S. 1] [Dey01, S. 4]. Es folgt ein Beispiel, das die Definition veranschaulicht.

Beispiel 2.3:
Ein Nutzer schreibt gerade ein Dokument, das er innerhalb weniger Stunden vollenden muss. Er steht unter Zeitdruck. Sein Computer stellt sich darauf ein und gestaltet Interaktionen mit dem Nutzer möglichst kurz.

Relevant für die Interaktion sind der Nutzer selbst und das Dokument als in der Definition genannte Entitäten. Die Situation des Nutzers ist unter Anderem durch seine mentale Verfassung, in diesem Fall Anspannung und Konzentration charakterisiert. Die Situation des Dokuments lässt sich dadurch charakterisieren, wie weit dessen Ausarbeitung fortgeschritten ist. Diese Informationen sind es, auf die der Computer zurückgreift.

Die Kontext-Definition von Dey et al. sieht Informationen als Kontext an, unabhängig davon, ob sie explizit vom Nutzer mitgeteilt oder implizit vom System erfasst wurden. Man könnte daher sogar den Text, den der Nutzer in Beispiel 2.3 über die Tastatur in sein Dokument eingibt, als Kontext ansehen, so dass dann jedes konventionelle Textverarbeitungsprogramm nach der Definition mit Kontext umgehen würde. Es wäre denkbar, dass Dey et al. das nicht explizit ausgeschlossen haben, weil es ihnen selbstverständlich schien, Derartiges nicht zum Kontext zu zählen. In allgemeinen Definitionen von Kontext jenseits der Informatik wird nämlich zwischen Kontext und dem, das vom Kontext begleitet wird, unterschieden [Tur06, S. 16]. In der Definition von Dey et al. scheint es die Interaktion zwischen Mensch und Anwendung zu sein, die vom Kontext begleitet wird. Nun gibt es aber Beispiele, in denen diese Unterscheidung schwer zu halten ist. Es handelt sich um Beispiele, bei denen eine klassische Interaktion zwischen Mensch und Anwendung wie bei PCs über Maus, Tastatur und Bildschirm gar nicht mehr stattfindet, also Beispiele, die gerade zum Ubiquitous Computing passen. Die Interaktion kann dann als implizite Interaktion über implizit erfassten Kontext vollständig indirekt erfolgen (vgl. Abschnitt 2.1.3). Eine Anwendung mit einer solchen Art der Kontextnutzung wird hier *kontextgetriebene Anwendung* genannt im Gegensatz zu *kontextgestützten Anwendungen*, bei denen der Kontext die Rolle des Begleiters einer klassischen Interaktion zwischen Mensch und Anwendung einnimmt. Abbildung 2.2 zeigt beide Typen von Anwendungen mit ihren Arten der Kontextnutzung und Interaktion.

Implizite Interaktion über den Kontext liegt z.B. vor beim Mobiltelefon, das zukünftiges Telefonierverhalten zum Energiemanagement prognostiziert (Beispiel 1.2). Wenn die Anwendung den Nutzer nicht explizit vor einem Energiemangel warnt, interagieren der Nutzer und die Anwendung nur über das Telefonierverhalten und den Energieverbrauch miteinander. Die Betrachtung von Kontext als Begleiter einer expliziten Interaktion zwischen Nutzer und Anwendung scheint hier kaum noch möglich zu sein.

Kontext ist nicht nur Beiwerk. Er verschmilzt mit dem eigentlichen Gegenstand der Anwendungen des Ubiquitous Computing, die allgegenwärtig sind, mit dem Alltäglichen verschmelzen, deren Gegenstand in der alltäglichen Umgebung besteht. Letztendlich macht die weite Auffassung von Kontext der Definition von Dey et al. doch durchaus Sinn, denn eine engere Auffassung würde kontextgetriebenen Anwendungen nicht gerecht werden.

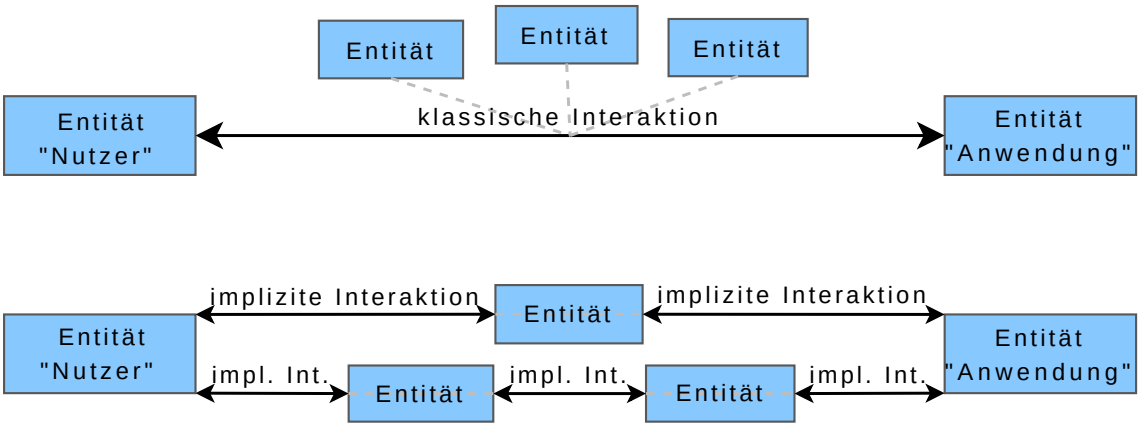

Abbildung 2.2.: kontextgestützte Anwendung (oben) und kontextgetriebene Anwendung (unten)

2.3.3. Kontextarten

Nachdem der Begriff Kontext abgegrenzt wurde, stellt sich die Frage, womit man es in diesem abgegrenzten Bereich zu tun hat. Es existieren zahlreiche Klassifizierungen von Kontext. Turjalei gibt eine umfassende Übersicht [Tur06, S. 34-40]. Die einzelnen Klassifizierungen sind allerdings mit recht unterschiedlichen Sichtweisen verbunden und benutzen teilweise gleiche Begriffe in unterschiedlichen Bedeutungen. Daher wird an dieser Stelle versucht, aufbauend auf [Tur06, S. 34-40] die wesentlichen Unterscheidungsmerkmale in einer vereinheitlichten Form zusammenzufassen. Dabei wird zunächst mit einer Unterscheidung von Kontext über die Art der Entitäten begonnen:

- Bezug zu Menschen (z.B. Nutzer, soziale Umgebung) nach Beigle et al. [BGS99]
- Bezug zu Technik (z.B. Geräte, Netzwerke) nach Schilit et al. [STW93]
- Bezug zu Nutzer oder seinen Geräten (auch „Self“ genannt) nach Schmidt et al. [SAT+99]

Verschiedene Arten von Kontext unterscheiden sich darin, welche dieser Unterscheidungsmerkmale sie besitzen. Es existieren noch zwei andersartige Unterscheidungen von Kontext, die sich nicht an den Arten von Entitäten orientieren:

- Primärkontext und Sekundärkontext nach Dey et al. [DA99]
- Low-Level-Kontext und High-Level-Kontext nach Chen et al. [CK00]

Low- und High-Level-Kontext ziehen die Abstraktionsebene des Kontextes zur Unterscheidung heran. *Primär- und Sekundärkontext* unterscheiden Kontext nach den betrachteten Attributen der Entitäten.

Dey et al. stellen fest, dass *Position*, *Identität*, *Aktivität* und *Zeit* Attribute sind, die die Situation einer Entität ausdrücken [DA99, S. 5]. Obwohl es etliche Arten Kontext gibt, die von Fall zu Fall wichtig sein können [DA99, S. 3], werden diese Attribute *primäre Teile des Kontextes* genannt und es wird ihnen eine besondere Bedeutung in der Praxis zugeschrieben [DA99, S. 4-5]. Die restlichen Teile des Kontextes werden als sekundär

bezeichnet [DA99, S. 5]. Für eine Entität können primäre Teile des Kontextes nach Dey et al. als Indizes fungieren, um für die Entität sekundäre Teile des Kontextes oder für andere Entitäten primäre Teile des Kontextes zu bestimmen [DA99, S. 5]. Die sekundären Teile sind von den primären abhängig. Roth hebt diesbezüglich die Bedeutung der Position hervor [Rot02, S. 249]. Man kann z.B. mit der Identität einer Person ihre Adresse und mit ihrem Aufenthaltsort die Personen in ihrer Nähe bestimmen [DA99, S. 5]. Man gibt also Werte für ein oder mehrere primäre Attribute an und kann so die konkrete Entität durch ihre Identität und ihren Zustand über Zeit, Ort und Aktivität auswählen und entsprechende Werte für bestimmte sekundäre Attribute erhalten. Als komplexeres Beispiel nennen Dey et al. die Wettervorhersage [DA99, S. 5]. Um die Wettervorhersage als sekundären Kontext zu bestimmen, muss man Zeit und Ort als primären Kontext angeben. Festzuhalten ist an dieser Stelle, dass es viele Arten an Kontext gibt, die wichtig werden können und deswegen von Systemen unterstützt werden sollten, und dass es primären Kontext gibt, der besonders wichtig ist, und deshalb möglichst noch besser unterstützt werden sollte.

Wichtig ist auch die Unterscheidung zwischen Low-Level- und High-Level-Kontext. Unter *Low-Level-Kontext* ist Kontext zu verstehen, der direkt von Sensoren stammt [Tur06, S. 35]. Man unterscheidet zwischen physikalischen Sensoren zur Messung physikalischer Größen und logischen Sensoren zur Erfassung von Informationen, die direkt aus dem mobilen Gerät stammen [Tur06, S. 35]. Zum Low-Level-Kontext zählen z.B.: räumliche Position, Helligkeit, Kamerabilder, Umgebungsgeräusche, Puls des Nutzers, Beschleunigung, Temperatur und Systemzeit [Tur06, S. 35-36]. Als *High-Level-Kontext* lassen sich beispielsweise der soziale Kontext [CK00, S. 9], die aktuelle Aktivität [CK00, S. 9] und der Gefühlszustand des Nutzers [Tur06, S.37] einstufen. High-Level-Kontext liegt auf einer höheren Abstraktionsebene als Low-Level-Kontext. Er muss erst durch Berechnungen aus Low-Level-Kontext bestimmt werden [Tur06, S. 35,37]. Abstraktion kann notwendig sein, um Anwendungen die Daten zu liefern, die sie benötigen [Dey00, S. 80]. Der Umgang mit High-Level-Kontext ist auch für das Ubiquitous Computing von großem Nutzen. Ein Assistent kann z.B. viele Aufgaben nicht angemessen durchführen, wenn er kein Wissen über die emotionale Verfassung des Nutzers besitzt wie ein menschlicher Assistent. In Beispiel 2.3 nutzt das System beispielsweise sein Wissen über den Zeitdruck und die Anspannung des Nutzers. Um den Wünschen des Nutzers gerecht werden zu können, ist es insbesondere natürlich wichtig, dass das System Kenntnis besitzt über die Intentionen des Nutzers als ein weiteres Beispiel für High-Level-Kontext (vgl. Abschnitt 2.1.2). Die Gewinnung von High-Level-Kontext kann allerdings sehr kompliziert und fehlerhaft sein [Tur06, S. 37], wie das folgende Beispiel zeigt.

Beispiel 2.4:
Ein Autofahrer und sein Navigationssystem fahren ihre geplante Route und befinden sich gerade in dichtem Verkehr. Sie können nicht erkennen, dass sie sich deshalb in dichtem Verkehr befinden, weil eine benachbarte Straße gesperrt ist und ein Teil ihrer geplanten Route als Umleitungsstrecke eingerichtet wurde.

Die *Unsicherheit* bei der Bestimmung von High-Level-Kontext kann zu falschen Entscheidungen eines Systems führen (vgl. Abschnitt 2.1.5). Manchmal können weder System noch Nutzer sichere Aussagen über den High-Level-Kontext machen. In solchen Situationen ist es wichtig, dass das System die *Unsicherheit* kennt, mit der die Bestimmung des High-Level-Kontext aus dem Low-Level-Kontext verbunden ist, denn die Bestimmung des High-Level-Kontext könnte beispielsweise zu der Annahme führen, dass sich auf der geplanten Route ein Unfall ereignet hat. Ohne Berücksichtigung der Unsicherheit könnte die Konsequenz das Ausweichen auf eine benachbarte Straße sein. Die einzige benachbarte Straße ist aber gerade die gesperrte Straße. Sinnvoller wäre es dagegen z.B. zu warten, bis genügend Informationen über die Verkehrslage verfügbar sind. Entscheidende Voraussetzung für einen solchen Umgang mit Unsicherheit ist, dass sich das System über Unsicherheit bewusst wird, was ermöglicht werden sollte. Ein schönes Beispiel dafür ist das REAL-Navigationssystem für Fußgänger, das zusätzlich zu einem Pfeil, der die zu gehende Richtung anzeigt, eine schematische Karte einblendet, falls aus dem Low-Level-Kontext nicht sicher bestimmt werden kann, in welche Richtung der Nutzer blickt [Tur06, S. 25].

2.3.4. Kontextmodelle

Um automatisiert mit Kontext umgehen zu können, muss man ihn repräsentieren. Zum einen wird bei Kontextdatenprognose intern eine geeignete Repräsentation von Kontext benötigt. Zum anderen wird ein Verfahren zur Kontextdatenprognose auch bei der Zusammenarbeit mit bestehender Software, die Context-Awareness besitzt, mit unterschiedlichen Repräsentationen von Kontext konfrontiert und sollte damit umgehen können.

Die Informatik verwendet zum Repräsentieren in der Regel Modelle, so auch bei Kontext. Modelle können durch Standardisierung als Grundlage zum Austausch von Kontext dienen [Tur06, S. 44]. Der Begriff Modell wird oft in unterschiedlicher Bedeutung verwendet. Die Object Management Group präzisiert den Begriff, indem sie zwischen den folgenden vier Ebenen unterscheidet [Obj09a, S. 16-20].

- Meta-Metamodell-Ebene (M_3)
- Metamodell-Ebene (M_2)
- Modell-Ebene (M_1)
- Instanz-Ebene (M_0)

Die Ebene M_i ist für $i < 3$ jeweils Instanz der Ebene M_{i+1}. Auf der Instanz-Ebene könnte sich z.B. ein konkreter Nutzer befinden, auf der Modell-Ebene das Konzept eines Nutzers im Allgemeinen und auf der Metamodell-Ebene das Konzept einer Entität. In der Regel existiert die Instanz-Ebene bei Software zur Laufzeit einer Anwendung. Die Modell-Ebene wird von den Anwendungsentwicklern definiert. Die Metamodell-Ebene stellt letztendlich für die Anwendungsentwickler das Ausdrucksmittel zum Formulieren ihres Modells dar [Obj09a, S. 16].

Diese Arbeit verwendet den Begriff Kontextdaten, um sich auf die Instanzebene zu beziehen. Die Instanzebene enthält die konkreten Daten, die eine Software verarbeiten muss.

Definition 2.3 (Kontextdaten):
Kontextdaten sind eine konkrete Repräsentation einer Instanz eines gegebenen Kontextmodells. Sie können die Instanz auch im zeitlichen Verlauf darstellen.

Diese Definition führt eine Differenzierung des Kontextbegriffes nach Definition 2.2 ein, indem sie zwischen Modell- und Instanzebene trennt. Kontextdaten drücken letztendlich den Zustand von Entitäten bzw. den zeitlichen Verlauf des Zustandes aus. Dabei wird die Existenz eines Modells des Kontextes vorausgesetzt.

Die Formulierung von Kontextmodellen basiert auf *Metamodellen*, die von anderen Autoren meistens auch Modelle genannt werden. Turjalei unterscheidet eine Reihe verschiedener Metamodelle (vgl. [Tur06, S. 44-48], ähnlich auch in [Kun08, S. 63] und [CK00, S. 10-11]). Für die Zusammenarbeit mit bestehender Software ist ein breites Spektrum an Metamodellen zu unterstützen. Zusätzlich zu allgemeinen Metamodellen existieren für *Ortsinformationen* als Teil des Kontextes spezielle Metamodelle. Chen et al. unterscheiden das auf Koordinaten basierende geometrische Metamodell und das symbolische Metamodell, das abstrakte Symbole zur Repräsentation der Position einsetzt [CK00, S. 10]. Neben dem Ort spielt auch die *Zeit* als Teil des primären Kontextes eine besondere Rolle. Sie kann entweder als unbegrenzt kontinuierlich wachsende oder als eine zyklische Größe (z.B. Wochentag) modelliert werden.

Aktivitäten und Zeit zählen zum primären Kontext als wichtige Art von Kontext. Deshalb scheint die explizite Darstellung durch Konzepte wie *aktive Elemente* wünschenswert. Diese spielen auch bei allgemeinen, nicht speziell für Kontext konzipierten Modellierungsmethoden eine Rolle. Die UML unterscheidet zwischen Struktur und Verhalten, wobei Verhalten durch Aktivitäten sogenannter aktiver Objekte erzeugt wird [Obj09b, S. 15]. Auch bei Petrinetzen unterscheidet man zwischen aktiven Elementen (Transitionen) und passiven Elementen (Plätzen) [Val07, S. 11]. Aktive Elemente sind speziell für eine interne Kontextrepräsentation zur Kontextdatenprognose von Bedeutung, weil aktive Elemente Verhalten, also Veränderungen darstellen und der Zweck von Prognose gerade darin besteht, (Ergebnisse von) Veränderungen vorherzusagen.

Kontext weist eine relativ hohe Dynamik, also relativ starke Veränderungen auf, die insbesondere durch Mobilität verursacht sind (vgl. Abschnitt 2.3.1). Durch Bewegung eines mobilen Gerätes können neue Entitäten im Kontext erscheinen und wieder verschwinden. Es können auch Entitäten kurzzeitig Teil des Kontextes werden, zu denen später nie wieder Kontakt bestehen wird. Da man nicht immer alle möglicherweise in Erscheinung tretenden konkreten Entitäten schon beim Erstellen eines Modells kennen kann, ist es für eine interne Kontextrepräsentation zur Kontextdatenprognose erforderlich, dass zu den Entitätstypen auf der Modellebene zur Laufzeit *Instanzen* auf der Instanzebene erzeugt und wieder zerstört werden können.

2.3.5. Kontextnutzung

Es stellt sich die Frage, wie Kontext eigentlich von Anwendungen eingesetzt wird. Viele bestehende Systeme in der Literatur erscheinen noch relativ simpel im Vergleich zur Vision des Ubiquitous Computing. Sie nutzen oft die räumliche Position als wesentliche Kontextart. Es seien zwei auch in [Tur06, S. 21-34] dargestellten Beispiele genannt, um einen Eindruck zu vermitteln.

Beispiel 2.5:
Tour Guides begleiten den Nutzer, bestimmen die räumliche Position und zeigen ortsbezogene Informationen an, in der Regel eine Karte vom aktuellen Standort und bei Systemen in Museen z.B. Informationen über Gegenstände in der Nähe. Navigationssysteme zeigen als ortsbezogene Information an, welchen Weg der Nutzer wählen muss, um ein bestimmtes Ziel zu erreichen. [Tur06, S. 21-26]

Beispiel 2.6:
Der Besuch einer Konferenz ist normalerweise mit einem gewissen Planungsaufwand verbunden. Der Conference Assistant von Dey und Salber soll Abhilfe schaffen. Er ermittelt entsprechend den Interessen des Konferenzbesuchers geeignete Vorträge. Der Konferenzbesucher trägt ein mobiles Gerät bei sich, das ihm Informationen zu Veranstaltungen in seiner Nähe anzeigt. Folien zur aktuellen Veranstaltung werden vom Gerät geladen und der Konferenzbesucher kann sich Notizen machen. [DSAF99]

Diese Anwendungen gelten als context-aware. Dey et al. haben eine allgemeine Definition entwickelt.

Definition 2.4 (context-aware):
„A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task." [DA99, S. 6]

Diese Definition unterscheidet zwischen der Nutzung von Kontext zur Bereitstellung von Informationen und der Nutzung von Kontext zur Bereitstellung von Services. Es existieren mehrere solcher Unterscheidungen von Arten der Kontextnutzung [SAW94] [CK00, S. 3]. Im Ubiquitous Computing ist besonders eine *automatische, aktive Nutzung* von Kontext [SAW94] [CK00, S. 3] durch ubiquitäre Systeme von Bedeutung, da im Ubiquitous Computing Unsichtbarkeit gefordert wird und sie deshalb wie ein autonomer Assistent eigenständig agieren müssen. Wegen der Forderung nach Allgegenwärtigkeit ubiquitärer Systeme rücken kontextgetriebene Anwendungen in den Mittelpunkt, denn Allgegenwärtigkeit bedeutet, dass Anwendungen zunehmend mit der alltäglichen Umgebung verknüpft sind. Dadurch wird die Umgebung relevant für die Interaktion zwischen An-

wendung und Nutzer und kann somit nach Definition 2.2 als Kontext betrachtet werden. Aus der Autonomie ubiquitärer Systeme und dem engen Bezug zum Kontext kann sich dann auch leicht eine enge Rückkopplung zwischen ubiquitären Systemen und Kontext ergeben, denn Autonomie bedeutet, dass Anwendungen vom Messen des Low-Level-Kontextes bis zur Handlung eigenständig vorgehen, ohne dass der Nutzer letztendlich die Handlung ausführt.

Die automatische Nutzung von Kontext erhöht die *Bedeutung der Bestimmung von High-Level-Kontext aus Low-Level-Kontext* durch das System, da diese Aufgabe bei automatischer Nutzung nicht vom Nutzer ausgeführt werden kann. Aktive, rückgekoppelte Kontextnutzung durch kontextgetriebene Anwendungen, also das Agieren in der alltäglichen Umgebung des Nutzers eröffnet Anwendungen weite Handlungsmöglichkeiten und kann schwere Schäden für den Nutzer hervorrufen. Dies ist ein weiterer Grund für den *bewussten Umgang mit Unsicherheit* bei Entscheidungen von Anwendungen.

2.3.6. Aktivitäten

Der vorhergehende Abschnitt hat sich damit beschäftigt, welche Rolle Kontext in einer Anwendung spielen kann. Bevor Kontext nutzbringend eingesetzt werden kann, ist jedoch die Durchführung der folgenden Schritte nötig bzw. möglich.

- **Messen**: In dieser Arbeit wird unter Messen das Bestimmen von Werten mit Sensoren verstanden. Ergebnis dieses Schrittes sind per Definition Low-Level-Kontextdaten.
- **Repräsentieren**: Kontext muss repräsentiert werden, damit der Umgang damit möglich ist. Dazu wird der Kontext als Kontextdaten im Rahmen eines Kontextmodells dargestellt (vgl. Abschnitt 2.3.4).
- **Austauschen**: Aufgrund der Verteilung, die beim Ubiquitous Computing und dem Mobile Computing vorausgesetzt wird, und der Dezentralisierung beim Ubiquitous Computing können die hier aufgeführten Schritte verteilt erfolgen, was einen Datenaustausch erforderlich macht.
- **Archivieren**: Es besteht die Möglichkeit, Daten andauernd zu speichern, so dass Historien von Daten entstehen [CK00, S. 2].
- **Verarbeiten**: Hierbei werden Kontextdaten verändert oder erweitert.
- **Nutzen**: In diesem Schritt werden die Ergebnisse obiger Schritte unmittelbar für den Anwendungszweck eingesetzt (vgl. Abschnitt 2.3.5).

Chen et al. fasst das Messen und Teile des Austausches und der Verarbeitung unter dem Begriff „sensing“ [CK00, S. 6-9] zusammen. Die Schritte sind partiell geordnet. Kontext muss zuerst gemessen und repräsentiert werden. Dann kann er in beliebiger Reihenfolge beliebig oft ausgetauscht, verarbeitet, archiviert und genutzt werden. Die einzelnen Schritte werden nun genauer erläutert.

Das *Messen* ist an logische und physikalische Sensoren gebunden (vgl. Abschnitt 2.3.3). Im Fall von physikalischen Sensoren muss die Hardware angesteuert werden. Dabei ist die Vielfalt von Kontext [CK00, S. 11] zu bedenken. Logische Sensoren können Daten

auch aus tiefer liegenden Schichten des Systems beziehen. Gemessene Daten können ungenau sein [BF05]. Darin liegt ein erster, zu berücksichtigender Unsicherheitsfaktor.

Der *Austausch* von Daten kann auf zentralisierten Architekturen mit einem Kontextserver oder verteilten Architekturen basieren [CK00, S. 11-12]. Wenn Geräte über Änderungen von Kontext informiert sein müssen, können Publish/Subscribe-Systeme eingesetzt werden. Der Austausch bringt in jedem Fall einige Probleme mit sich. Fehlende Standards verhindern den Datenaustausch zwischen unterschiedlichen Anwendungen [Tur06, S. 44]. Wenn Geräte wegen ihrer Mobilität bzw. instabiler Vernetzung nicht erreichbar, abwesend oder ausgeschaltet sind, ist die Verfügbarkeit von Daten beeinträchtigt [Sig08, S. 77]. Messungen auf unterschiedlichen Geräten erfolgen in unterschiedlichen Intervallen. Solche Schwierigkeiten erschweren insbesondere die Möglichkeiten der Verarbeitung und müssen berücksichtigt werden.

Die *Archivierung* von Kontextdaten kann einer späteren Verarbeitung wie der Prognose [Dey01, S. 6] oder auch schlicht der späteren Nutzung dienen. Nach [CK00, S. 6] aus dem Jahr 2000 werden Historien allgemein als nützlich angesehen für Context-Awareness, jedoch kaum benutzt. Durch die Speicherung von Historien können große Datenmengen anfallen, die den Einschränkungen mobiler Geräte (vgl. Abschnitt 2.2.2) nicht gerecht werden. Deshalb sollte eine Archivierung möglichst nur in beschränktem Umfang eingesetzt werden.

Bei der *Verarbeitung* handelt es sich um ein etwas weiteres Gebiet. Sigg unterscheidet zwischen *horizontaler Verarbeitung* und *vertikaler Verarbeitung* [Sig08, S. 53]. Vertikale Verarbeitung verändert im Gegensatz zu horizontaler Verarbeitung den Abstraktionsgrad der Daten [Sig08, S. 53], ermöglicht also die Gewinnung von High-Level-Kontextdaten aus Low-Level-Kontextdaten. Beispiele sind Aggregation, Interpretation und die Entfernung von Rauschen [Sig08, S. 51-52]. Leider werden die Begriffe Aggregation und Interpretation in unterschiedlichen Bedeutungen verwendet. In dieser Arbeit soll unter der Aggregation von Kontextdaten das bloße Zusammenfassen der Daten verstanden werden. Zu den Arten horizontaler Verarbeitung zählt nach Sigg die Prognose von Kontextdaten [Sig08, S. 53]. Bei einer weiteren Auffassung von Prognose könnte diese allerdings auch unter Rückgriff auf Wissen über Low-Level-Kontext Aussagen über High-Level-Kontext in der Zukunft machen und somit vertikale Verarbeitung beinhalten. Es ist außerdem hervorzuheben, dass eine enge Beziehung zwischen Verarbeitung und dem Treffen von Entscheidungen durch die Anwendung besteht. Im Extremfall könnte eine Verarbeitung als Ergebnis sogar die beste Entscheidung bestimmen. Kritisch ist bei Verarbeitungen, die auch sehr komplex sein können, der Zeitbedarf. Es ist zu vermeiden, dass aufwändige Verarbeitungen die Reaktionszeiten von Anwendungen unangenehm für den Nutzer verlängern. Das würde die Unsichtbarkeit im Ubiquitous Computing beeinträchtigen. Zusätzlich muss bei Rückkopplungen zwischen ubiquitären Systemen und dem Kontext zeitgerecht auf Ereignisse im Kontext reagiert werden, z.B. bei Navigationssystemen. Es herrschen dann Echtzeitbedingungen.

Die *Nutzung* von Kontext wurde bereits in Abschnitt 2.3.5 behandelt. Ein noch nicht geklärter, softwaretechnischer Gesichtspunkt besteht darin, wie die Anwendung für die Nutzung über Änderungen des Kontextes informiert wird. Die bevorzugte, allgemein

anerkannte Lösung für ein solches Problem ist das Beobachter-Muster von Gamma et al. [GHJV95, S. 293-295], das dem Publish-Subscribe-Mechanismus entspricht. Solche Beobachtungsmechanismen werden typischerweise auch bei Anwendungen mit Context-Awareness angewandt (vgl. auch [CK00, S. 9]) und wurden oben schon als Methode für den Austausch erwähnt.

2.3.7. Systemunterstützung

In der Vergangenheit sind Anwendungen mit Context-Awareness eher auf eine Ad-Hoc-Art entwickelt worden, was die Entwicklung schwierig machte [Dey00, S. 79]. Inzwischen existiert Software, die von Anwendungen benutzt werden kann und Aktivitäten der Context-Awareness wie z.B. das Messen und Repräsentieren von Kontext übernimmt. Für eine solche Software werden eine lose Kopplung mit der Anwendung und eine hohe Wiederverwendbarkeit [CK00, S. 11] [Tur06, S. 44] sowie das Verbergen von Details zur Durchführung der Aktivitäten [Tur06, S. 48] angestrebt. Durch eine solche *Systemsoftware*, die auch in dieser Arbeit speziell für die Aufgabe der Kontextdatenprognose entwickelt werden soll ↑, kann der Aufwand für Anwendungsentwickler verringert werden, da diese sich im Idealfall nur noch mit der Nutzung von Kontext beschäftigen müssen.

Es ist aber bei der Entwicklung von Systemsoftware in Form einer Bibliothek, eines Frameworks oder einer Middleware auch der *Aufwand* zu berücksichtigen, der durch den Einsatz für Anwendungsentwickler erst entsteht, z.B. der Aufwand zur Einarbeitung. Der Aufwand für Anwendungsentwickler sollte möglichst gering gehalten werden, damit sich der Einsatz einer Systemsoftware für diese auch lohnt. Es soll dabei im Folgenden unterschieden werden zwischen *Aufwand beim Einsatz*, z.B. beim Aufruf von Methoden einer Bibliothek, und *Vorbereitungsaufwand*, der überwiegend einmalig anfällt und nicht unbedingt alle Anwendungsentwickler betrifft. Der Aufwand der Einarbeitung und auch der Aufwand beim Erstellen von Kontextmodellen sind Beispiele für Vorbereitungsaufwand.

Die in dieser Arbeit zu entwickelnde Systemsoftware, die sich auf Prognose beschränkt, sollte laut Zielsetzung mit bestehender Context-Awareness-Systemsoftware zusammenarbeiten können. Systemsoftware zur Kontextdatenprognose, allgemeine Context-Awareness-Systemsoftware, die Anwendung und Personen, die dazu in Beziehung stehen, nehmen bestimmte Rollen ein, die an dieser Stelle im Zusammenhang festgelegt werden.

> Unter einer *Anwendung* wird in dieser Arbeit Software verstanden, die mit dem *(Anwendungs-)Nutzer* direkt oder indirekt über seine Umgebung interagiert und versucht, für ihn bestimmte Ziele zu erreichen. Ein *Kontextsystem* wird dabei von der Anwendung benutzt, um Aktivitäten durchführen zu lassen, die in Verbindung mit Kontext stehen. Ebenso wird ein *Prognosesystem* entweder vom Kontextsystem oder von der Anwendung direkt benutzt, um Prognoseaktivitäten durchführen zu lassen. Die Entwicklung von Anwendungen erfolgt durch *Anwendungsentwickler*.

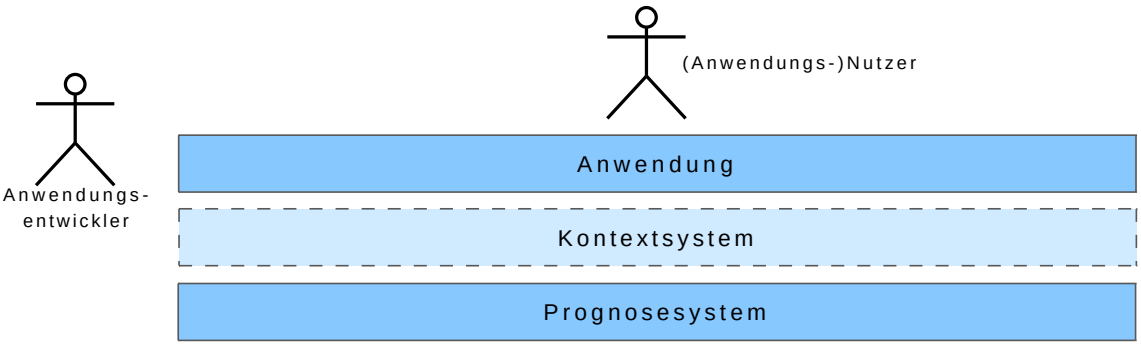

Abbildung 2.3.: Schichtenarchitektur, die dieser Arbeit zugrunde liegt

Abbildung 2.3 zeigt die Rollen in Form von Schichten, wobei das Prognose- und das Kontextsystem nicht unbedingt in eigenen Prozessen laufen müssen, sondern auch als Bibliotheken realisiert sein können, auf die die Anwendung zugreift.

Ein Beispiel für ein weiteres Kontextsystem ist der Kontextdienst der DEMAC-Middleware. Diese wird im Rahmen des DEMAC-Projektes [Kun05] entwickelt und setzt die Konzepte kontextbasierter Kooperation und mobiler Prozesse [Kun08] um, die die Ausführung langlebiger, komplexer Aufgaben in mobilen Systemumgebungen unterstützen [Kun08, S. III]. Eine Aufgabe wird dabei als Prozess repräsentiert, der sich aus auszuführenden Diensten zusammensetzt [Kun08, S. 141-142]. Um den Prozess erfolgreich ausführen zu können, werden Teile des Prozesses durch dessen Migration auf verschiedenen Geräten ausgeführt [Kun08, S. 141-142]. Dadurch werden die unterschiedlichen Potentiale der beteiligten Geräte genutzt, die sich aus der Heterogenität mobiler Geräte ergeben [Kun08, S. 141, 263]. Für das Migrieren werden die Kontexte der Geräte als Ausführungskontexte berücksichtigt [Kun08, S. 166].

Die auf Kontext basierende Kooperation benötigt letztendlich eine Planung, die vom Kontext der Geräte als Informationen ausgeht und die Migration des Prozesses steuert. Da für kontextbasierte Kooperation eine hochdynamische Umgebung vorausgesetzt wird [Kun08, S. 139], die sich insbesondere aus der Mobilität ergibt ↑, sind Informationen über den momentanen Kontext nur von bedingtem Nutzen für die Planung sowie für andere Nutzungsmöglichkeiten von Kontextdaten. Daher bietet sich für die DEMAC-Middleware neben einem Kontextsystem auch ein Prognosesystem wie das im Rahmen dieser Arbeit zu entwickelnde als zusätzliche Systemunterstützung an. Die Unterstützung von Planungen und die Rekonfiguration von Systemen werden auch im Allgemeinen als Einsatzgebiete für Kontextdatenprognose angesehen [May05, S. 32] [NMF05, S. 163-164]. Im Rahmen dieser Arbeit ist eine Umsetzung der Prognose der Verfügbarkeit von Diensten für einzelne Geräte zum Einsatz in der DEMAC-Middleware vorgesehen ↑. Dadurch können Informationen darüber gewonnen werden, ob der Kontext eines Gerätes in der Zukunft die Voraussetzungen zur erfolgreichen Ausführung eines Teils des Prozesses erfüllt. Weitere Möglichkeiten zur Unterstützung der Planung liegen beispielsweise in der Prognose darüber, ob der Prozess auf einem bestimmten Gerät durch einen Fehler scheitern wird, und über nicht-funktionale Rahmenbedingungen [Kun08, S. 6] wie z.B. die Ausführungsdauer auf einem Gerät.

Zu den Anforderungen an ein Prognosesystem für die DEMAC-Middleware, die auch in dieser Arbeit berücksichtigt werden, zählen die Eignung für das Ubiquitous Computing sowie die Unterstützung von Verteilung und Mobilität als Konzepte, auf denen kontextbasierte Kooperation aufbaut [Kun08, S. 5, 19, 266]. Auch eine gute Unterstützung der hohen Dynamik ist von Bedeutung. Zudem ist eine hohe Generik des Prognosesystems von Nutzen, um auch prozessspezifische bzw. aufgabenspezifische Prognosen durchführen zu können. Diese Anforderung ergibt sich daraus, dass das Konzept kontextbasierter Kooperation auf Aufgaben abzielt, die vom Nutzer stammen [Kun08, S. III, 132-133], so dass sich Prozesse und dafür benötigte Prognosen stark unterscheiden können. Es folgt abschließend ein Beispiel für eine prozessspezifische Prognose, das auch schon einen ersten Eindruck vom Thema der Prognose vermittelt, dem sich Abschnitt 2.4 widmet.

Beispiel 2.7:
Ein Versicherungsunternehmen beschäftigt eine Reihe von Vertretern, die zu Interessenten fahren, um vor Ort den Abschluss von Versicherungen abzuwickeln. Eine Person wird als Interessent registriert, wenn sie einen Termin vereinbart. Dabei gibt sie erste persönliche Daten an und, für welche Art von Versicherung sie sich interessiert, sowie Wünsche bezüglich Leistungsumfang der Versicherung. Bei der Registrierung einer Person als Interessent wird ein Prozess gestartet. Der entsprechende Prozess beschreibt die Abläufe von der Registrierung als Interessent bis zum Abschluss der Versicherung. Wenn ein Versicherungsvertreter einen Interessenten besucht, migriert der Prozess auf ein mobiles Gerät, das er bei sich trägt. Der Prozess enthält Aktivitäten wie das Erheben persönlicher Daten, die Festlegung von Parametern bezüglich Leistungsumfang und das Drucken des Vertrages. Das Aushandeln der Parameter erfolgt mittels eines entsprechenden Tools auf dem Gerät des Vertreters. Für das Erheben der persönlichen Daten und das Drucken des Vertrages werden vom Smart Space der Wohnung des Interessenten angebotene Dienste genutzt.

Zur Planung der Termine der Versicherungsvertreters ist eine Prognose in den Prozess integriert, die vorhersagt, wie lange sich ein Vertreter bei einem Interessenten aufhält, und dazu die Daten benutzt, die beim Registrieren des Interessenten angegeben werden. Voraussetzung für die Durchführung der Prognose ist Wissen darüber, wie die Dauer des Aufenthalts von der gewünschten Versicherung und persönlichen Daten des Interessenten abhängt. Dieses Wissen kann entweder schon bei der Einrichtung der Prognose angegeben oder automatisch erlernt werden. Die Einrichtung der Prognose erfolgt zusammen mit der Modellierung des Prozesses. Die Prognose selbst erfolgt dann, wenn ein Interessent registriert wird. Da im Allgemeinen bei prozessspezifischen Prognosen nicht feststeht, auf welchem Gerät sie ausgeführt werden, muss das für die Prognose verwendete Wissen zusammen mit dem Prozess migrieren.

2.4. Prognose

Ziel dieser Arbeit ist die Realisierung eines Prognosesystems. Dieser Abschnitt beschäftigt sich mit Prognose im Allgemeinen. Es werden die Fragen beantwortet, was Prognose ist, welche Arten es gibt, wie sie prinzipiell funktioniert und realisiert werden kann, worauf sie basiert und wo ihre Möglichkeiten und Grenzen liegen. Dabei wird eine Perspektive eingenommen, die recht universell ist, so dass möglichst keine Methoden zur Realisierung von Prognose aus dem Blickfeld verloren gehen. Die Verbindung zwischen dem Thema der Prognose und den Inhalten der vorhergehenden Abschnitte dieses Kapitels wird erst in Abschnitt 2.5 über Kontextdatenprognose hergestellt.

2.4.1. Begriff der Prognose

Es existieren verschiedene Verständnisse von Prognose. Diese Arbeit unterscheidet zwischen Zukunftsprognose und Folgerungsprognose.

Definition 2.5 (Zukunftsprognose):
Zukunftsprognose ist das Gewinnen von bestimmten gesuchten Informationen über die Zukunft unter Rückgriff auf gegebene Informationen über Gegenwart und Vergangenheit.

Definition 2.6 (Folgerungsprognose):
Folgerungsprognose ist das Gewinnen von bestimmten gesuchten Informationen aus gegebenen Informationen.

Der Begriff der Prognose wird in der Literatur in beiden Bedeutungen verwendet [SHD07, S. 31-32]. In der Bedeutung der Zukunftsprognose wird der Begriff Prognose (bzw. prediction oder forecasting) beispielsweise verwendet in [Kri99] und [SZ04, siehe S. 5]. In der Bedeutung der Folgerungsprognose verwenden den Begriff Prognose z.B. [SGS00, siehe S. 157-158], [Bra07a, siehe S. 4] und [VGS05, siehe S. 2]. Vovk et al. bringen die Auffassung von Prognose als Folgerungsprognose auf den Punkt: „Recognition, diagnosis, and estimation can all be thought of as special cases of prediction. A person or a computer is given certain information and asked to predict the answer to a question.“ [VGS05, S. 2]

Die Beantwortung der Frage, wie viel der Besitzer eines Mobiltelefons in den nächsten 24 Stunden telefonieren wird, stellt eine Zukunftsprognose dar (vgl. Beispiel 1.2). Eine Folgerungsprognose erfolgt z.B. bei der Schlussfolgerung, dass das Bild eines Fernsehers heller erscheint, wenn die Raumbeleuchtung gedimmt wird ↑, ebenso wie bei der Ermittlung der Ursache für eine Verkehrssituation mit starkem Verkehr ↑ und bei der Bestimmung von Kaufvorlieben einer Person aus ihren persönlichen Daten durch ein Unternehmen (vgl. auch [Bra07a, S. 23]). Die Zukunft spielt bei Folgerungsprognose nur

insofern eine Rolle, dass in der Gegenwart die Korrektheit der Prognose in der Regel noch nicht bekannt ist.

Folgerungsprognose ist nicht nur im Sinne von logischen Schlussfolgerungen zu verstehen und nicht nur auf sichere Schlussfolgerungen beschränkt. Der Begriff der Folgerungsprognose ist sehr weit gefasst. Viele Methoden der Folgerungsprognose wie z.B. neuronale Netze können für weite Aufgabenbereiche eingesetzt werden. Die beiden Verständnisse von Prognose sind letztendlich auch durch die unterschiedlichen Methoden zur Zukunfts- und zur Folgerungsprognose motiviert und werden in Kapitel 3 eine Rolle spielen. Da die gegebenen und gesuchten Informationen bei Folgerungsprognose keinen zeitlichen Restriktionen unterliegen, können Methoden der Folgerungsprognose auch für Zukunftsprognosen, mit denen sich diese Arbeit laut Zielsetzung beschäftigt, verwendet werden. Im Folgenden ist der Begriff Prognose im Sinne von Zukunftsprognose zu verstehen.

2.4.2. Prinzip der Prognose

Eine wichtige Frage besteht darin, nach welchem allgemeinen Prinzip Prognose überhaupt möglich ist. Man kann es mit dem Konzept der *Deduktion* erklären. Das Prinzip liegt darin, dass man Vorwissen einsetzt [SM05, S. 60]. Wenn man z.B. über das Vorwissen verfügt, dass eine bestimmte Person während der Arbeit keine Anrufe auf ihrem privaten Mobiltelefon annimmt, kann man beim Klingeln auf der Arbeit prognostizieren, dass die Person nicht abnehmen wird. Es ist wohl offensichtlich, dass man nur Prognosen über ein System machen kann, wenn man Wissen über das System besitzt. Die einzige Alternative zu Vorwissen besteht vermutlich in dem von Hübner genannten Ansatz, durch „Ausprobieren“ zu Aussagen zu kommen [Hüb03, S. 3], also den Zustand des Systems, über das Prognosen gemacht werden sollen, auf ein gleichartiges System zu übertragen und dessen Verhalten zu beobachten. Vorwissen schließt auch Wissen über Selbstverständlichkeiten ein, z.B. dass Gegenstände nach unten fallen. Man muss direkt oder indirekt etwas darüber wissen, wie der aktuelle Zustand und die vergangenen Zustände des Systems in zukünftige überführt werden. *Vorwissen* entspricht einem Abbild der Realität. In dieser Bedeutung wird auch häufig der Begriff Modell verwendet [KB08, S. 18], der in dieser Arbeit aber schon in anderer Bedeutung Benutzung findet (vgl. Abschnitt 2.3.4). Die Form von Vorwissen ist nicht festgelegt. Es kann sowohl im menschlichen Gehirn als auch in Computern repräsentiert werden, muss nicht komplex sein und ist nicht auf eine Darstellung in Form von Daten oder von Abläufen durch Programmcode beschränkt. Vorwissen kann z.B. auch als Realisierung einer Funktion zwischen Ein- und Ausgabewerten interpretiert werden [Wit02, S. 59]. Hübner äußert sich sogar so, dass für jedes komplexe Problem der realen Welt ein mathematisches Modell nötig sei, um eine quantitative Lösung ohne direkte Beobachtung zu erhalten [Hüb03, S. 3], wobei der Begriff des Modells in diesem Fall Vorwissen entspricht. Vorwissen kann abstrakter oder weniger abstrakt sein. Eine besondere, wenig abstrakte Art von Vorwissen liegt bei einer Simulation vor [Hüb03, S. 163].

Man kann sich das Prinzip der Prognose auch anhand des Mobiltelefons aus der Einleitung, das das Telefonierverhalten des Besitzers prognostiziert, veranschaulichen (vgl.

Gegebene Informationen	Gesuchte Informationen
"ekr"	ist ein Stringdenotierer
lskdf"	ist kein Stringdenotierer
"lsk"df"	ist kein Stringdenotierer

Tabelle 2.1.: Beispiel für Trainingsdaten beim induktiven Lernen [Pre99, S. 37]

Beispiel 1.2). Wenn man z.B. annimmt, dass der Besitzer häufig nach der Arbeit telefoniert, dann kann man prognostizieren, dass er mit einer gewissen Sicherheit zu der entsprechenden Zeit an einem bestimmten Tag auch telefonieren wird. Das Vorwissen besteht dann in dem Wissen darüber, dass der Nutzer häufig nach der Arbeit telefoniert. Das Vorwissen drückt Zusammenhänge aus, nach denen sich das betrachtete System, in diesem Fall der Nutzer und seine Umgebung verhalten. Solche Zusammenhänge sind es, auf denen Prognose aufbauen kann.

Trotz Zusammenhängen ist das Prognostizieren im Allgemeinen keine leichte Aufgabe, wie man schon an dem Beispiel des Mobiltelefons erkennt, denn es sind keine sicheren Aussagen über die Zukunft möglich. Im Alltag sieht man das z.B. an der Wettervorhersage. Im Bereich des Wetters sind Prognosen mit einer erheblichen Unsicherheit behaftet. Im Fall von Unsicherheit genügt für viele Anwendungen keine Prognose der Art, dass es regnen wird, denn es kann mit nicht viel geringerer Sicherheit auch sonnig werden oder schneien. Deshalb ist es für viele Anwendungen eine wichtige Anforderung, dass ein Prognoseverfahren mit Unsicherheit umgehen kann, indem es z.B. Auskunft über mehrere Szenarien gibt, die eintreten können.

2.4.3. Lernen

Abschnitt 2.4.2 zeigt, dass durch Deduktion und Anwendung von Vorwissen Prognosen möglich sind. Offen geblieben ist, wie man das nötige Vorwissen erhält. Es besteht die Möglichkeit, dass es Anwendungsexperten gibt, die das Vorwissen schon besitzen [Jen01, S. 79]. Man wird allerdings das konkrete System, über das Prognosen gestellt werden sollen, nicht immer kennen. Insbesondere kann es vorkommen, dass man beim Entwickeln von Prognosesoftware nicht alle Systeme kennt, über die unterschiedliche Nutzer zur Laufzeit Prognosen benötigen. Hinzu kommt, dass Menschen nicht jede Art von Wissen und Fähigkeiten, die sie besitzen, leicht in explizites Vorwissen überführen können. Vovk et al. stellen dies für den Fall des Schreibens von Programmcode fest [VGS05, S. 1]. Bei Wittig ist die Rede von „schwierig handhabbaren Domänen“ [Wit02, S. 60]. Die übliche Lösung für diese Probleme stellt *Lernen* dar. Als Ergebnis des Lernens erhält man Vorwissen [Wit02, S. 58]. Der entscheidende Vorteil von lernender Software ist nach Vovk Adaptionsfähigkeit [VGS05, S. 1]. Auch bei Wittig wird dieser Vorteil genannt [Wit02, S. 60]. Prognosen über Systeme, die erst zur Laufzeit bekannt sind, werden mit gesammelten Trainingsdaten verbessert und dadurch vielleicht überhaupt erst möglich. Deshalb sollte Prognosesoftware beim Vorliegen obiger Probleme lernfähig sein.

Die wichtigste Art von Lernen ist *induktives Lernen*. Anschaulich formuliert bedeutet induktives Lernen, dass die Realität beobachtet und aus den Beobachtungen gelernt wird.

Zusammenhänge, die die Geschehnisse in der Realität erklären, werden aus den Beobachtungen extrahiert und können später für Prognosen verwendet werden. Menschen lernen jeden Tag auf diese Weise. Sie sammeln jeden Tag neue Erfahrungen. Induktion bedeutet, dass aus Prognose-Datensätzen, in denen neben den gegebenen Informationen auch schon die korrekten Prognoseergebnisse, also die korrekten gesuchten Informationen vorgegeben sind, Vorwissen abgeleitet wird [SM05, S. 61]. Man benötigt mehrere solcher Sätze von Paaren gegebener und gesuchter Informationen. Tabelle 2.1 zeigt ein Beispiel im Zusammenhang mit Folgerungsprognose, bei der induktives Lernen nach dem gleichen Prinzip abläuft. Wenn in den Sätzen nicht explizit festgelegt ist, welche Informationen gesucht sind, spricht man von *unüberwachtem Lernen* (unsupervised learning), bei dem ohne festes Ziel möglichst viel gelernt werden soll [Bra07a, S. 4]. Ansonsten spricht man von *überwachtem Lernen* (supervised learning) [Bra07a, S. 4]. Unüberwachtes Lernen muss nicht, wie das Wort suggeriert, ohne menschliche Kontrolle erfolgen. Man nennt die Informationen, aus denen gelernt wird, häufig auch *Trainingsdaten* [Wit02, S. 59] [VGS05, S. 6]. Die Psychologie kennt neben induktivem noch *deduktives Lernen* [Pre99, S. 36]. Bei deduktivem Lernen wird das zu erlernende Vorwissen aus einer Formulierung des Vorwissens in einer bekannten Sprache [Pre99, S. 36-37] statt aus Trainingsdaten gelernt. Deduktives Lernen eignet sich also zum Austausch von Vorwissen. Das Gebiet des maschinellen Lernens kennt noch weitere Formen von Lernen, die aber wegen ihrer Spezialisierung in dieser Arbeit nicht berücksichtigt werden (vgl. [Bis06, S. 3], [Min90]).

Ein Nachteil von (induktivem) Lernen liegt darin, dass erst nach dem Lernen von genügend Trainingsdaten gute Prognosen möglich sind. Erst dann stellt sich eine ausreichende *Generalisierung* ein, die sich von den spezifischen Zusammenhängen in einer kleinen Menge an Trainingsdaten löst [Wit02, S. 59]. Bei einer zu geringen Generalisierungsfähigkeit spricht man von *Overfitting* [Wit02, S. 59]. Das bedeutet, dass in Situationen, die durch die Trainingsdaten trainiert wurden, gute Prognosen möglich sind, aber die durch die Trainingsdaten gelernten Zusammenhänge nicht so verallgemeinert werden, dass sie auch in anderen Situationen zu guten Prognosen führen. Unterschiedliche Lernmethoden generalisieren beim Training mit geringen Mengen an Trainingsdaten unterschiedlich gut [Wit02, S. 63-64].

Abbildung 2.4 zeigt die Vorgänge der Prognose und des Lernens mit den entsprechenden Datenflüssen. Im Folgenden wird statt Deduktion auch der Begriff Prognose verwendet, da Prognose ein so allgemeines Prinzip darstellt, dass für diese Arbeit keine Alternative zur Deduktion im Sinne der Nutzung von Vorwissen erkennbar ist (vgl. Abschnitt 2.4.2). Der Begriff Lernen wird im Sinne von induktivem Lernen verwendet.

Im Fall der Zukunftsprognose bestehen die Trainingsdaten zum Lernen aus Daten über das betrachtete System im zeitlichen Verlauf. Je nach Lernmethode können die gegebenen Informationen bestimmte Daten aus einem Zeitraum und die gesuchten Informationen bestimmte Daten aus einem späteren Zeitraum sein. Es ist aber auch möglich, dass allgemein aus den Daten im zeitlichen Verlauf gelernt wird, ohne dass es klar abgrenzbare Datensätze gibt. Wenn nicht angegeben ist, welches die gesuchten Informationen sind, handelt es sich um unüberwachtes Lernen. Bei der Prognose wird dann das gelernte Vorwissen verwendet, um aus bestimmten Informationen über die meist jüngere Vergan-

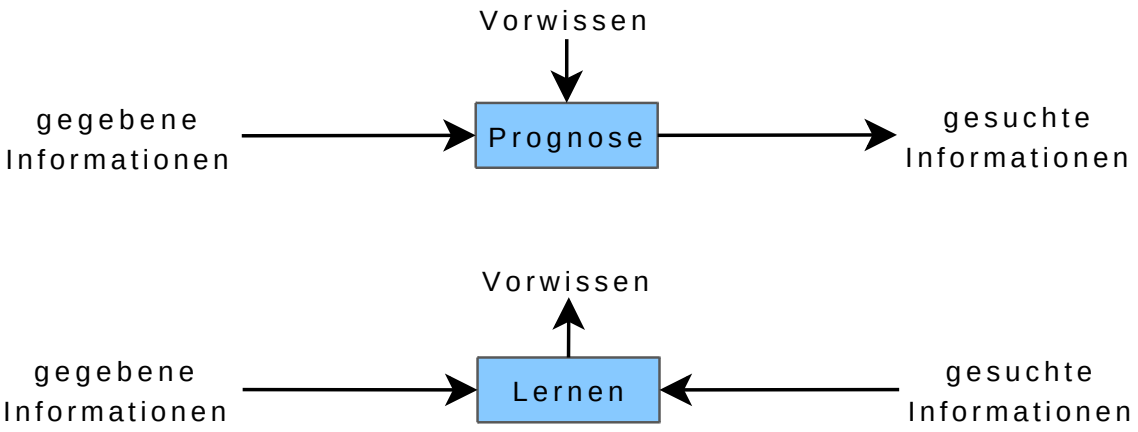

Abbildung 2.4.: Prognose/Deduktion (oben) und induktives Lernen (unten)

genheit und die Gegenwart bestimmte Informationen über die Zukunft zu gewinnen.

Zusammen mit obiger Unterscheidung zwischen überwachtem und unüberwachtem Lernen existieren die folgenden drei wichtigen Unterscheidungsmerkmale für Lernarten.

- überwacht oder unüberwacht
- batch oder adaptiv
- offline oder online

Beim Batchlernen wird eine potentiell große Menge an Trainingsdaten in einem Arbeitsschritt verarbeitet [Wit02, S. 59]. Batchlernmethoden besitzen oft eine hohe Komplexität [Wit02, S. 59]. Adaption bezeichnet die sequentielle Aktualisierung von vorhandenem Vorwissen [Wit02, S. 59]. Unter Online-Lernen ist zu verstehen, dass das Lernen zur Laufzeit erfolgt [Wit02, S. 59]. Offline-Lernen bedeutet entsprechend, dass vor der eigentlichen Laufzeit gelernt wird [Wit02, S. 59]. Häufig verhält es sich so, dass Batchlernen offline und Adaptionslernen online eingesetzt wird [Wit02, S. 59]. Der Online-Einsatz von Adaptionslernen bietet sich an, wenn die Trainingsdaten beim Nutzer gesammelt werden müssen und eine Lernphase zur Laufzeit vor der eigentlichen Nutzung vermieden werden soll.

Beim Lernen kann unterschiedlich stark abstrahiert werden. Es existieren auch Methoden, die die kompletten Trainingsdaten als Vorwissen übernehmen und erst bei der Prognose verarbeiten (z.B. nearest neighbour classifier, vgl. [Bra07a, S. 31-38]). Das allgemeine Konzept der *Transduktion* beschreibt ein Vorgehen dieser Art. Transduktion geht nicht explizit die Schritte der Induktion und der Deduktion, sondern stellt durch direkten Zugriff auf die Trainingsdaten eine Prognose [VGS05, S. 6]. Es wird also nicht von den Trainingsdaten abstrahiert bzw. gar nicht gelernt. Für Zukunftsprognose würde das bedeuten, dass bei jeder Prognose potentiell auf die kompletten Daten der Vergangenheit zurückgegriffen wird und dass kein abstraktes, aus der kompletten Vergangenheit extrahiertes Vorwissen existiert.

Der *Grad der Abstraktion* beim Lernen ist ein wichtiger Parameter für die Effizienz. Unter Abstraktion wird in der Regel in der Informatik verstanden, dass Informationen fallen

gelassen und nicht mehr betrachtet werden. Eine Erhöhung der Abstraktion führt also zu einem potentiell geringeren Speicherbedarf. Daher ist Abstraktion als wesentliche Funktion von Lernen anzusehen. Abstraktion von wichtigen Informationen kann jedoch auch die Prognosegenauigkeit verschlechtern. Daher sollte die Abstraktion darauf abgestimmt werden, welche Informationen für die benötigten Prognosen wichtig sind.

Der Rechenaufwand bei Prognose und Lernen hängt von der *Verteilung der Berechnungen* auf Lernen und Prognose ab. Wenn man ausgehend von Transduktion einen Prognoseschritt und einen separaten Lernschritt einführen möchte, hat man die Möglichkeit, Berechnungen auf diese beiden Schritte zu verteilen. Dabei wird auch der Rechenaufwand verteilt. Wie man den Rechenaufwand minimieren kann, der insgesamt im Betrieb entsteht, hängt von verschiedenen Faktoren ab. Wenn man offline lernt, lohnt es sich, beim Lernen möglichst viel Vorarbeit zu leisten. Wenn man online lernt, hängt es von der Anzahl benötigter Prognosen und der Menge der Trainingsdaten ab. Eine große Menge an Trainingsdaten spricht eher dafür, den Berechnungsaufwand pro Datensatz gering zu halten, also Berechnungen zur Prognose zu verlagern. Eine hohe Anzahl benötigter Prognosen spricht eher dafür, den Berechnungsaufwand pro Prognose gering zu halten, also Berechnungen zum Lernen zu verlagern. Neben dem Rechenaufwand insgesamt spielt aber auch die benötigte Zeit für eine Prognose insbesondere bei interaktiven Systemen eine Rolle. Das spricht wiederum dafür, mehr Berechnungsaufwand zum Lernen zu verlagern.

Die Verteilung der Berechnungen auf Lernen und Prognose beeinflusst auch den Speicherplatzbedarf. Wenn alle Berechnungen bei der Prognose erfolgen, besteht keine Möglichkeit zur Abstraktion von den Trainingsdaten vor deren Speicherung. Wenn nahezu alle Berechnungen beim Lernen erfolgen, muss das Vorwissen, das beim Lernen gewonnen wird, schon Vorstufen der Prognoseergebnisse für alle möglichen Prognosen beinhalten, die überhaupt gestellt werden können. In diesen beiden Extremfällen ist also in der Regel mit relativ hohem Speicherplatzbedarf zu rechen. Eine Ausnahme bildet der Sonderfall, dass es nur wenige mögliche Prognosen gibt, die gestellt können werden müssen.

In den folgenden Abschnitten wird überwiegend die Unterscheidung zwischen Prognose und Lernen fallen gelassen und Lernen als impliziter Teil von Prognose angesehen. Dies ist begrifflich unproblematisch, da die Eingaben eines Lernschrittes als gegebene Informationen von Prognose aufgefasst werden können (vgl. Definition 2.5). Diese Sichtweise ist eher theoretisch motiviert und vereinfacht die folgenden Abschnitte.

2.4.4. Historische Daten

Bei Zukunftsprognose erfolgt sowohl das Prognostizieren als auch das Lernen auf Basis von Daten im zeitlichen Verlauf, also *historischen Daten*. Durch das Hinzuzählen des Lernens zur Prognose kann Prognose als Fortschreiben von historischen Daten angesehen werden.

Für weitere Überlegungen sollte definiert werden, was unter historischen Daten zu verstehen ist. Dabei orientiert sich die folgende Definition historischer Daten (Definition 2.7) an üblichen Definitionen von *Zeitreihen* (und entfernter auch an der Definition von stochastischen Prozessen). Zeitreihen beschreiben Daten im zeitlichen Verlauf als Folge

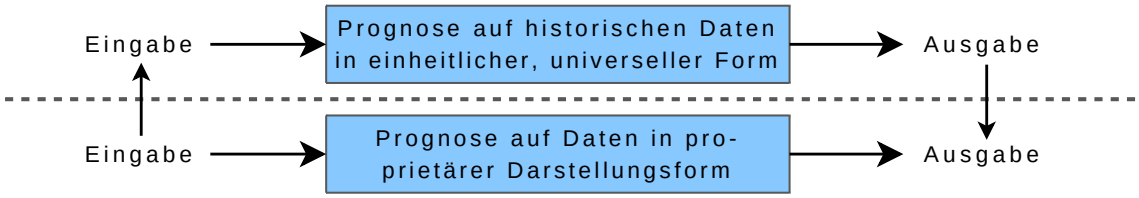

Abbildung 2.5.: Übertragbarkeit einer Prognose mit Daten in einem proprietären Format auf eine Prognose mit Daten in einem universellen Format durch Umwandlung der Daten (senkrechte Pfeile), orientiert an [Jän04, S. 84]

von (ein- oder mehrdimensionalen) Werten [ABH85, S. 7-8] [Kri99, S. 5] (vgl. [Hüb03, S. 123] zu stochastischen Prozessen).

Man beschreibt das betrachtete System zu einem bestimmten Zeitpunkt also durch einen Wert, den man als Zustand des Systems auffassen kann, wobei der Zustand im Grunde Element eines beliebigen Vektorraumes sein darf, man aber häufig die Darstellung im isomorphen Standardvektorraum $\mathbb{K}^n$ [Jän04, S. 86] wählen wird. Man kann sich also unter einem mehrdimensionalen Wert als Zustand ein Tupel von eindimensionalen Werten (Skalaren) vorstellen, die nicht notwendigerweise Zahlen sein müssen.

Zeitreihen stammen aus dem Gebiet der Zeitreihenanalyse, das im Gegensatz zu anderen überprüften Gebieten, die im Bezug zu Prognose stehen, eine präzise festgelegte Sichtweise auf Daten im zeitlichen Verlauf anbietet. Es wird im Folgenden begründet, warum eine solche Sichtweise allgemein genug ist, so dass keine (benötigten) Anwendungsfelder und keine Methoden zur Realisierung von Prognose aus dem Blickfeld verloren werden.

Die Darstellung von Daten im zeitlichen Verlauf als Folge von Zuständen lässt sich dadurch rechtfertigen, dass man andere Darstellungen aufgrund der Mächtigkeit der Zustandssichtweise in Zeitreihen überführen kann. Wenn die Darstellung des Verhaltens des betrachteten Systems durch einen der mächtigen Formalismen der theoretischen Informatik möglich ist, dann kann man sich die in der Regel vorhandene Möglichkeit der Überführung in einen Zustandsgraphen zunutze machen. Die Möglichkeit besteht z.B. bei Petrinetzen und Prozessalgebra [Val07, S. 28] [Val07, S. 142]. Ein so gewonnener Zustandsgraph stellt dann ebenfalls das Verhalten des Systems dar. Ein konkreter Ablauf des Systemverhaltens kann dann als Folge von Zuständen im Zustandsgraphen, also als Zeitreihe ausgedrückt werden. Insgesamt kann man dann bei einer Prognose durch eine konkrete Methode die Daten im zeitlichen Verlauf, die diese Methode als Eingabe entgegen nimmt, in eine Zeitreihe überführen, die Prognose auf einer allgemeinen Ebene auf Basis von Zeitreihen betrachten und das auf der allgemeinen Ebene erhaltene Prognoseergebnis im Idealfall sogar wieder in die Darstellung überführen, die die konkrete Methode erzeugt (vgl. Abbildung 2.5).

Auch die Tatsache, dass man bei Zeitreihen den zeitlichen Verlauf durch diskrete Zeitpunkte ausdrückt, die vielfach zusätzlich *äquidistant* sind, also einen konstanten Abstand aufweisen [ABH85, S. 7-8] [Kri99, S. 5] [Hüb03, S. 123], stellt keine wesentliche Einschränkung dar. Stetige Zeitreihen können durch diskrete genähert werden [Hüb03, S. 123] [Kri99, S. 5].

Die Sichtweise von Zeitreihen ist also als universell anzusehen und eignet sich für die folgende Definition historischer Daten. Neben allgemeinen Betrachtungen zum Thema Prognose in den nachfolgenden Abschnitten kann eine einheitliche, universelle Darstellung historischer Daten auch der Beschreibung und dem Vergleich bestehender Methoden zur Realisierung von Prognose im nachfolgenden Kapitel sowie der Beschreibung eigener Ansätze dienen. Deshalb zielt die nachfolgende Definition historischer Daten darauf ab, dass die Betrachtung unterschiedlicher Methoden unter Rückgriff auf die definierte Notation nicht nur prinzipiell möglich, sondern auch in der Praxis einfach handhabbar ist. Dies soll erreicht werden, indem die besonders bei der Zeitreihenanalyse üblichen mehrdimensionalen Zustände explizit in ihre einzelnen Dimensionen zerlegt und als Werte entsprechender Variablen angesehen werden.

Definition 2.7 (Historische Daten):
Seien $V_1, ..., V_n$ Variablen, die zu unterschiedlichen Zeitpunkten unterschiedliche Werte annehmen können. Es gelte für alle Variablen V_i mit $i \in \mathbb{N}$ und $1 \leq i \leq n$: V_i besitze den Wertebereich $D(V_i)$ mit $? \in D(V_i)$ als Symbol für einen unbekannten Wert. $\mathcal{V}_i$ beschreibe die Werte $v_{i,j}$ von V_i zu unterschiedlichen Zeitpunkten j.

$$\mathcal{V}_i = (v_{i,1}, ..., v_{i,m})$$

Die $v_{i,j} \in D(V_i)$ seien dabei beliebige Konstanten mit $j \in \mathbb{N}$, $1 \leq j \leq m$ und den Werten von j als Darstellung der Zeitpunkte. Dann bezeichnet $\mathcal{H}$ historische Daten über den Variablen $V_1, ..., V_n$ genau dann, wenn es die entsprechenden $\mathcal{V}_i$ in folgender Weise zusammenfasst.

$$\mathcal{H} = (\mathcal{V}_1, ..., \mathcal{V}_n)$$

Statt der Bezeichnungen V_1, V_2 und V_3 können auch X, Y und Z verwendet werden. Für die Werte $v_{1,j}$, $v_{2,j}$ und $v_{3,j}$ dieser Variablen kann auch x_j, y_j und z_j geschrieben werden. Wenn der Zeitpunkt j nicht von Bedeutung oder aus dem Zusammenhang klar ist, seien die Schreibweisen v_i sowie x, y und z für die Werte der Variablen V_i sowie X, Y und Z erlaubt.

Unbekannte Werte werden in der Definition explizit berücksichtigt wegen der möglichen Erschwernis durch diese bei Verarbeitungen der Daten [BK02, S. 133]. Abbildung 2.6 stellt ein abstraktes Beispiel historischer Daten grafisch dar.

Man kann sich historische Daten nach obiger Definition ebenso wie Zeitreihen auch als *Datenwürfel* [RP06, S. 975] vorstellen, der mit n Dimensionen die n Variablen V_1, ..., V_n und mit einer Dimension die Zeit darstellt.

Der weiteren Erläuterung der Definition dient das folgende Beispiel. Es enthält historische Daten, die Beispiel 1.2 über die Prognose des Telefonierverhaltens eines Mobiltelefonbesitzers aufgreifen. Die historischen Daten zeigen einen möglichen Ablauf des Telefonierverhaltens. Der Nutzer erhält einen Anruf auf der Arbeit, der Anruf wird als verpasst registriert und der Nutzer ruft auf dem Weg nach Hause zurück.

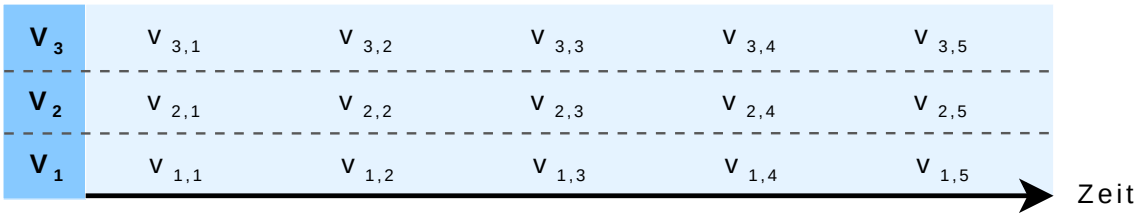

Abbildung 2.6.: grafische Darstellung historischer Daten über den Variablen V_1, V_2 und V_3 für fünf Zeitpunkte

Beispiel 2.8:
Es wird die Nutzung eines Mobiltelefons beobachtet (vgl. Beispiel 1.2). Dafür werden die drei Variablen V_1, V_2 und V_3 gewählt. Für jeden Zeitpunkt wird gespeichert, an welchem Ort sich das Gerät gerade befindet (V_1), wie viele verpasste Anrufe noch offen sind (V_2) und ob der Nutzer gerade telefoniert (V_3). Die Wertebereiche werden entsprechend festgelegt:

$$D(V_1) = \{Wohnung, Arbeit, Park, ?\} \quad D(V_2) = \{0, 1, 2, 3, ?\} \quad D(V_3) = \{false, true, ?\}$$

Die Variablen könnten zu einem bestimmten Zeitpunkt folgende Werte besitzen:

$$X = V_1 = Arbeit \qquad Y = V_2 = 0 \qquad Z = V_3 = false$$

Wenn bekannt ist, dass die Werte vom Zeitpunkt 8 stammen, kann geschrieben werden:

$$V_1 = v_{1,8} = Wohnung \qquad V_2 = v_{2,8} = 0 \qquad V_3 = v_{3,8} = false$$

Nun sei angenommen, dass die Werte an einem Tag von 16:00 bis 17:30 Uhr alle 15 Minuten ermittelt wurden.

$$\begin{array}{lll}
v_{1,1} = Arbeit & v_{2,1} = 0 & v_{3,1} = false \\
v_{1,2} = Arbeit & v_{2,2} = 0 & v_{3,2} = false \\
v_{1,3} = Arbeit & v_{2,3} = 1 & v_{3,3} = false \\
v_{1,4} = Arbeit & v_{2,4} = 1 & v_{3,4} = false \\
v_{1,5} = Arbeit & v_{2,5} = 1 & v_{3,5} = false \\
v_{1,6} = Park & v_{2,6} = 0 & v_{3,6} = true \\
v_{1,7} = Park & v_{2,7} = 0 & v_{3,7} = false
\end{array}$$

Es ergeben sich die formalen historischen Daten über V_1, V_2, V_3:

$$\begin{aligned}
\mathcal{V}_1 &= (Arbeit, Arbeit, Arbeit, Arbeit, Arbeit, Park, Park) \\
\mathcal{V}_2 &= (0, 0, 1, 1, 1, 0, 0) \\
\mathcal{V}_3 &= (false, false, false, false, false, true, false) \\
\mathcal{H} &= (\mathcal{V}_1, \mathcal{V}_2, \mathcal{V}_3)
\end{aligned}$$

Wie man sieht, können die Wertebereiche einer Variablen recht unterschiedlich sein. Die deskriptive Statistik, die sich mit der Darstellung empirischer Daten beschäftigt [Ste07, S. 1], unterscheidet verschiedene Arten von Wertebereichen. Die Statistik spricht von *Skalen* [Ste07, S. 6]. Es existieren folgende Arten von Skalen [Ste07, S. 6-7].

- Nominalskala
- Ordinalskala
- Metrische Skala
  - Intervallskala
  - Ratioskala

Bei einer *Nominalskala* stehen die Werte des Wertebereiches in keiner Ordnungsrelation zueinander. Operationen wie Subtraktion und Division sind nicht definiert. Der Aufenthaltsort in Beispiel 2.8 über historische Daten ist nominal skaliert. Eine *Ordinalskala* entspricht einer Nominalskala mit dem Unterschied, dass sie über eine Ordnungsrelation auf dem Wertebereich verfügt, so dass Werte verglichen werden können. Ein Beispiel für ordinale Skalierung ist der Schulabschluss einer Person. Man kann Schulabschlüsse zwar miteinander vergleichen (Hauptschulabschluss < Realschulabschluss < ...), aber die Differenz zwischen zwei Schulabschlüssen auszurechnen, ist nicht möglich. *Intervallskalen* überwinden diesen Nachteil und entsprechen ansonsten Ordinalskalen. Ein Beispiel für eine Intervallskala ist die Temperatur in °C, denn Temperaturen in °C können subtrahiert, aber nicht sinnvoll dividiert werden. *Ratioskalen* besitzen wiederum die gleichen Eigenschaften wie Invallskalen mit dem Unterschied, dass sie auch die Berechnung von Quotienten ermöglichen. In der Literatur findet man auch als konkretere Anforderung, dass die Axiome eines Körpers erfüllt sein müssen [May04, S. 45]. Die Anzahl verpasster Anrufe aus Beispiel 2.8 über historische Daten weist die Eigenschaften einer Ratioskala auf. *Metrische Skalen* werden in Intervallskalen und Ratioskalen unterteilt. Zusätzlich zu der Unterscheidung der Skalenniveaus kann zwischen diskreten und stetigen Werten unterschieden werden [May04, S. 45-46] [Sig08, S. 231]. [Ste07, S. 6-7]

Entsprechend dieser Ausführungen stehen die unterschiedlichen Skalentypen in einer Rangordnung bezüglich der Möglichkeiten beim Umgang mit den Daten. Nominalskalen bieten die geringsten Möglichkeiten zur Analyse und Verarbeitung und metrische Skalen die umfangreichsten. Deshalb ist es sinnvoll, die Möglichkeiten des Skalenniveaus der betrachteten Daten auch auszunutzen. Auf der anderen Seite sind Methoden, die die Möglichkeiten eines bestimmten Skalenniveaus nutzen, auch von diesem Skalenniveau abhängig und somit nicht grundsätzlich auf Daten mit niedrigerem Skalenniveau anwendbar. Die Anwendung von Methoden auf Daten mit höherem Skalenniveau ist allerdings immer möglich [Ste07, S. 7]. Das Skalenniveau der Daten kann dazu durch Vergröberung der Daten verringert werden [Ste07, S. 7]. Eine Methode, die nur die Möglichkeiten des geringsten Skalenniveaus nutzt, ist also auf alle Daten anwendbar, aber bei Daten mit höherem Skalenniveau nicht optimal, da es nicht die Möglichkeiten des Skalenniveaus ausnutzt. Eine Methode, die die Möglichkeiten eines höheren Skalenniveaus nutzt, ist zwar in dieser Hinsicht bei Daten des entsprechenden Skalenniveaus optimal,

kann jedoch normalerweise nicht einfach auf Daten mit niedrigerem Skalenniveau angewendet werden. Wenn man Daten mit unterschiedlichen Skalenniveaus möglichst gut unterstützen möchte, sollte man daher mehrere Methoden verwenden, die die Skalenniveaus direkt unterstützen, ohne dass eine Umwandlung der Daten nötig wird. Allerdings fordern viele Methoden entweder nur nominales Skalenniveau oder gleich metrisches [Sig08, S. 209].

2.4.5. Zusammenhänge und Unsicherheit im Überblick

Prognose lebt davon, dass das Verhalten des betrachteten Systems auf *Zusammenhängen* basiert, mit denen man dann das zukünftige Verhalten des Systems vorhersagen kann ↑. Auf der anderen Seite sind Prognosen in der Regel mit *Unsicherheit* behaftet ↑. Die nachfolgenden Abschnitte befassen sich vertieft mit Zusammenhängen und Unsicherheit. Sie zeigen Möglichkeiten und Grenzen von Prognose auf und arbeiten Arten von Zusammenhängen und Unsicherheit heraus, um mögliche Ansatzpunkte für Methoden zur Prognose zu zeigen und den Umgang mit Unsicherheit zu unterstützen.

Zeitreihen als Ausgangspunkt der Definition von historischen Daten werden in deterministische und stochastische Zeitreihen unterteilt [Kri99, S. 5]. Diese Unterteilung wird auf historische Daten übertragen. *Deterministische historische Daten* können durch eine Funktion beschrieben werden [Kri99, S. 5], sind also theoretisch sicher vorhersagbar, wenn man die Funktion genau ermitteln kann, was sich noch als recht schwierig erweisen wird. *Stochastische historische Daten* können nur durch Wahrscheinlichkeitsverteilungen beschrieben werden [Kri99, S. 5], so dass eine sichere Prognose von zukünftigen Systemzuständen des betrachteten Systems ausgeschlossen ist. Deterministische historische Daten erlauben jedoch auch nicht immer einfach durchführbare sichere Prognosen. Insbesondere deterministisches Chaos kann nämlich auch zu irregulärem Verhalten und Anomalien führen, ohne dass das System von Natur aus stochastisch zu sein scheint [KS97, S. i]. Im Allgemeinen kann die Ursache von Unsicherheit von sehr unterschiedlicher Art sein [Hüb03, S. 1]. Nach Hübner kann entweder eine *prinzipielle* oder eine *faktische Unmöglichkeit* einer exakten Aussage vorliegen [Hüb03, S. 1].

Die wahrscheinlich verbreitetste Möglichkeit zum Ausdruck von und zum Umgang mit Unsicherheit ist die Angabe von Wahrscheinlichkeiten unter Rückgriff auf die Stochastik. Die Stochastik befasst sich mit der Beschreibung und Untersuchung von Vorgängen und Ereignissen, die zufällig stattfinden im Sinne von „nicht (exakt) vorhersagbar“ [Hüb03, S. 1]. Die Stochastik ermöglicht die Quantifizierung von Unsicherheit durch Wahrscheinlichkeiten und stellt damit ein wichtiges Werkzeug dar. Insbesondere ermöglicht sie auch die Angabe von Eintrittswahrscheinlichkeiten verschiedener möglicher Zukunftsszenarien.

Das Gegenstück zu Unsicherheit sind *Zusammenhänge*. Unsicherheit erschwert Prognose und Zusammenhänge ermöglichen sie. Es wird im Folgenden der Begriff „Zusammenhänge“ definiert. Zusätzlich wird der Begriff „Zusammenhang im konkreteren Sinne“ festgelegt werden. Die beiden Begriffe sind nicht äquivalent, liegen aber im Kern recht nah beieinander und sind deshalb ähnlich benannt.

Definition 2.8 (Zusammenhänge):
Zusammenhänge sind die Gesetzmäßigkeiten, denen das Verhalten eines Systems unterliegt.

Diese Definition ist durch das Prinzip der Prognose motiviert. Vorwissen bildet Zusammenhänge ab. Prognose kann also auch als Ausnutzen von Zusammenhängen aufgefasst werden.

Definition 2.9 (Zusammenhang im konkreteren Sinne):
Ein *Zusammenhang im konkreteren Sinne* zwischen Teilen von historischen Daten besteht genau dann, wenn die Wahrscheinlichkeit, dass ein Teil der Daten bestimmte Werte annimmt, davon abhängig ist, welche Werte die anderen Teile besitzen.

Wenn z.B. der Mobiltelefonnutzer aus Beispiel 2.8 nach der Arbeit bei einem offenen verpassten Anruf ($v_{2,5} = 1$) mit hoher Wahrscheinlichkeit kurz darauf telefoniert ($v_{3,6} = true$), während er sich im Park befindet ($v_{1,6} = Park$), besteht ein Zusammenhang zwischen $v_{1,6}$, $v_{2,5}$ und $v_{3,6}$ als Teile der historischen Daten. Diese Definition ist unter Anderem durch die stochastische Unabhängigkeit und bedingte Wahrscheinlichkeiten motiviert [Hüb03, S. 27, 29-30].

Dass der Begriff „Zusammenhang im konkreteren Sinne“ eine ähnliche Mächtigkeit besitzt wie der Begriff „Zusammenhänge“, wird dadurch deutlich, dass Prognose als Gewinnen von gesuchten Informationen aus gegebenen Informationen definiert wurde (vgl. Definition 2.5). Wenn gegebene und gesuchte Informationen nicht in einem Zusammenhang im konkreteren Sinne stünden, wären sie voneinander unabhängig, ebenso wie z.B. das Fallen eines Blattes in Deutschland unabhängig davon ist, ob eine U-Bahn in den USA eine Minute zuvor gerade den Endbahnhof erreicht hat. Wenn man gegebene Informationen darüber besitzt, dass der Zug vor einer Minute den Endbahnhof erreicht hat, kann man trotzdem nicht die gesuchte Information gewinnen, ob das Blatt gerade fällt. Man kann höchstens das Vorwissen nutzen, dass Blätter im Allgemeinen nur selten vom Baum fallen. In Fällen, bei denen man sich mit Prognosen zufrieden gibt, die keiner gegebenen Informationen bedürfen, ist das Ergebnis bei gleichem Vorwissen immer das gleiche. Zusammenhänge im konkreteren Sinne spielen also, wenn man von einfachen Fällen absieht, immer eine Rolle. Statt von einem Zusammenhang im konkreteren Sinne wird im Folgenden manchmal auch einfach von einem Zusammenhang die Rede sein, wenn aus dem Kontext heraus erkennbar ist, dass es sich um einen Zusammenhang im konkreteren Sinne handelt.

2.4.6. Arten von Zusammenhängen

Dieser Abschnitt trägt verschiedene Arten von Zusammenhängen aus der Literatur zusammen. Arten von Zusammenhängen sind mögliche Ansatzpunkte für die Realisierung von Methoden zur Prognose, da man versuchen kann, eine oder mehrere Arten zu nutzen. Bei vielen der hier genannten Arten kann man auch den Bezug zu Zusammenhängen

im konkreteren Sinn erkennen, also dass Teile historischer Daten voneinander abhängig sind. Es können die folgenden Arten von Zusammenhängen identifiziert werden.

- Trends [ABH85, S. 8-9] [Sig08, S. 96] [May04, S. 66]
- Sequentielle Muster [May04, S. 66]
- Periodische Muster [ABH85, S. 9, „Saisonkomponente“] [Sig08, S. 96, „patterns that repeatedly occur“] [May04, S. 66]
- Deterministische Zusammenhänge [And71, S. 2, „systematic part“]
- Gemischte Zusammenhänge [And71, S. 2, „systematic part“]
- Lineare Zusammenhänge [KS97]
- Kausale Zusammenhänge [SGS00]
- Einzelnen Zeitpunkt betreffende Zusammenhänge

Die verwendeten Begriffe für einzelne Arten von Zusammenhängen variieren von Autor zu Autor. Es folgt eine Erklärung der Zusammenhangsarten.

Trends drücken langfristige Entwicklungen aus [ABH85, S. 8] [May04, S. 66]. Sie beziehen sich auf metrisch skalierte Werte [Sig08, S. 96]. *Sequentielle Muster* sind zeitliche Abfolgen, die sich wiederholen [May04, S. 66]. Wenn z.B. ein Telefon klingelt, besteht eine hohe Wahrscheinlichkeit, dass darauf folgend auch telefoniert wird. *Periodische Muster* wiederholen sich mit einer bestimmten Periode [May04, S. 66]. Ein Arbeitnehmer hält sich beispielsweise jeden Tag zur gleichen Zeit an seinem Arbeitsplatz auf. Dieser und einige andere periodische Zusammenhänge sind gleichzeitig Beispiele für (nahezu) *deterministische Zusammenhänge*. Insbesondere deterministische Zusammenhänge treten häufig nicht allein, sondern als Teil eines gemischten Zusammenhangs auf. Mit einem *gemischten Zusammenhang* ist gemeint, dass der Wert einer Variablen durch mehrere der hier genannten Zusammenhangsarten beeinflusst wird. Gemischte Zusammenhänge können z.B. deterministische und nichtdeterministische Zusammenhänge beinhalten [And71, S. 2] oder in der Ökonomie eine Trend-, Konjunktur- und (periodische) Saisonkomponente, deren Summe beispielsweise die Anzahl der Arbeitslosen ergeben kann [Ste07, S. 64, 66-67]. Lineare und nichtlineare Zusammenhänge sind hier nur in Bezug auf Variablen mit metrischem Skalenniveau, für die eine Multiplikation definiert ist, zu verstehen. *Lineare Zusammenhänge* basieren auf linearen Abhängigkeiten, z.B. $x_j = 0,5 \cdot x_{j-5} + 0,5 \cdot x_{j-6}$.

Ergänzend zur Literatur sind oben einen *einzelnen Zeitpunkt betreffende Zusammenhänge* genannt. Diese sind durch *kausale Zusammenhänge* motiviert. Beide Zusammenhangsarten beziehen sich auf Zusammenhänge im konkreteren Sinne des Wortes und können durch allgemeine Methoden der Folgerungsprognose genutzt werden. Ein Beispiel für beide Zusammenhangsarten besteht in dem Zusammenhang, dass sich jemand im Raum befindet (X), wenn das Licht eingeschaltet ist (Y). Der Wert der Variablen X ist zu einem bestimmten Zeitpunkt vom Wert der Variablen Y abhängig. Ein kausaler Zusammenhang liegt vor, weil der Aufenthalt einer Person im Raum die Ursache für das eingeschaltete Licht ist. Es handelt sich um einen Zusammenhang, der einen einzelnen Zeitpunkt betrifft, weil Werte beider Variablen zum gleichen Zeitpunkt betroffen sind. Einen einzelnen Zeitpunkt betreffende Zusammenhänge können zu jedem Zeitpunkt gleichermaßen

gelten, müssen es aber nicht. Es ist z.B. möglich, dass Urlauber bei sich zu Hause als Einbruchsicherung gelegentlich automatisch das Licht anschalten lassen.

In der Literatur werden außerdem Ereignisse bei der Suche nach Zusammenhangsarten herangezogen [Sig08, S. 96] [May05, S. 33]. Diesem Vorgehen wird aber nicht gefolgt, denn Ereignisse werden in dieser Arbeit nur als Basis einer speziellen Sichtweise aufgefasst, die nicht Zustände, sondern Zustandsübergänge in den Mittelpunkt stellt. Zustandsübergänge (Transitionen) werden im Allgemeinen durch Ereignisse ausgelöst [Dum]. Da es häufig nicht von Interesse ist, wie es zu einer Zustandsänderung kommt, wird die Unterscheidung zwischen Transitionen und Ereignissen in dieser Arbeit fallen gelassen und das Wort Ereignis als Synonym für eine Transition verwendet.

Interessant im Zusammenhang mit Ereignissen ist die Annahme, dass zeitlich nah beieinander liegende Ereignisse eher in Zusammenhang stehen als weit auseinander liegende [And71, S. 3], was sich auch von Ereignissen auf Variablenwerte / Zustände übertragen lässt, da Ereignisse Änderungen von Zuständen ausdrücken. Die *Bedeutung zeitlicher Nähe* für Zusammenhänge ist von großem Vorteil, weil sie das Löschen älterer historischer Daten bzw. älteren Vorwissens erlaubt und so das Einsparen von Speicherplatz ermöglicht. Allerdings gilt dieses Prinzip offensichtlich nicht im Allgemeinen. Bei periodischen Mustern mit großer Periode, z.B. Geburtstagen, die jährlich auftreten, gilt es beispielsweise nicht.

2.4.7. Wahrscheinlichkeitsmodell zu historischen Daten

Nach dieser Würdigung konkreter Arten von Zusammenhängen bleibt die Frage nach Zusammenhängen im Allgemeinen offen. Dieser Abschnitt trifft Vorbereitungen zur Beantwortung dieser Frage. Es beginnt an dieser Stelle ein eher theoretischer Gedankengang, der sich über mehrere Abschnitte erstreckt. Der Gedankengang wird von Zusammenhängen (Abschnitt 2.4.8) zu ihren Grenzen und damit zur Unsicherheit (Abschnitt 2.4.8, 2.4.9) kommen. Die theoretische Betrachtung von Unsicherheit wird dann wieder zum Konkreten führen, nämlich einem Vergleich verschiedener Arten von Unsicherheit (Abschnitt 2.4.10), die unterschiedlich gravierend und nicht alle bei konkreten Prognosen erkennbar und quantifizierbar sind. Die theoretische Betrachtung von Zusammenhängen und Unsicherheit verdeutlicht, was die theoretisch beste mögliche Prognose wäre, warum sie in der Regel nicht erreichbar ist und unter welchen Bedingungen man doch nah an das optimale Ergebnis kommen kann. Diese Überlegungen sind sowohl für die Realisierung von Methoden zur Prognose als auch zur Entwicklung von Ansätzen zur Evaluation solcher Methoden von Bedeutung.

Es werden zunächst kurz die notwendigen Grundlagen der Stochastik und der Statistik präsentiert. Es wird mit dem sehr allgemeinen Konzept von Wahrscheinlichkeitsmodellen begonnen, um damit das Entstehen historischer Daten über ein betrachtetes System zu beschreiben.

Ein Wahrscheinlichkeitsmodell beinhaltet verschiedene Ergebnisse, die mit bestimmten Wahrscheinlichkeiten eintreten können. Mengen von Ergebnissen bilden Ereignisse, die mit einer Wahrscheinlichkeit eintreten können, die sich aus der Wahrscheinlichkeit ihrer Ergebnisse ergibt.

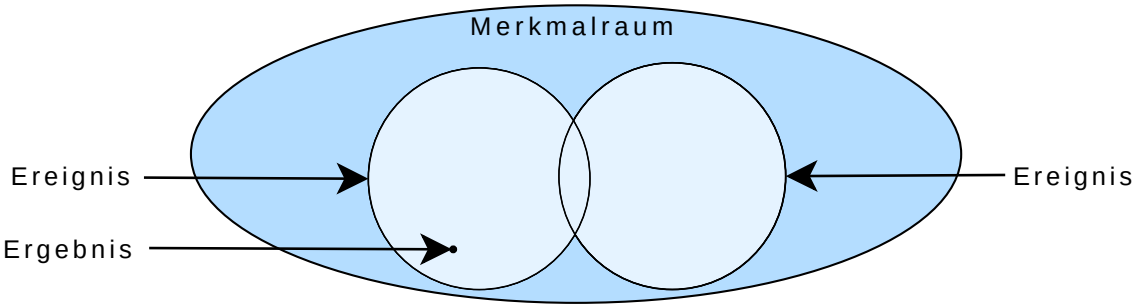

Abbildung 2.7.: Merkmalraum mit einem der Ergebnisse und zweien der Ereignisse eines Wahrscheinlichkeitsmodells als Venn-Diagramm

Definition 2.10 (Wahrscheinlichkeitsmodell):
Ein *Wahrscheinlichkeitsmodell* ist ein Tripel der Form $(\Omega, \mathcal{A}, P)$. Der *Merkmalraum* Ω ist eine nicht leere Menge mit Elementen $\omega \in \Omega$. Das *Ereignissystem* $\mathcal{A}$ ist eine Mengen-σ-Algebra über Ω. Insbesondere gilt $\mathcal{A} \subseteq 2^{\Omega}$. Die Abbildung $P : \mathcal{A} \to \mathbb{R}$ heißt *Wahrscheinlichkeitsmaß* auf $\mathcal{A}$ mit:

$P(A)$	$\geq$	0	für alle $A \in \mathcal{A}$
$P(\Omega)$	$=$	1	
$P(\bigcup_{i=1}^{\infty} A_i)$	$=$	$\sum_{i=1}^{\infty} P(A_i)$	für alle abzählbaren Mengen $\{A_1, A_2, ...\} \subseteq \mathcal{A}$ wobei alle A_i paarweise disjunkt sind

basierend auf [Hüb03, S. 11-24]

Der Merkmalraum Ω enthält sogenannte *Ergebnisse*. Es kann genau ein Ergebnis eintreten. Das Ereignissystem enthält Mengen von Ergebnissen, sogenannte *Ereignisse*. Man sagt, dass ein Ereignis eintritt, wenn eines der darin enthaltenen Ergebnisse eintritt. Ereignisse im Zusammenhang mit Wahrscheinlichkeitsmodellen sind zu unterscheiden vom allgemeinen Begriff des Ereignisses in der Informatik. Das *Wahrscheinlichkeitsmaß* ordnet den Ereignissen *Wahrscheinlichkeiten* zu. Die Wahrscheinlichkeit eines Ereignisses ist die Wahrscheinlichkeit, dass eines der im Ereignis enthaltenen Ergebnisse eintritt. Die Wahrscheinlichkeit der Vereinigung mehrerer disjunkter Ereignisse ist dementsprechend die Summe der Wahrscheinlichkeiten der einzelnen Ereignisse. Das leere Ereignis $\emptyset$, das keine Ergebnisse enthält, besitzt die Wahrscheinlichkeit 0. Das Ereignis Ω, das alle Ergebnisse enthält, tritt mit Wahrscheinlichkeit 1 ein. Die Forderung, dass das Ereignissystem $\mathcal{A}$ eine sogenannte Mengen-σ-Algebra sein soll, hat eher theoretische Bedeutung. Es existieren Regeln zur Wahl von $\mathcal{A}$, bei deren Anwendung diese Forderung immer erfüllt ist. Wenn der Merkmalraum Ω abzählbar ist, genügt es, als Ereignissystem $\mathcal{A}$ die Potenzmenge 2^{Ω} von Ω zu wählen, die alle Mengen von Ergebnissen, also alle überhaupt möglichen Ereignisse enthält. [Hüb03, S. 11-24]

Abbildung 2.7 veranschaulicht die Definition. Es folgt ein einfaches Beispiel für ein Wahrscheinlichkeitsmodell.

Beispiel 2.9:
Ein Würfel wird einmal geworfen. Mögliche Ergebnisse sind die sechs Augenzahlen.

$$\begin{aligned}\Omega &= \{1,2,3,4,5,6\}\\ \mathcal{A} &= 2^{\Omega}\\ &= \{\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\\ &\quad \{1,2\},\{1,3\},\{1,4\},\{1,5\},\{1,6\},\{2,3\},\{2,4\},\{2,5\},...,\\ &\quad \{1,2,3\},\{1,2,4\},\{1,2,5\},\{1,2,6\},\{1,3,4\},\{1,3,5\},...,\\ &\quad ...,\\ &\quad \{1,2,3,4,5,6\}\}\end{aligned}$$

Die Wahrscheinlichkeit für eine bestimmte Augenzahl beträgt $1/6$. Basierend darauf und auf den Gesetzmäßigkeiten für das Wahrscheinlichkeitsmaß können dann Wahrscheinlichkeiten von Ereignissen wie dem folgenden berechnet werden.

$$\begin{aligned}A &= \{2,4,6\}\\ P(\{2,4,6\}) &= P(\{2\})+P(\{4\})+P(\{6\}) = 1/6+1/6+1/6 = 1/2\end{aligned}$$

Beim Werfen eines Würfels können mehrere mögliche Ergebnisse eintreten. Ähnlich verhält es sich bei historischen Daten über bestimmten Variablen. Historische Daten geben genau eine von mehreren Möglichkeiten an, wie die Dinge verlaufen. In Beispiel 2.8 zu historischen Daten geht der Mobiltelefonbesitzer um 17 Uhr nach Hause. Es wäre aber auch möglich gewesen, dass er aufgrund eines zufallsbeeinflussten Missgeschicks länger arbeiten muss, um verlorene Arbeitsergebnisse neu zu erarbeiten. Es soll nun jeder mögliche Verlauf der Dinge, dargestellt durch historische Daten, als ein Ergebnis im Merkmalraum Ω aufgefasst werden. Es erfolgt eine entsprechende Definition.

Definition 2.11 (Wahrscheinlichkeitsmodell zu historischen Daten):
Das *Wahrscheinlichkeitsmodell zu historischen Daten* über den Variablen V_1, ...,V_n für das Zeitintervall $[1,m]$ ist festgelegt als $(\Omega,\mathcal{A},P)$ mit:

$$\begin{aligned}\Omega &= \{\mathcal{H} : \mathcal{H} \text{ sind historische Daten über } V_1, ...,V_n \text{ im Zeitintervall } [1,m]\}\\ \mathcal{A} &= 2^{\Omega}\\ P &\quad \text{gegeben durch prinzipielle Unsicherheit der Realität (vgl. Abschnitt 2.4.8)}\end{aligned}$$

So wird historischen Daten ein Wahrscheinlichkeitsmodell zugeordnet. Man könnte auch davon sprechen, dass historische Daten durch ein Wahrscheinlichkeitsmodell erzeugt werden. Die Definition setzt eine feste Struktur, also eine feste Wahl der Variablen und einen festen Zeitraum der historischen Daten voraus und drückt die verschiedenen mög-

lichen historischen Daten auf der Wert-Ebene durch ein Wahrscheinlichkeitsmodell aus. Die in Beispiel 2.8 dargestellten historischen Daten $\mathcal{H}$ über die Nutzung eines Mobiltelefons stellen zusammen ein $\omega \in \Omega$ unter vielen anderen dar. Wenn der Mobiltelefonbesitzer z.B. auch zusätzlich um 16:15 Uhr und um 17:45 Uhr telefoniert, ergibt sich ein weiteres $\omega \in \Omega$.

Das Konstruieren eines Wahrscheinlichkeitsmodells, dessen Ergebnisse einen ganzen Zeitraum beschreiben, stellt eine Idee dar, die auch bei stochastischen Prozessen Anwendung findet. Ein stochastischer Prozess basiert auf einem Wahrscheinlichkeitsmodell und sogenannten Zufallsvariablen X_t, die ein Ergebnis auf den Zustand zum Zeitpunkt t abbilden [Hüb03, S. 123].

Wenn man einen bestimmten Zeitpunkt in historischen Daten als Gegenwart festlegt, kann man den Begriff vergangener Ereignisse im Wahrscheinlichkeitsmodell zu den historischen Daten benutzen. Zur Erklärung des Begriffes eines vergangenen Ereignisses wird eine Bedingung an die Variablenwerte der Ergebnisse gestellt, die im Ereignis enthalten sind. Unter einem *vergangenen Ereignis* soll verstanden werden, dass alle möglichen Kombinationen von Werten der $v_{i,j}$ für alle Variablen V_i und alle Zeitpunkte j in der Gegenwart oder Zukunft jeweils in mindestens einem Ergebnis repräsentiert sind und dass nur für die $v_{i,j}$ mit j in der Vergangenheit Einschränkungen bezüglich der möglichen Kombinationen von Werten vorliegen. Entsprechend sind zukünftige Ereignisse zu verstehen.

2.4.8. Prinzipielle Unsicherheit und Zusammenhänge

Das im vorhergehenden Abschnitt beschriebene Wahrscheinlichkeitsmodell zu historischen Daten und die entsprechenden Wahrscheinlichkeiten beschreiben die *prinzipielle Unsicherheit*, wobei die Bezeichnung an Hübner angelehnt ist [Hüb03, S. 1]. In dieser Arbeit ist mit prinzipieller Unsicherheit eine Art von Unsicherheit gemeint, die man bei Prognose nicht umgehen kann. Die Annahme der Existenz dieser Art von Unsicherheit wird durch die frequentistische Auffassung von Wahrscheinlichkeit gerechtfertigt, die dieser Arbeit zugrunde liegt. Die frequentistische Sichtweise geht davon aus, dass es objektive Wahrscheinlichkeiten als „physikalische Eigenschaft der Domäne“ gibt [Wit02, S. 91]. Auch im Bereich der Quantentheorie wird die Existenz prinzipieller Unsicherheit deutlich.

Die Wahrscheinlichkeit, die das Wahrscheinlichkeitsmodell zu historischen Daten jedem möglichen Ablauf des Verhaltens des betrachteten Systems zuordnet, drückt die prinzipielle Unsicherheit aus. Aufgrund der Mächtigkeit der Zustandssichtweise ↑, auf der historische Daten basieren, und der Berücksichtigung prinzipieller Unsicherheit beschreibt ein Wahrscheinlichkeitsmodell zu historischen Daten das Verhalten des betrachteten Systems in einer so allgemeinen Weise, dass es in dieser Arbeit als universelle Darstellung von Vorwissen für eine allgemeine Sicht auf Prognose verwendet wird (vgl. auch Abbildung 2.5). Manche der folgenden Formulierungen unterscheiden daher kaum zwischen Realität und Wahrscheinlichkeitsmodell (ähnlich wie in [Sig08, S. 80]).

Da prinzipielle Unsicherheit als Eigenschaft der Realität festgelegt ist, wird sie nicht als Hindernis, sondern als Teil des Ziels von Prognose begriffen. Eine Prognose ist demnach

perfekt, wenn sie die Wahrscheinlichkeiten zukünftiger Ereignisse als Ausdruck prinzipieller Unsicherheit bestimmen kann unter der Berücksichtigung vergangener Ereignisse.

Auf der gleichen Ebene wie prinzipielle Unsicherheit sind Zusammenhänge (im konkreteren Sinne des Wortes) angesiedelt. Zusammenhänge sind ebenso wie prinzipielle Unsicherheit der Realität innewohnend, da sie über das Wahrscheinlichkeitsmaß des Wahrscheinlichkeitsmodells ausdrückbar sind, was unten in diesem Abschnitt gezeigt wird.

Prinzipielle Unsicherheit führt dazu, dass keine sicheren Aussagen darüber möglich sind, welchen konkreten Zustand das betrachtete System in der Zukunft einnehmen wird, sondern dass stattdessen nur Wahrscheinlichkeiten angegeben werden können. Es stellt sich die Frage, inwieweit sichere Aussagen über konkrete zukünftige Systemzustände möglich sind, wenn man das Wahrscheinlichkeitsmaß des Wahrscheinlichkeitsmodells kennt. Von Interesse ist also der Grad an (prinzipieller) Unsicherheit, der mit dem Grad an Zusammenhängen verbunden ist. Je stärker die Zusammenhänge sind, desto geringer ist die Unsicherheit. Je eindeutiger die Wahrscheinlichkeiten zugunsten bestimmter Ergebnisse verteilt sind, desto höher ist der Grad an Zusammenhängen und desto geringer ist der Grad an Unsicherheit. Wenn wenig prinzipielle Unsicherheit vorliegt, kann auf die Prognose von Wahrscheinlichkeiten verzichtet werden. Es können dann konkrete Zustände prognostiziert werden. Wenn z.B. einem Ergebnis eine Wahrscheinlichkeit von 1 zugeordnet ist, kann nur dieses eintreten, also nur eine Kombination von Variablenwerten. Die Variablen hängen dann maximal miteinander zusammen. Unsicherheit existiert keine. Das System ist deterministisch. Wenn die komplette Wahrscheinlichkeit gleichmäßig auf alle Ergebnisse verteilt ist, besteht maximale Unsicherheit darüber, welches Ergebnis eintritt. Es existieren dann keine Gesetzmäßigkeiten. Der Zusammenhang zwischen den Variablen ist minimal, weil alle Wertekombinationen die gleiche Wahrscheinlichkeit besitzen. Es ist also nicht der Fall, dass ein Wert einer Variablen mit besonders hoher Wahrscheinlichkeit in Kombination mit bestimmten Werten der anderen Variablen eintritt.

Der konkrete Umgang mit prinzipieller Unsicherheit und Zusammenhängen ist in der Praxis relativ gut möglich. Sie lassen sich mit Wahrscheinlichkeiten beschreiben. Das Wahrscheinlichkeitsmaß gibt die Wahrscheinlichkeit von Ereignissen an. Relevant für die Praxis wäre es aber auch, Zusammenhänge im konkreteren Sinne des Wortes beschreiben zu können. Was dafür benötigt wird, sind *bedingte Wahrscheinlichkeiten*. Eine bedingte Wahrscheinlichkeit $P(A|B)$ drückt aus, dass man das Eintreten eines bestimmten Ereignisses B voraussetzt und die Wahrscheinlichkeit bestimmt, dass zusätzlich ein anderes Ereignis A eintritt. Man möchte die Wahrscheinlichkeit bestimmen, dass A eintritt unter der Bedingung, dass B eintritt. Die Definition für bedingte Wahrscheinlichkeiten folgt.

Definition 2.12 (Bedingte Wahrscheinlichkeit):

Seien A und B Ereignisse. Dann gilt $P(A|B) := \frac{P(A \cap B)}{P(B)}$. [Hüb03, S. 27]

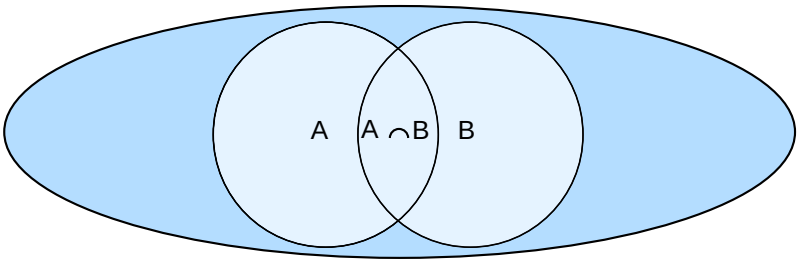

Abbildung 2.8.: zwei Ereignisse und deren Schnittmenge in einem Merkmalraum

$P(A \cap B)$ ist die Wahrscheinlichkeit, dass ein Ergebnis eintritt, das sowohl im vorausgesetzten Ereignis B, als auch in A enthalten ist (vgl. Abbildung 2.8). Da das Ereignis B vorausgesetzt ist, wird der Anteil der Wahrscheinlichkeit $P(A \cap B)$ an der Wahrscheinlichkeit, dass B eintritt, berechnet. Man kann nun z.B. annehmen, dass die Wahrscheinlichkeit für das gemeinsame Eintreten von A und B im Verhältnis zur Wahrscheinlichkeit für das Eintreten von B relativ groß ist. Dann ist die Wahrscheinlichkeit $P(A|B)$, dass A eintritt unter der Bedingung, dass B eintritt, auch relativ groß. Wenn A und B dagegen disjunkt sind, ist $P(A|B) = 0$, weil A und B gar nicht zusammen eintreten können. Ein weiterer, der Veranschaulichung dienender Spezialfall liegt vor, wenn man $B = \Omega$ setzt. Dann degeneriert $P(A|B)$ zu $P(A)$.

Für den nächsten Schritt bietet es sich an, die Definition 2.7 historischer Daten noch ein wenig zu ergänzen. Es sei angenommen, dass eine Menge von Variablen ausgewählt und ein Zeitraum festgelegt ist, aber die Werte der Variablen noch nicht vorliegen. Es werden in der Definition historischer Daten die Ebene der Variablen V_i und die Ebene ihrer Werte $v_{i,j}$ unterschieden. Jeder Variablen ist ein Wert pro Zeitpunkt j zugeordnet. Seit der Einführung eines Wahrscheinlichkeitsmodells zu historischen Daten kann eine Variable aber nicht nur zu unterschiedlichen Zeitpunkten unterschiedliche Werte annehmen, sondern auch zum gleichen Zeitpunkt, da der konkrete Wert einer Variablen zu einem Zeitpunkt nicht feststeht, sondern durch Wahrscheinlichkeiten bestimmt ist. Wegen der unterschiedlichen möglichen Werte zu bestimmten Zeitpunkten macht es Sinn, für unterschiedliche Zeitpunkte unterschiedliche Variablen $V_{i,j}$ einzuführen. Die Variablen V_i werden im Gegensatz zu den Variablen $V_{i,j}$ benutzt werden, wenn der Zeitpunkt keine Rolle spielt oder klar ist. Somit sind jetzt sowohl auf der Variablen- als auch auf der Wert-Ebene eine Variante mit und eine ohne Zeit definiert. Tabelle 2.2 gibt eine Übersicht.

In der Stochastik existiert das Konzept von *Zufallsvariablen* [Hüb03, S. 19-21]. Die in der Definition historischer Daten benutzten und jetzt wieder aufgegriffenen Variablen werden im Weiteren auch als Zufallsvariablen interpretiert werden. Eine Zufallsvariable ist eine Abbildung, die einem Ergebnis etwas zuordnet, das einen bestimmten Aspekt des Ergebnisses ausdrückt [Hüb03, S. 19-21]. Eine vollständige Definition von Zufallsvariablen ist in [Hüb03, S. 20] zu finden. Wenn man zu Beispiel 2.8 über die Nutzung eines Mobiltelefons und den entsprechenden historischen Daten das Wahrscheinlichkeitsmodell konstruiert, kann man z.B. $Y_4 = V_{2,4}$ als Zufallsvariable verwenden, die jedem Er-

	Ohne Zeit	Mit Zeit
Wertebene	v_i	$v_{i,j}$
	$x = v_1$	$x_j = v_{1,j}$
	$y = v_2$	$y_j = v_{2,j}$
	$z = v_3$	$z_j = v_{3,j}$
Variablenebene	V_i	$V_{i,j}$
	$X = V_1$	$X_j = V_{1,j}$
	$Y = V_2$	$Y_j = V_{2,j}$
	$Z = V_3$	$Z_j = V_{3,j}$

Tabelle 2.2.: Übersicht über Variablen- und Wertebene historischer Daten

gebnis die Anzahl offener verpasster Anrufe zum Zeitpunkt 4 zuordnet. Zufallsvariablen bieten eine kompakte und exakte Art, Ereignisse zu formulieren, zu denen man dann die Wahrscheinlichkeit bestimmen kann, z.B. $P(\max_{j=1}^{m}(Y_j) > 0)$ = P(„mindestens ein offener verpasster Anruf im Zeitraum“).

Nun kann das *Prognoseproblem über bedingte Wahrscheinlichkeiten* formuliert werden, also das Problem der Gewinnung von gesuchten aus gegebenen Informationen, wobei hier auch Folgerungsprognose eingeschlossen ist. Wenn man den Zeitpunkt j_{jetzt} als aktuellen Zeitpunkt annimmt und die tatsächlich in Vergangenheit und Gegenwart aufgetretenen Werte mit $v_{i,j}$ bezeichnet, lässt sich das Prognoseproblem formulieren als Bestimmung von P(gesuchte Informationen | gegebene Informationen) $\widehat{=}$ P(gesuchtes Ereignis | „$V_{i,j} = v_{i,j}$ für alle $V_{i,j}$ mit $j \leq j_{jetzt}$“). Dabei ist es auch möglich, dass man sich für mehrere gesuchte Ereignisse interessiert. Normalerweise wird man auch nicht die komplette Vergangenheit und Gegenwart als gegebene Informationen einbringen. Eine etwas realistischere Prognose ergibt sich aus Beispiel 2.8 über die Nutzung eines Mobiltelefons. Man könnte sich dafür interessieren, ob ein offener verpasster Anruf um 16:30 Uhr ($Y_3 = V_{2,3} = 1$) dazu führen wird, dass der Nutzer um 17:00 Uhr telefoniert ($Z_5 = V_{3,5} = true$). Man müsste also P($Z_5 = true \,|\, Y_3 = 1$) bestimmen, wobei sich in diesem Fall das Letztere aus dem Ersten ergibt. Das benutzte Vorwissen bei der Prognose stellt in dieser Formulierung der Prognose das Wahrscheinlichkeitsmaß P dar.

Wenn P($Z_5 = true \,|\, Y_3 = 1$) abweicht von P($Z_5 = true$), liegt ein Zusammenhang zwischen Y_3 und Z_5 vor, ansonsten sind die Variablen unabhängig. An dieser Stelle wird deutlich, dass Zusammenhänge (im konkreteren Sinne des Wortes) unmittelbar mit prinzipieller Unsicherheit in Verbindung stehen. Das Beispiel legt nahe, dass ein Zusammenhang existiert, denn bei einem offenen verpassten Anruf neigt ein Nutzer in der Regel eher dazu zu telefonieren. Es ist aber auch vorstellbar, dass der Nutzer sowieso um 17 Uhr telefonieren möchte. In diesem Fall könnte gelten, dass $P(Z_5 = true) \approx 1$ und auch $P(Z_5 = true | Y_3 = 1) \approx 1$, allgemein: $P(A) = P(A|B)$. Die Gleichung $P(A) = P(A|B)$ ist für $P(B) \neq 0$ äquivalent zu $P(A \cap B) = P(A) \cdot P(B)$ [Hüb03, S. 29-30]. Die durch die letzte Gleichung gegebene Eigenschaft nennt sich *stochastische Unabhängigkeit* [Hüb03, S. 29-30].

Ein Zusammenhang zwischen zwei Variablen X und Y wird in dieser Arbeit geschrieben als $X \Leftrightarrow Y$. Die Symmetrie des Pfeilsymbols ist gerechtfertigt durch die Symmetrie der stochastischen Unabhängigkeit. Wenn der Zusammenhang zu Zwecken wie Progno-

se in eine bestimmte Richtung genutzt werden soll, wird dies durch die Notation $X \rightarrow Y$ bzw. $Y \rightarrow X$ ausgedrückt. Eine der beiden Variablen wird dann *unabhängige* und die andere *abhängige Variable* genannt, wobei sich die Begriffsbildung an der Statistik orientiert. Diese Terminologie bietet sich im Vergleich zu gesuchten und gegebenen Informationen an, wenn beim Vorgang der Prognose mehrere Zusammenhänge im konkreteren Sinne des Wortes in Kombination genutzt werden.

2.4.9. Schätz-, Variations- und Methodenunsicherheit

Prinzipielle Unsicherheit stellt kein Hindernis für eine perfekte Prognose dar ↑. Die Frage nach weiteren Unsicherheitsfaktoren ist jedoch noch offen. Eine besonders gravierende Unsicherheit liegt darin, dass das Wahrscheinlichkeitsmaß des Wahrscheinlichkeitsmodells zu historischen Daten nicht immer gut bestimmt werden kann. Wenn man zur Prognose einen separaten Lernschritt durchführt, handelt es sich dabei um ein Problem des Lernens. Prognosen können durch diese Unsicherheit sogar unbeschränkt schlecht werden. Warum das so ist, soll nun erläutert werden.

Zufallsexperimente

Das Ermitteln von Wahrscheinlichkeitsmaßen zu Wahrscheinlichkeitsmodellen ist jenseits von Zukunftsprognose nicht unbedingt ein Problem. Man kann zur Ermittlung ein sogenanntes *Zufallsexperiment* durchführen [Wit02, S. 91]. Ein Zufallsexperiment ist ein Vorgang, der ein potentiell vom Zufall beeinflusstes Ergebnis besitzt [Hüb03, S. 11]. Die möglichen Ergebnisse stammen aus dem Merkmalraum Ω des Wahrscheinlichkeitsmodells [Hüb03, S. 11-12]. Das Zufallsexperiment sollte unter Beachtung der Versuchsbedingungen reproduzierbar sein [Hüb03, S. 11]. Wenn man dann das Zufallsexperiment wiederholt, kann man über die Häufigkeit $h_n(A)$ eines Ereignisses A die Wahrscheinlichkeit $P(A)$ nähern [Hüb03, S. 21-22]. Die schließende Statistik nennt einen solchen Vorgang *Schätzen* [Ste07, S. 136-137]. Die Häufigkeit $h_n(A)$ gibt den Anteil der Anzahl an Durchführungen des Zufallsexperiments an, bei dem A aufgetreten ist [Hüb03, S. 22]. Der Index n stellt dabei die Anzahl der Durchführungen des Zufallsexperiments dar [Hüb03, S. 22]. Das empirische Gesetz der großen Zahlen sichert zu, dass die Häufigkeit für $n \rightarrow \infty$ konvergiert [Hüb03, S. 22]. Daher kann man durch Erhöhung von n beliebig genaue Schätzungen der Wahrscheinlichkeiten erhalten. Es handelt sich hier um die Domäne der schließenden Statistik [Ste07, S. 135]. Häufig ist auch schon bekannt, dass das Wahrscheinlichkeitsmaß einer sogenannten Verteilung (z.B. Normal- oder Binomialverteilung) entspricht, und man muss nur noch einen Parameter der Verteilung bestimmen [Ste07, S. 135], wobei eine Verteilung ein Wahrscheinlichkeitsmaß beschreibt. Es ist anzumerken, dass eine beliebige Verbesserung der Genauigkeit der Schätzung eines Parameters einer Verteilung durch Erhöhung der Anzahl an Durchführungen des Zufallsexperiments voraussetzt, dass der verwendete sogenannte *Schätzer* für den Parameter erwartungstreu ist [Ste07, S. 147]. Der Schätzer ist eine Funktion, die aus den Ergebnissen der Durchführungen des Zufallsexperiments den geschätzten Wert berechnet.

Schätzunsicherheit

Schätzen ist mit einer gewissen Unsicherheit verbunden, da Zufallsexperimente vom Zufall beeinflusste Vorgänge sind. Dieser Zufall gehorcht einem gewissen Wahrscheinlichkeitsmaß, aber es kann trotzdem z.B. der Fall eintreten, dass bei mehreren Durchführungen eines Zufallsexperiments immer wieder eher unwahrscheinliche Ergebnisse eintreten, was zu einer ungenauen Schätzung führen kann. Diese Art von Unsicherheit wird in dieser Arbeit *Schätzunsicherheit* genannt. Wegen des empirischen Gesetzes der großen Zahlen hilft es, die Anzahl der Durchführungen des Zufallsexperiments zu erhöhen, wenn man die Schätzunsicherheit verringern möchte. Allerdings verfügt man nicht immer über die Möglichkeiten für sehr viele Durchführungen oder scheut den Ressourcenaufwand. Die gute Nachricht lautet aber, dass man die Schätzunsicherheit quantifizieren und so in gewissem Sinne kontrollieren kann. Man kann z.B. den mittleren quadratischen Fehler einer Schätzung bestimmen [Ste07, S. 150].

Variationsunsicherheit

Das empirische Gesetz der großen Zahlen setzt eine Wiederholung des Zufallsexperiments unter gleichen Bedingungen voraus [Hüb03, S. 22]. Diese Voraussetzung wird konkretisiert durch die (mindestens in der schließenden Statistik) übliche *Forderung nach identischer und unabhängiger Verteilung* (wie z.B. in [VGS05, S. 2], [DGL97, S. 2], [Ste07, S. 137], vgl. auch [Kri99, S. 4]). In der Formulierung „identisch und unabhängig verteilt“ meint man mit einer identischen Verteilung, dass bei allen Durchführungen des Zufallsexperiments das gleiche Wahrscheinlichkeitsmaß der Realität zugrunde liegt und die ermittelten Werte bestimmt. Unabhängigkeit soll bedeuten, dass keine Zusammenhänge zwischen ermittelten Werten unterschiedlicher Durchführungen bestehen dürfen. Wenn es z.B. um Würfel geht, um mit einem einfachen Beispiel jenseits von Zukunftsprognose zu beginnen, sind diese Forderungen recht einfach zu erfüllen. Ein ähnlicher, praxisnaher Fall liegt bei einem Roulette-Gerät vor [Hüb03, S. 187], wie das folgende Beispiel zeigt.

Beispiel 2.10:
Es soll ein Roulette-Gerät für ein Kasino daraufhin untersucht werden, ob bestimmte Zahlen mit höherer Wahrscheinlichkeit auftreten als andere [Hüb03, S. 187]. Es werden die Wahrscheinlichkeiten für das Eintreten der einzelnen Zahlen geschätzt und miteinander verglichen. Dazu wird die Benutzung des Gerätes als Zufallsexperiment 1000 mal durchgeführt. Mögliche Ergebnisse im Merkmalraum Ω sind die einzelnen 37 Zahlen.

In diesem Beispiel unterscheidet sich das Wahrscheinlichkeitsmaß zwischen unterschiedlichen Durchführungen des Zufallsexperiments nicht. Es ist immer der Fall, dass jede Zahl mit einer festen Wahrscheinlichkeit auftritt, im Idealfall $1/37$. Die Durchführungen sind auch nicht voneinander abhängig. Wenn bei einer Durchführung das Ergebnis 15 eintritt, wirkt sich dies nicht darauf aus, welche Zahl bei der nächsten Durchführung das Ergebnis darstellt. Schwieriger wird es beim nächsten Beispiel.

Beispiel 2.11:
Die Leistung von Photovoltaik-Anlagen sollte kaum von der Nennleistung abweichen [Ste07, S. 1]. Deshalb möchte ein Produzent für eine beliebige frisch produzierte Anlage die Wahrscheinlichkeit ermitteln, dass sie eine bestimmte gerade noch tolerable Abweichung überschreitet. Dazu wird die Leistung von 1000 Anlagen gemessen.

Es handelt sich um ein problematisches Beispiel, weil die Wahrscheinlichkeiten durch *äußere Umstände* variieren können. Die den einzelnen Durchführungen des Zufallsexperiments zugrunde liegenden Wahrscheinlichkeitsmaße sind *nicht identisch*, wenn die Durchführungen mit zugelieferten Bauteilen unterschiedlicher Qualität durchgeführt werden. Die Wahrscheinlichkeitsmaße sind *nicht unabhängig*, wenn aufgrund der Existenz einer ganzen Charge minderwertiger Bauteile zu erwarten ist, dass auf eine Durchführung mit minderwertigen Bauteilen eine weitere Durchführung mit minderwertigen Bauteilen folgt. Diese Probleme werden besonders dann gravierend, wenn Aussagen über eine Situation gemacht werden sollen, in der sich die äußeren Bedingungen von denen unterscheiden, die bei den Durchführungen des Zufallsexperiments bestanden haben. Wenn das Zufallsexperiment mit Anlagen durchgeführt wird, die mit einwandfreien Bauteilen produziert wurden, aber jede zweite verbaute Charge zugekaufter Bauteile Mängel aufweist, kann das zu fatalen Fehlschlüssen führen. Auch langfristige Wiederholungen des Zufallsexperiments helfen nicht unbedingt, wenn man eine Aussage über ganz bestimmte Anlagen benötigt. Es könnte z.B. der Fall sein, dass die Bauteile der Zulieferer über lange Zeit hinweg keine Mängel aufweisen. Trotzdem kann es passieren, dass gerade dann, wenn ein besonders wichtiger Auftrag zu einem bestimmten Termin fertiggestellt werden soll, mangelhafte Bauteile von den Zulieferern geliefert werden.

Auch beim Lernen für Zukunftsprognosen ist man auf Zufallsexperimente angewiesen. Wenn man Zusammenhänge erkennen möchte, muss man in einer mehr oder weniger expliziten Weise Wiederholungen von Abläufen über die Zeit betrachten, denn ein einzelner Ablauf in Form von historischen Daten zeigt nur eine Möglichkeit des Verlaufes der Dinge. Es sind, abgesehen von deterministischen Systemen, unterschiedliche Abläufe über die Zeit möglich, die man berücksichtigen muss. Auch bei Anderson ist die Rede von der Wiederholung von Experimenten, die zu Zeitreihendaten führt [And71, S. 3].

Bei Prognose und dem Durchführen von Zufallsexperimenten für Wahrscheinlichkeitsmodelle zu historischen Daten tritt ebenso das Problem auf, dass die Forderungen nach identischer und unabhängiger Verteilung nicht im Allgemeinen erfüllt werden können. Kriebel hebt im Speziellen als besondere Eigenschaft von Zeitreihen hervor, dass aufeinander folgende Beobachtungen normalerweise nicht unabhängig sind [Kri99, S. 4]. Für das oben eingeführte Wahrscheinlichkeitsmodell zu historischen Daten kann man zunächst einmal feststellen, dass ein Zufallsexperiment zu einem solchen Wahrscheinlichkeitsmodell überhaupt nicht wiederholbar ist, da sich das Wahrscheinlichkeitsmodell auf einen bestimmten Zeitraum bezieht und man die Zeit nicht zurück drehen kann. Wenn nur ein Exemplar des betrachteten Systems zur Verfügung steht, muss man also den absoluten Zeitbezug fallen lassen und unterschiedliche Zeitintervalle als Wiederholungen eines Zufallsexperiments ansehen. Explizite Zusammenhänge mit der Zeit im

konkreteren Sinne des Wortes können jedoch weiterhin ausgedrückt werden, wenn man eine Variable mit der Zeit als Wert einführt.

Unterschiedliche Zeitbereiche als Durchführungen eines Zufallsexperiments sind in der Regel nicht unabhängig und basieren auch nicht auf dem gleichen Wahrscheinlichkeitsmaß. Während man es bei Betrachtung der gesamten Zeit mit einem festen prinzipiellen Wahrscheinlichkeitsmaß zu tun hat, können sich die Wahrscheinlichkeitsmaße für einzelne Zeitbereiche aus dem gesamten Zeitintervall von Zeitbereich zu Zeitbereich ändern. Formal gesehen liegen nun durch die unterschiedlichen Wahrscheinlichkeitsmaße auch mehrere, unterschiedliche Wahrscheinlichkeitsmodelle vor. Die nachfolgende Sichtweise besteht aber darin, dass eigentlich ein einzelnes Wahrscheinlichkeitsmodell angestrebt wird, die Gleichheit der Wahrscheinlichkeitsmaße jedoch gestört ist.

Dass sich die Wahrscheinlichkeitsmaße für einzelne Zeitbereiche unterscheiden, kann wie in Beispiel 2.11 darauf zurückzuführen sein, dass sich mit der Zeit *äußere Umstände* ändern, die nicht ermittelt und durch Variablen repräsentiert werden, was in manchen Anwendungsbereichen evtl. auch gar nicht möglich ist. Ähnlich zur Forderung nach identischer Verteilung in unterschiedlichen Zeitbereichen ist die *Stationaritäts-Eigenschaft*. Stationarität bedeutet, dass Wahrscheinlichkeiten unabhängig von der Zeit sind [KS97, S. 14]. Nicht-Stationarität ist sehr verbreitet bei Phänomenen aus Natur und Kultur [KS97, S. 15]. Stationarität impliziert, dass alle Parameter des betrachteten Systems, die relevant für dessen Dynamik sind, konstant bleiben [KS97, S. 13]. Kantz et al. heben also auch die Bedeutung dessen hervor, was hier als äußere Umstände bezeichnet wird. Die Qualität der Chargen von zugekauften Bauteilen in Beispiel 2.11 stellen ein Beispiel für äußere Umstände dar. Bei der Mobiltelefonnutzung in Beispiel 2.8 könnte ein Feiertag ein äußerer Umstand sein, der die Mobiltelefonnutzung beeinflusst. Sigg spricht auf einer nicht mathematischen Ebene das Phänomen makroskopischer Veränderungen einer Umgebung an [Sig08, S. 77], die als Veränderungen eines Wahrscheinlichkeitsmaßes interpretiert werden könnten. Als einen möglichen Grund identifiziert er äußere Einflüsse [Sig08, S. 77]. Bei Wittig ist im Zusammenhang mit Nutzern die Rede davon, dass Eigenschaften, Interessen, Ziele usw. oftmals zeitlichen Veränderungen unterliegen [Wit02, S. 65]. Das Gebiet des maschinellen Lernens beschäftigt sich mit einem ähnlichen Problem, das sich Concept Drift nennt [Wit02, S. 65] [Tsy04]. Das ganze Problem wird umso gravierender, je unschärfer das betrachtete System von seiner Umgebung abgegrenzt ist.

Äußere Umstände hängen auch mit dem Problem *mangelnder Unabhängigkeit* der Wahrscheinlichkeitsmaße unterschiedlicher Zeitbereiche, also unterschiedlicher Durchführungen des Zufallsexperiments zusammen. Ein konstantes Fortbestehen äußerer Umstände über eine gewisse Zeit kann zu einer solchen Abhängigkeit führen. Wenn in Beispiel 2.11 eine mangelhafte Charge zugekaufter Bauteile eintrifft, ist es wahrscheinlich, dass zwei nacheinander produzierte Photovoltaik-Anlagen beide von eingeschränkter Qualität sind. Anders gesagt ist nach dem Feststellen der Minderwertigkeit einer Anlage mit erhöhter Wahrscheinlichkeit zu erwarten, dass die Qualität der danach produzierten Anlage ebenfalls nicht einwandfrei ist. Es sind aber auch direktere Abhängigkeiten losgelöst von äußeren Umständen möglich. Wenn man die Zeitbereiche zu den Durchführungen des Zufallsexperiments zusammen als historische Daten betrachtet, bestehen Abhängig-

keiten einfach in Zusammenhängen zwischen Teilen der Daten.

Der Kern des Problems bei der Verletzung der Forderungen nach identischer und unabhängiger Verteilung liegt darin, dass das Wahrscheinlichkeitsmaß über die Zeit, also von Zeitbereich zu Zeitbereich variiert. Daraus resultiert eine gewisse Unsicherheit. Diese in den vorhergehenden Absätzen und auch im Folgenden beschriebene Unsicherheit wird in dieser Arbeit *Variationsunsicherheit* genannt. Wegen der Unkenntnis von äußeren Umständen kann die Variation nicht einfach vorhergesagt werden. Man weiß beim Lernen nicht, unter welchen Umständen man eigentlich lernt, und beim Prognostizieren ist unklar, inwieweit die Umstände beim Lernen mit denen übereinstimmen, die zu der Zeit herrschen, über die prognostiziert werden soll. Man stelle sich z.B. vor, dass der Mobiltelefonbesitzer aus Beispiel 2.8 in Urlaub fährt. Dann erscheint plötzlich alles anders als vorher. Was vorher gelernt wurde, trifft im Urlaub nur noch recht eingeschränkt zu. Umgekehrt eignet sich Vorwissen, das im Urlaub gelernt wurde, nur bedingt für die Anwendung im normalen Alltag. Das Problem kann sich weiter verschärfen, wenn die Wahrscheinlichkeitsmaße nicht nur unterschiedlich sind, sondern das Wahrscheinlichkeitsmaß eines Zeitbereiches auch noch von den aufgetretenen Ergebnissen in vergangenen Zeitbereichen abhängt. Das trifft z.B. in dem Fall zu, dass nur im Urlaub gelernt wurde. Durch die Abhängigkeit der Zeitbereiche im Urlaub wurde aus Durchführungen des Zufallsexperiments gelernt, die nicht repräsentativ [Ste07, S. 4] für alle Nutzungssituationen sind. Die Verletzung der Unabhängigkeit kann also recht problematisch beim Lernen sein.

Durch Variationsunsicherheit können deterministische Systeme so erscheinen, als wären sie nichtdeterministisch, wie das folgende Beispiel zeigt.

Beispiel 2.12:
Man stelle sich eine Maschine vor, die ein Auto montiert. Wenn nur ein kleiner Ausschnitt der Maschine beobachtet wird, erscheinen die beobachteten Vorgänge nichtdeterministisch. Die äußeren Umstände, unter denen die beobachteten Vorgänge erfolgen, verändern sich ständig, wobei auch alle Teile der Maschine außerhalb des betrachteten Ausschnitts als äußerer Umstand anzusehen sind. Bei zusätzlicher unglücklicher Wahl der Zeitbereiche lassen sich kaum Zusammenhänge feststellen. Tatsächlich unterliegt das Verhalten der Maschine und jedes seiner Einzelteile aber einem völlig deterministischen, prinzipiell vorhandenen Wahrscheinlichkeitsmaß. Ihr Zustand ist für jeden Zeitpunkt deterministisch bestimmt. Erst durch die Notwendigkeit, zum Lernen Zeitbereiche als Durchführungen von Zufallsexperimenten zu betrachten, zusammen mit der Einschränkung auf nur einen Ausschnitt der Maschine entsteht Variationsunsicherheit, die das System dann schließlich trotz prinzipiellem Determinismus beim Lernen nichtdeterministisch erscheinen lässt. Wenn man allerdings als Zeitbereich das Montieren genau eines Autos wählt und die komplette Maschine beobachtet, offenbart sich der Determinismus und man kann nach einer ausreichend hohen Anzahl an Durchführungen des Zufallsexperiments gut prognostizieren.

Umgang mit Variationsunsicherheit

Letztendlich ist Prognose darauf angewiesen, dass sich Wahrscheinlichkeitsmaße nicht zu häufig in unvorhersehbarer Weise ändern. Es kann nur etwas gelernt werden, wenn eine gewisse Stabilität in der Umgebung vorherrscht und wenn sie von konstanten oder sich nur langsam entwickelnden Gesetzen, also Zusammenhängen beherrscht wird [VGS05, S. 2]. Der traditionelle Weg, eine stabile Umgebung zu präzisieren, ist die Forderung nach identischer, unabhängiger Verteilung [VGS05, S. 2]. Wie man schon an den Beispielen gesehen hat, muss man sich bei Prognose häufig mit schwächeren Voraussetzungen zufrieden geben. Am letzten Beispiel wurde schon erkennbar, wie man Variationsunsicherheit verringern kann. Es werden im Folgenden grundsätzliche Möglichkeiten zur Verringerung von Variationsunsicherheit gesammelt. Dabei wird angestrebt, das Spektrum denkbarer Möglichkeiten so gut wie möglich abzudecken, da die Möglichkeiten als Anforderungen an Prognoseverfahren dienen werden.

Das Problem der Variationsunsicherheit kann verkleinert werden durch *Einbeziehen zusätzlicher Informationen* als Variablen in die historischen Daten, die in Zusammenhang mit dem betrachteten System stehen, denn dadurch vermeidet man eine einflussreiche, zu Variationen führende äußere Umgebung. Solche Variablen führen dazu, dass beim Lernen des Wahrscheinlichkeitsmaßes deren Werte einbezogen wird. Wenn dann beim Prognostizieren ein bestimmter Wert für eine solche Variable als Teil der gegebenen Informationen bekannt ist, können Ergebnisse des Wahrscheinlichkeitsmodells mit anderen Werten der Variablen beim Bestimmen der gesuchten Informationen ausgeschlossen werden. Für die Prognosequalität ist es demnach vorteilhaft, möglichst viele potentielle Einflussfaktoren einzubeziehen. Dieser Lösungsansatz steht allerdings in Konflikt mit dem generellen Wunsch nach der Schonung von Ressourcen, denn es ist zu erwarten, dass ein größerer Umfang historischer Daten auch zu einem höheren Ressourcenaufwand führt. Ein weiteres Problem besteht darin, dass das Einführen zusätzlicher Variablen nicht viel hilft, wenn entsprechende Trainingsdaten, also Durchführungen des Zufallsexperiments mit den interessanten Werten dieser Variablen nicht verfügbar sind. Eine Variable, die angibt, ob sich der Besitzer eines Mobiltelefons gerade im Urlaub befindet (vgl. Beispiel 2.8), nützt nicht viel, wenn noch nie gelernt werden konnte, während der Mobiltelefonbesitzer sich im Urlaub befunden hat.

Variationsunsicherheit entsteht, wenn die Forderungen nach unabhängiger und identischer Verteilung nicht erfüllt werden. Ob die Forderungen erfüllt werden, hängt von der *Wahl des Zufallsexperiments und der Zeitbereiche der Durchführungen* ab. Man kann z.B. aus bestimmten Gruppen von Variablen separat lernen oder sonstige Fokussierungen auf bestimmte Teile der Daten und entsprechende Zusammenhänge vornehmen. Durch geeignete Wahl von Zufallsexperimenten, also von Datenbereichen, aus denen man lernt, kann die Forderung nach identischer, unabhängiger Verteilung für einzelne Zufallsexperimente näherungsweise erfüllt sein, so dass man stabile Zusammenhänge lernt. Das kann auch beim Mobiltelefonbesitzer aus Beispiel 2.8 gelingen. Er wird vermutlich gewisse Gewohnheiten in seinem Telefonierverhalten aufweisen, die unabhängig davon sind, ob er sich im Urlaub befindet. Wer beispielsweise bei verpassten Anrufen im normalen Alltag nie zurückruft, wird es im Urlaub wahrscheinlich auch nicht tun. Es kommt also

darauf an, die *stabilen Inseln von Zusammenhängen* zu finden und so die Variationsunsicherheit zu meiden.

Durch geschickte Wahl von Zufallsexperimenten, die nur Inseln von Zusammenhängen lernen, kann man auch die im vorhergehenden Absatz angesprochenen *Ressourcenprobleme* verkleinern. Das Lernen des kompletten Wahrscheinlichkeitsmaßes als Ganzes würde nämlich wegen der Zuordnung einer Wahrscheinlichkeit zu jeder Wertekombination der Variablen $V_{i,j}$ zu einem hohen Aufwand führen. Erstens würde der Speicherplatzbedarf in Abhängigkeit von der Anzahl der Variablen $V_{i,j}$ exponentiell steigen und zweitens bräuchte man auch entsprechend viele Trainingsdaten, also eine zu lange insgesamt zu beobachtende Zeit beim Lernen.

Wenn man alle Möglichkeiten zur Verringerung von Variationsunsicherheit ausgeschöpft hat, kann man noch versuchen, möglichst gut mit der vorhandenen Variationsunsicherheit umzugehen. Es existieren Ansätze zur Anpassung an die unterschiedlichen Wahrscheinlichkeitsmaße unterschiedlicher Zeitbereiche [Wit02, S. 65]. Mayrhofer unterscheidet (für den Spezialfall von Nutzerverhalten) zwischen langsamen und schnellen Änderungen sowie zwischen temporären und permanenten Änderungen [May04, S. 131]. Wenn z.B. bekannt ist, dass sich das Wahrscheinlichkeitsmaß von Zeitbereich zu Zeitbereich langsam ändert, kann man beim Prognostizieren bevorzugt Vorwissen anwenden, das aus jüngeren Zeitbereichen gewonnen wurde [Wit02, S. 65].

Methodenunsicherheit

Die bisher genannten Arten von Unsicherheit erfassen noch keine Unsicherheit, die sich aus Ungenauigkeiten der eingesetzten Prognosemethode und deren Implementierung ergibt. Gemeint sind Vergröberungen von Daten, approximative Algorithmen und ähnliche Maßnahmen zur Verringerung der Komplexität und des Ressourcenaufwands sowie eine mangelnde Unterstützung der Zusammenhangsarten und der prinzipiellen Unsicherheit in den Trainingsdaten. Für diese Art der Unsicherheit wird in dieser Arbeit der Begriff *Methodenunsicherheit* eingeführt, um möglichst zu einer erschöpfenden Unterteilung von Unsicherheitsarten zu gelangen. In den Bereich der Methodenunsicherheit fällt z.B. auch, dass man ein komplettes Wahrscheinlichkeitsmaß nicht schätzen kann, wenn das betrachtete System zu viele Variablen zur Repräsentation benötigt. Eine Verringerung der Methodenunsicherheit führt nach der Bedeutung des Begriffes zu einem erhöhten Ressourcenaufwand. Methodenunsicherheit kann aber auch bewusst erhöht werden, um den Ressourcenaufwand zu verringern bzw. Ressourcen freizugeben und mit den frei gewordenen Ressourcen die Schätz- oder Variationsunsicherheit zu verringern.

Gemeinsame Aspekte von Schätz-, Variations- und Methodenunsicherheit

Es ist bereits an verschiedenen Stellen erwähnt worden, dass eine Verminderung von Schätz-, Variations- und Methodenunsicherheit in Konflikt mit der Schonung von *Ressourcen* steht. Daher ist zu vermuten, dass die Verfügbarkeit von Ressourcen in Fällen mit hoher Unsicherheit dieser drei Arten ein entscheidender Faktor für die Qualität von

Prognosen darstellt, so dass *Effizienz* gefordert ist.

Abschließend sollte nicht unerwähnt bleiben, dass Schätz-, Variations- und Methodenunsicherheit bei Prognosen über *deterministische chaotische Systeme* besonders groß werden können, wenn Prognosen gefordert sind, die über die nahe Zukunft hinaus gehen. Chaotisches Verhalten kann dazu führen, dass der mittlere Fehler bei der Vorhersage des konkreten Systemzustandes in Abhängigkeit von dem Zeitpunkt, für den der Zustand prognostiziert werden soll, exponentiell steigt [KS97, S. 4]. Das liegt an der Empfindlichkeit gegenüber Anfangsbedingungen [KS97, S. 59]. Bei Anfangsbedingungen geht es darum, dass sich das System in der Gegenwart in einem bestimmten Zustand befindet, mit dem man dann als gegebene Informationen eine Prognose durchführen kann. Die Eigenschaft der Empfindlichkeit gegenüber Anfangsbedingungen bezieht sich auf die Auswirkungen von unterschiedlichen Anfangsbedingungen in der Gegenwart auf die Zukunft. Empfindlichkeit gegenüber Anfangsbedingungen bedeutet, dass sich jede winzige Änderung des Anfangszustandes in der Gegenwart im Vergleich zu einem anderen vorstellbaren Anfangszustand in der Gegenwart mit der Zeit „aufbläht“ zu einer großen Änderung späterer Zustände [KS97, S. 59]. Es ergeben sich mit der Zeit völlig unterschiedliche zeitliche Verläufe des Zustandes [KS97, S. 59].

In der Praxis wird man chaotische Systeme häufig nur grob betrachten und nicht auf der Ebene, auf der die deterministischen Zusammenhänge liegen. Die Unsicherheit erscheint dann wie prinzipielle Unsicherheit, die zwar theoretisch vermeidbar ist, in der Praxis jedoch nicht. Zur Prognose der Augenzahl beim Werfen eines Würfels wird man den Vorgang z.B. nicht auf der Ebene mechanischer Gesetze darstellen, sondern als zufälligen Vorgang, bei dem jede Augenzahl mit Wahrscheinlichkeit $\frac{1}{6}$ eintritt. Eigentlich liegt in diesem Fall hauptsächlich Variationsunsicherheit vor, da man Lage und Bewegung des Würfels nicht zu 100% genau messen kann. Beim Beispiel des Würfels erscheint es aber intuitiver, eine solche Variationsunsicherheit als unvermeidlich und als prinzipielle Unsicherheit statt als Variationsunsicherheit anzusehen. So kann man von Anwendungsfall zu Anwendungsfall unterschiedlich festlegen, wo prinzipielle Unsicherheit anfangen und wo sie aufhören soll. Im Folgenden werden auch solche im Vergleich zur vorherigen Sichtweise aufgeweichten Auffassungen von prinzipieller Unsicherheit zugelassen.

2.4.10. Unsicherheitsarten im Vergleich

Es sind in den vorhergehenden Abschnitten die folgenden Arten von Unsicherheit identifiziert worden.

- Prinzipielle Unsicherheit
- Schätzunsicherheit
- Variationsunsicherheit
- Methodenunsicherheit

Es erfolgt ein zusammenfassender Vergleich der Unsicherheitsarten: Prinzipielle Unsicherheit ist schon aus der Bedeutung des Begriffes heraus unumgänglich und wird in

dieser Arbeit nicht als zu lösendes Problem angesehen ↑. Was als unumgänglich anzusehen ist, kann anwendungsspezifisch festgelegt werden ↑. Sie ist Ausdruck des zu prognostizierenden Systemverhaltens und lässt sich quantitativ durch Wahrscheinlichkeiten angeben ↑. Ein hoher Grad an Zusammenhängen geht mit einem niedrigen Grad an prinzipieller Unsicherheit einher und umgekehrt ↑. Ein niedriger Grad an prinzipieller Unsicherheit ermöglicht neben Prognosen von Wahrscheinlichkeiten auch Prognosen konkreter Zustände ↑. Das Ermitteln und Nutzen des Wahrscheinlichkeitsmaßes, auf dem Zusammenhänge und prinzipielle Unsicherheit basieren, ist das theoretische Optimum für die Güte einer Prognose ↑. Schätzunsicherheit lässt sich wie prinzipielle Unsicherheit gut quantifizieren ↑. Im Gegensatz zu prinzipieller Unsicherheit kann sie verringert werden, nämlich durch eine erhöhte Anzahl an Durchläufen des Zufallsexperiments ↑. Variationsunsicherheit lässt sich kaum erfassen, schwer handhaben und kann im Allgemeinen zu einer beliebig schlechten Prognose führen ↑. Diese Gefahr ist jedoch weniger gravierend, wenn das betrachtete System die näherungsweise Erfüllung der Voraussetzung von identischer, unabhängiger Verteilung ermöglicht ↑. Die näherungsweise Erfüllung dieser Voraussetzung und somit die Verringerung von Variationsunsicherheit kann dann erreicht werden durch Einbeziehen von zusätzlichen Informationen, die in Zusammenhang mit dem betrachteten System stehen, in die historischen Daten ↑. Eine weitere Verringerung der Variationsunsicherheit kann durch Fokussierung auf Inseln stabiler Zusammenhänge gelingen ↑. Methodenunsicherheit entsteht erst bei der Entwicklung von Prognosemethoden ↑. Maßnahmen zur Verringerung von Schätz-, Variations- und Methodenunsicherheit führen teilweise zu erhöhtem Ressourcenaufwand ↑, können aber die Prognosegenauigkeit verbessern. Schätz-, Variations- und Methodenunsicherheit betreffen alle die Korrektheit und Genauigkeit von geschätzten Wahrscheinlichkeiten als Ausdruck von prinzipieller Unsicherheit. Daher bietet sich für Schätz-, Variations- und Methodenunsicherheit auch der Begriff der *Unsicherheit zweiter Ordnung* an, den Jensen in einem ähnlichen Zusammenhang verwendet [Jen01, S. 87].

2.4.11. Gesuchte Informationen

Was prognostiziert werden soll, hängt natürlich vom Anwendungsbereich ab. Welche Darstellungsformen für die gesuchten Informationen vorstellbar sind und von welcher Art sie sein können, ist eine andere Frage. Ausgehend von den bisherigen Ausführungen werden hier einige Ideen zu diesem Thema gesammelt und nachfolgend als Unterscheidungsmerkmale für gesuchte Informationen dargestellt.

- Betrachtungseinheiten:
  - Variablen
  - Ereignisse
- Rolle der Zeit:
  - Informationen über Betrachtungseinheiten gesucht, Zeitpunkt möglicherweise vorgegeben
  - Zeitpunkt gesucht [Sig08, S. 207]

- Arten von Informationen über Variablen (bei Variablen als Betrachtungseinheit):
  - Wahrscheinlichkeiten verschiedener Werte (Wahrscheinlichkeitsverteilung)
  - Wert mit höchster Wahrscheinlichkeit (Modalwert)
  - Wahrscheinlichkeit für Wahrheit logischer Formel über Variablen
  - Erwartungswert und Varianz einer metrisch skalierten Variablen

Die Definition historischer Daten in dieser Arbeit (Definition 2.7) kann als zustandsorientiert charakterisiert werden. Das schließt aber nicht aus, dass gesuchte Informationen auf der Ebene von Verhalten [Obj09b, S. 15] und aktiven Elementen [Val07, S. 11] betrachtet werden ↑. Ereignisse im Sinne von Zustandsübergängen stellen ein Mittel zur Beschreibung von Verhalten dar [Obj09b, S. 11]. Daher werden in obiger Aufzählung neben Variablen auch Ereignisse als mögliche *Betrachtungseinheiten* angeführt.

Die *Rolle der Zeit* besteht bei Prognose in der Regel darin, dass man sie vorgibt, um dann Informationen über den entsprechenden zukünftigen Zeitpunkt oder das zukünftige Zeitintervall zu erhalten. Das ist z.B. der Fall, wenn man das Wetter für die nächsten Tage prognostiziert. Wenn das Wetter jedoch durch ein stabiles Hoch geprägt ist, wird man eher fragen, wie lange die gute Wetterlage noch anhalten wird. Darin liegt die Motivation für das Prognostizieren von Zeitpunkten, zu denen die Betrachtungseinheiten bestimmte Bedingungen erfüllen. Die Idee dazu stammt von Sigg [Sig08, S. 207].

In dem Fall, dass man Ereignisse im Sinne von Zustandsübergängen als Betrachtungseinheiten wählt, sind *Informationen über die Betrachtungseinheiten* relativ einfach. Man kann z.B. Aussagen folgender Art formulieren: Ein Ereignis tritt mit gewisser Wahrscheinlichkeit ein. Etwas komplizierter wird es bei Informationen über Variablen, wobei es sich um eine oder mehrere Variablen handeln kann. Der Wertebereich einer Variablen enthält Werte, die alternativ mit gewissen Wahrscheinlichkeiten eintreten können. Interessieren könnte eine Anwendung statt Wahrscheinlichkeiten auch, welcher Wert mit höchster Wahrscheinlichkeit eintritt und wie hoch die Wahrscheinlichkeit ist. In dem Fall handelt es sich um die Prognose eines konkreten Zustands. Der Wert mit der höchsten Wahrscheinlichkeit wird auch *Modalwert* genannt [Hüb03, S. 89]. Eine etwas allgemeinere gesuchte Information besteht darin, mit welcher Wahrscheinlichkeit Variablen eine bestimmte Bedingung erfüllen. Schließlich kann man bei metrisch skalierten Variablen auch deren *Erwartungswert* und je nach Bedarf zusätzlich die *Varianz* angeben [Hüb03, S. 89-105]. Der Vorteil liegt darin, dass man statt mit vielen Wahrscheinlichkeiten nur mit einem bis zwei Werten umgehen muss. Die Werte sind allerdings weniger aussagekräftig als eine Wahrscheinlichkeitsverteilung. Der Erwartungswert (bzw. der arithmetische Mittelwert als Schätzer dafür) ist relativ anfällig für Ausreißer, also besonders kleine oder große Werte [Ste07, S. 26]. Er wird durch sie stark beeinflusst, insbesondere auch durch extreme Ausreißer bei Messfehlern [Ste07, S. 26]. Eine Alternative zum Erwartungswert, die diesen Nachteil nicht aufweist, stellt der *Median* dar [Ste07, S. 26]. Der Median teilt den Wertebereich einer Variablen in zwei Bereiche und gibt die Grenze zwischen diesen an, so dass möglichst mit Wahrscheinlichkeit $0,5$ ein Wert aus der linken Hälfte und ebenso mit Wahrscheinlichkeit $0,5$ ein Wert aus der rechten Hälfte auftritt [Hüb03, S.

87-88]. Er berücksichtigt aber nicht, ob Werte weit entfernt von der Grenze oder nah an dieser sind.

Zusätzlich zu den bisher dargestellten Arten von gesuchten Informationen können *Qualitätsangaben* zur Prognose gemacht werden, die die Schätz-, Variations- und Methodenunsicherheit widerspiegeln. Wie man gesehen hat, sind je nach Fall unterschiedliche Arten von gesuchten Informationen und Darstellungsformen angemessen. Prognosemethoden müssen sich also am Anwendungsbereich orientieren.

2.5. Kontextdatenprognose als Herausforderung

Dieser Abschnitt arbeitet auf den Anforderungskatalog für Verfahren zur Kontextdatenprognose auf mobilen Geräten hin und führt gleichzeitig die Inhalte des Abschnitts 2.4 zur Prognose und der Abschnitte 2.1, 2.2 und 2.3 zusammen. Dabei werden Aussagen und insbesondere Anforderungen aus vorhergehenden Abschnitten aufgegriffen (zu erkennen am anklickbaren Verweissymbol „↑“) und neue Anforderungen entwickelt.

2.5.1. Nutzung von Prognose

Diese Arbeit beschäftigt sich mit der Zukunftsprognose von Kontextdaten, die im Folgenden wie bisher einfach Prognose genannt wird. Die Bedeutung der Prognose von Kontextdaten hat sich schon an mehreren Stellen in dieser Arbeit gezeigt, etwa durch das Beispiel der Prognose der Verfügbarkeit von Diensten zur kontextbasierten Kooperation ↑ und der Vorhersage des Telefonierverhaltens eines Mobiltelefonbesitzers für das Energiemanagement ↑. Das Beispiel über den Versand von Daten auf einem Flughafen ↑ beinhaltet eine Prognose über die Möglichkeit des rechtzeitigen Versandes von Daten über das ausgelastete lokale Netz am Gate. Ein Navigationssystem kann durch die Prognose des Verkehrsaufkommens die Streckenführung optimieren und so die Fahrtzeit verringern ↑.

Mayrhofer sieht neben der Verbesserung der Mensch-Computer-Interaktion durch ein gemeinsames Verständnis von zukünftigem Kontext Nutzungsmöglichkeiten von Kontextdatenprognose in automatisierenden Anwendungsbereichen wie Verkehrsplanung, Logistik, Just-In-Time-Produktion, medizinischer Frühwarnung, vorausschauendem Netzverbindungsmanagement und Hausautomation (z.B. Bestellung von Heizöl und Lebensmitteln) [May05, S. 31]. Mayrhofer und Nurmi et al. unterscheiden eine Reihe im Folgenden zusammengestellter *Nutzungszwecken*.

- Rekonfiguration (z.B. vorausschauendes Laden von Systembibliotheken) [May05, S. 32] [NMF05, S. 163]
- Energiemanagement [NMF05, S. 163-164]
- Frühzeitige Warnungen (z.B. im Verkehr, der Medizin und vor Systemüberlastungen) [May05, S. 32] [NMF05, S. 164]
- Planungsunterstützung [May05, S. 32] [NMF05, S. 164]

Die Prognose von Dienstverfügbarkeiten für das DEMAC-Projekt stellt ein Beispiel für Prognose zur Rekonfiguration dar. Bei der Prognose des Telefonierverhaltens eines Mobiltelefonbesitzers besteht der Zweck im Energiemanagement.

Für ein ubiquitäres System, das wie ein Assistent arbeitet, ergibt sich aus Prognosen ein ähnlicher Vorteil wie für Menschen, die auch regelmäßig versuchen, die Zukunft vorherzusehen. Ein Autofahrer ohne Navigationssystem versucht z.B., zukünftige Verkehrssituationen abzuschätzen, um sich dann für die richtige Strecke zu entscheiden. Wie Menschen treffen auch Assistenten Entscheidungen. Prognose ermöglicht *Proaktivität* [Fer07, S. 7]. Entscheidungen können durch Prognose also vorausschauend getroffen werden. Ferscha sieht unter Anderem die Vorhersage des Verhaltens von Akteuren und Objekten als eine essentielle Voraussetzung für die Realisierung intelligenter Systeme an [Fer07, S. 6].

Die Unterstützung einer Anwendung durch Prognose soll in dieser Arbeit in Form eines *generischen Prognosesystems* realisiert werden, das laut Zielsetzung in Verbindung zur Laufzeit mit genau einer Anwendung arbeitet, durch die Anwendung bzw. ein bestehendes Kontextsystem benutzt wird und das dafür zur Entwicklungszeit an die Anwendung bzw. das Kontextsystem angepasst werden muss.

2.5.2. Kontextdaten als historische Daten

Kontextdaten sind definiert worden als konkrete Repräsentation einer Instanz eines gegebenen Kontextmodells, wobei auch eine Darstellung im zeitlichen Verlauf möglich ist ↑. *Historische Daten* wurden als Grundlage zur Betrachtung von Prognose formal definiert (vgl. Definition 2.7). Es kann wie bei der Definition von Kontextdaten zwischen Instanzebene und Modellebene [Obj09a, S. 16-20] unterschieden werden. Das Modell historischer Daten, das hier auch Struktur der historischen Daten genannt wird, ist gegeben durch die Wahl der verwendeten Variablen V_i und des betrachteten Zeitraums $[1, m]$. Dass die Struktur historischer Daten auf diese Weise festgelegt werden kann und dass es überhaupt Variablen und Zeiträume gibt, bestimmt das Metamodell als Teil der Definition historischer Daten. Für die Bildung von Kontextmodellen wurde in dieser Arbeit kein bestimmtes Metamodell vorgeschrieben. Kontextdaten können aufgrund der Universalität des Metamodells historischer Daten ↑ als historische Daten dargestellt werden, zumindest wenn sich das Kontextmodell an der Definition von Kontext orientiert und auf Entitäten und Attributen basiert. Die Darstellung solcher Kontextdaten als historische Daten kann erfolgen, indem die Attribute aller Entitäten den Variablen der historischen Daten zugeordnet werden, wobei auch eine beliebige Anzahl an Instanzen von Entitäten erzeugt werden kann, da die Anzahl an Variablen nicht endlich sein muss. Die Eignung historischer Daten zur Darstellung von Kontext wird auch anhand eines Archivs im Internet namens *Context Database* [May] deutlich, das Kontextdaten im zeitlichen Verlauf für Forschungszwecke bereitstellt. Alle Datensätze in diesem Archiv verwenden eine Darstellungsweise, die dem Prinzip der Definition historischer Daten entspricht.[2]

[2]Zum Zeitpunkt der Erstellung dieser Arbeit waren die folgenden sechs Datensätze in der Context Database vorhanden: ICANN_2001_accdata.txt, wall0-bt-wlan-logfile.txt, LocomotionAnalysis.zip, NokiaContextData.zip, 40acc_basic.txt und benchmarks.zip. In den Datensätzen NokiaContextData.zip und

Neben dem gerade erörterten formalen Rahmen sind auch Inhalte von Bedeutung. Die *Charakteristika von Kontext* sollten durch ein Prognosesystem berücksichtigt werden. Kontext kann sehr vielfältig sein. Bei kontextgetriebenen Anwendungen kann die gesamte alltägliche Umgebung des Nutzers Teil des Kontextes werden ↑. Diese Vielfalt sollte aufgrund der angestrebten Generik ↑ bei der Prognose von Kontextdaten unterstützt werden.

Dazu sollten nicht zuletzt möglichst viele *Zusammenhangsarten* berücksichtigt werden, wobei deterministische, einen einzelnen Zeitpunkt betreffende und kausale Zusammenhänge aus folgenden Gründen als Anforderung an das Prognosesystem ausgeklammert werden. Deterministische Zusammenhänge sind nur ein Spezialfall von den als selbstverständlich vorausgesetzten nichtdeterministischen Zusammenhängen. Die Berücksichtigung von Zusammenhängen, die einen einzelnen Zeitpunkt betreffen, erfolgt bei Zukunftsprognose in der Regel implizit. Die Unterstützung kausaler Zusammenhänge wird nicht gefordert, weil schwer festlegbar ist, welche Zusammenhänge als kausal anzusehen sind.

Auch die Unterstützung von *Skalenniveaus* stellt eine wichtige Anforderung dar ↑, die sich unter Anderem aufgrund der Heterogenität der Messwerte unterschiedlicher Sensoren [May05, S. 34] stellt. Nominalem Skalenniveau ist jedoch eine eher höhere Bedeutung als ordinalem und metrischem beizumessen, weil es universell anwendbar ist ↑, viele Methoden entweder nur nominales oder gleich metrisches Skalenniveau fordern [Sig08, S. 209] und weil nur wenige Arten von Kontext metrisches Skalenniveau aufweisen [Sig08, S. 56].

Ebenso ist eine gute Unterstützung möglichst vieler *Kontextarten* gefordert. Entsprechende Kontextdaten sollten also von einem Verfahren zur Kontextdatenprognose dargestellt werden können und es sollten basierend auf diesen Daten möglichst gute Prognosen durchgeführt werden können. Entsprechend der Unterscheidung von Kontext nach der Art der Entitäten ↑ sollten möglichst viele Arten von Entitäten unterstützt werden.

Eine besonders wichtige Art von Kontext stellt primärer Kontext dar (vgl. Abschnitt 2.3.3). Da *Position*, *Identität*, *Aktivität* und *Zeit* die einzigen Arten primären Kontextes darstellen, sollten sie durch entsprechende Optimierungen besonders gut unterstützt werden. Wie gezeigt wurde, spielt die Position von diesen vier Arten eine ganz besonders wichtige Rolle ↑ ↑. Die Bedeutung von primärem Kontext ergibt sich unter Anderem daraus, dass sekundärer Kontext aus primärem bestimmt werden kann ↑. Es bestehen also zwischen primärem und sekundärem Kontext Zusammenhänge im konkreteren Sinne des Wortes. Sekundäre Teile des Kontextes sind von primären abhängig ↑. Das Wissen über diese Zusammenhänge sollte genutzt werden, da es zur geschickten Wahl von Zufallsexperimenten beitragen und so Variationsunsicherheit verringern ↑ und die Effizienz verbessern ↑ kann. Außerdem kann aus der Aussage, dass sekundärer Kontext aus primärem bestimmt werden kann, die Richtung entnommen werden, in der die Zusammenhänge im konkreteren Sinne des Wortes bei der Prognose genutzt werden.

Neben der Unterscheidung von primärem und sekundärem Kontext ist für Kontextdatenprognose die Unterteilung von Kontext in *Low-Level- und High-Level-Kontext* rele-

benchmarks.zip werden Variablenwerte bei Zustandsänderungen, also nicht zu äquidistanten Zeitpunkten gespeichert.

vant. Als sinnvoll erscheinen sowohl die Prognose auf Low-Level-Ebene (z.B. Prognose der Signalstärke von Netzwerkverbindungen), als auch die Prognose auf High-Level-Ebene (z.B. Prognose des Verkehrsaufkommens). Daher sollte die Prognose auf beiden Abstraktionsebenen unterstützt werden. Nach der Definition von (Zukunfts-)Prognose, die dieser Arbeit zugrunde liegt, können gesuchte und gegebene Informationen auch auf unterschiedlichen Abstraktionsebenen liegen, so dass es sich bei einer solchen Prognose um vertikale Verarbeitung handelt. Der Bestimmung von High-Level-Kontext aus Low-Level-Kontext ist eine hohe Bedeutung beizumessen ↑. Daher ist es wünschenswert, dass ein Prognoseverfahren diese auch beherrscht.

2.5.3. Dynamik von Kontext

Krämer schreibt der *Umgebung des Nutzers* Dynamik, Adaptivität und Interaktivität zu [KH07, S. XI]. Die Umgebung als Synonym für Kontext ist definiert über die Interaktion zwischen Nutzer und Anwendung ↑, wobei sich diese Arbeit auf Anwendungen beschränkt, die auf mobilen Geräten laufen. Häufig sind mobile Geräte genau einem Nutzer zugeordnet und werden vom Nutzer bei sich getragen ↑. Dynamik kann durch *Mobilität* des Nutzers mit dem Gerät sowie von anderen Geräten und deren Nutzern in der Umgebung entstehen ↑. Um eine solche Dynamik als Vorwissen der Prognose abbilden zu können, sollte die Darstellungsweise von Vorwissen die Erzeugung von *Instanzen* erlauben ↑. Wenn man bei der Modellierung von Kontext der Definition von Kontext folgt, handelt es sich um Instanzen von Entitäten. Neben der Darstellung des Erscheinens und Verschwindens von Instanzen sollten auch Prognosen über das Erscheinen und Verschwinden möglich sein.

Eine hohe Dynamik bedeutet, dass über die Zeit hinweg starke Veränderungen auftreten. Veränderungen können gut durch *aktive Elemente* dargestellt werden, die daher zur Modellierung zur Verfügung stehen sollten. Entsprechend sollten Prognosen über Ereignisse möglich sein.

Hohe Dynamik kann leider zu hoher *Variationsunsicherheit* führen, weil häufige Änderungen äußerer Umstände zu erwarten sind. Das gilt besonders für kontextgetriebene Anwendungen, bei denen der Alltag des Nutzers den Kontext bilden kann, denn dieser ist nur schwer abgrenzbar, z.B. aufgrund der Existenz sozialer Beziehungen. Schwere Abgrenzbarkeit erschwert die Verringerung von Variationsunsicherheit ↑.

Einen wichtigen Faktor für die Dynamik von Kontext stellt *menschliches Verhalten* dar. Menschliches Verhalten kann eine hohe Komplexität aufweisen, denn es stehen oft komplizierte Überlegungen dahinter. Wenn sich z.B. ein Termin einer Person verzögert, wird es sich möglicherweise für die Person nicht mehr lohnen, mit einer größeren Aufgabe zu beginnen, die sie für den Tag eingeplant hatte. Die Tagesplanung kann sich durch den verzögerten Termin komplett ändern. Man könnte auch vermuten, dass es sich in dieser und ähnlichen Situationen um *chaotisches Verhalten* handelt. Das Verhalten weist nämlich das Merkmal auf, dass kleine Unterschiede vorstellbarer Szenarien in Gegenwart und Vergangenheit zu großen Veränderungen in der Zukunft führen ↑.

Auf der anderen Seite hebt Sigg die Existenz von *häufigen, typischen Mustern* in menschlichem Verhalten hervor [Sig08, S.75-76]. Auch von Gewohnheiten ist die Rede [Sig08, S.

77]. Häufige Muster ermöglichen Prognose, denn man kann beim Erkennen eines Teils eines häufigen Musters auf den Rest schließen. Die einzelnen Teile eines häufigen Musters bilden einen Zusammenhang im konkreteren Sinne des Wortes.

Die Mischung von möglicherweise chaotischem Verhalten und durch äußere Umstände beeinflusstem Verhalten einerseits und von typischen Mustern andererseits unterstreicht die Notwendigkeit der geschickten Wahl von Zufallsexperimenten und des *Lernens stabiler Inseln von Zusammenhängen* ↑.

2.5.4. Anpassung an Nutzer

Kontext ist definiert durch den Nutzer und die Anwendung als entscheidende Ausgangspunkte ↑. Worin Kontext im konkreten Fall besteht und welchen Zusammenhängen er unterliegt, hängt daher davon ab, wer der konkrete Nutzer ist.

Nach Sigg ähneln sich die *häufigen Muster* im Verhalten unterschiedlicher Menschen [Sig08, S. 76]. Auf der anderen Seite ist aus dem alltäglichen Leben bekannt, dass sich Menschen in ihren Gewohnheiten, Interessen und Neigungen unterscheiden können. Auch Wittig baut seine Arbeit auf der These auf, dass es *inter-individuelle Unterschiede* zwischen Nutzern gibt [Wit02, S. 64]. Einen Beleg für die These im Bereich von Mobiltelefonen als Vertreter mobiler Geräte liefert eine Studie von Siemens, die die unterschiedlichen Bedürfnisse von Mobiltelefonnutzern beschreibt [Pie06]. Manche Menschen nutzen ihr Mobiltelefon zum Schreiben von E-Mails [Pie06, S. 131], andere besitzen ein Mobiltelefon, das nur über drei Tasten verfügt: die erste für Notrufe, die zweite zum Anrufen des Hausarztes und die dritte zum Anrufen der Familie [Pie06, S. 127]. Entsprechend unterschiedlich sind auch die Nutzungsweisen. Kunze trägt der Bedeutung des individuellen Nutzers Rechnung, indem er das Konzept benutzerorientierter Middleware vorschlägt [Kun08, S. 132-133].

Es stellt sich die Frage, inwieweit sich eine Prognose dem Nutzer anpassen sollte (vgl. auch [Wit02, S. 135-136]). Es geht darum, inwieweit für unterschiedliche Nutzer das gleiche Vorwissen verwendet werden kann und inwieweit es für jeden Nutzer individuell gebildet werden muss. Um die Genauigkeit der Prognose zu wahren, sollte individuelles Vorwissen verwendet werden, wo individuelle Zusammenhänge bestehen. Zusammenhänge, die sich bei allen Nutzern gleichen, müssen natürlich nicht als individuelles Vorwissen eingebracht werden. Ausgenutzt werden kann auch, dass manche Zusammenhänge innerhalb von Gruppen bestimmter Nutzer kaum variieren.[3]Die Studie von Siemens identifiziert z.B. mittels einer Clusteranalyse verschiedene Nutzergruppen mit unterschiedlichen Nutzungsweisen [Pie06]. Allerdings muss für die Nutzung gruppenspezifischen Vorwissens auch die Aufgabe bewältigt werden, zur Laufzeit zu erkennen, welcher Gruppe ein Nutzer angehört. Allgemein sollte vom individuellen Nutzer unabhängiges Vorwissen möglichst schon zur Entwicklungszeit der Anwendung vorbereitet werden, damit der Ressourcenbedarf und die Komplexität zur Laufzeit verringert werden. Außerdem kann dabei auch Vorwissen zu Zusammenhängen zwischen bestimmten Variablen gebildet werden, das zur Laufzeit nicht gewonnen werden könnte, da die Wer-

[3]Dank an Dr. Magdalena Kotnis von der Universität Stettin für Hinweise in diese Richtung!

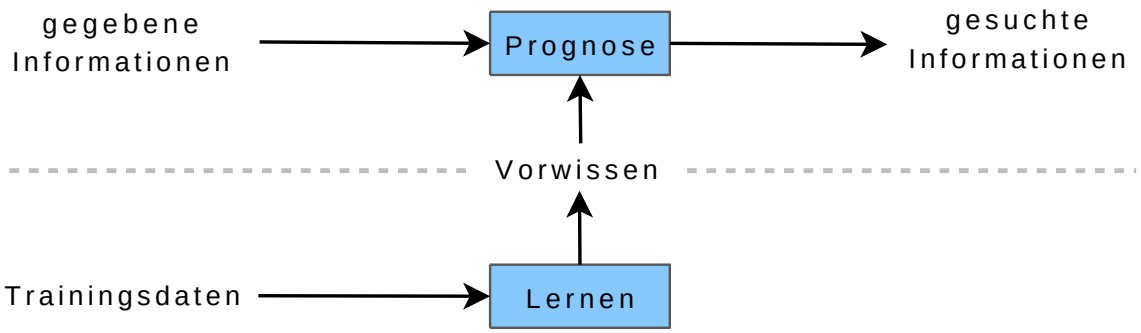

Abbildung 2.9.: unter den gestellten Anforderungen prinzipiell notwendige Aufgaben zur Kontextdatenprognose mit ihren Ein- und Ausgaben

te der Variablen nicht mit angemessenem Aufwand und unter Wahrung der Unsichtbarkeit im Ubiquitous Computing zur Laufzeit messbar sind.

Aufgrund der laut Zielsetzung der Arbeit angestrebten Generik sollte das Prognosesystem möglichst vielen Anwendungen gerecht werden, einschließlich Anwendungen, für die *inter-individuelle Unterschiede* eine Rolle spielen. Deshalb sollte das Prognosesystem die Fähigkeit zum Lernen besitzen. *Lernen* löst das Problem von Zusammenhängen, die sich von Nutzer zu Nutzer unterscheiden und somit noch nicht zur Entwicklungszeit bekannt sind ↑. Inter-individuelle Unterschiede spielen für viele Anwendungen eine Rolle. Das zeigt sich z.B. bei der Prognose des Telefonierverhaltens eines Mobiltelefonnutzers ↑ und der Prognose der Verfügbarkeit von Diensten ↑, da die Verfügbarkeit bei Einsatz von Ad-Hoc-Netzen von der Position des Gerätes abhängt. Ein Assistent muss seinen Nutzer in gewissem Maße kennen, um effektiv seinen Bedürfnissen und Wünschen gerecht werden zu können. Auch Sigg sieht Lernen als obligatorisch für Kontextdatenprognose an [Sig08, S. 77]. Da Lernen zur Überwindung inter-individueller Unterschiede nicht zur Entwicklungszeit erfolgen kann und somit zur Laufzeit erfolgen muss, handelt es sich um Online-Lernen. Zudem sollte adaptives Lernen statt Batch-Lernen eingesetzt werden, damit das gewonnene Vorwissen mit der Zeit verbessert und an Änderungen von Zusammenhängen angepasst und so die Prognosegenauigkeit verbessert werden kann. Außerdem wird durch adaptives Lernen eine explizite Lernphase, bevor überhaupt Prognosen möglich sind, aus Sicht des Nutzers vermieden ↑ und so die für das Ubiquitous Computing geforderte Unsichtbarkeit gewahrt. Radi sieht die Herausforderung bei Kontextdatenprognose gerade im Lernen ohne explizite Lernphase und mit möglichst wenig Nutzerinteraktion [Rad06, S. iv]. Auch Mayrhofer fordert adaptives Online-Lernen für Kontextdatenprognose [May05, S. 34].

Abbildung 2.9 zeigt die unter obigen Anforderungen prinzipiell notwendigen Aufgaben des Prognosesystems. Die in der Abbildung dargestellten Aufgaben der Prognose und das Lernens laufen zeitlich unabhängig voneinander ab.

Bei adaptivem Lernen ist zu berücksichtigen, dass zwar zu jedem Zeitpunkt Prognosen möglich sind, aber die Schätzunsicherheit zu Beginn sehr hoch sein kann, so dass der Angabe der Schätzunsicherheit eine besondere Bedeutung zukommt.

Für Vorwissen, das nicht von individuellen Unterschieden abhängt, wurde oben gefordert, dass dieses schon zur Entwicklungszeit eingebracht wird. Dies kann durch Nutzung

von Expertenwissen oder durch Analyse von Kontextdaten im zeitlichen Verlauf erfolgen und ist nicht immer einfach ↑. Die Analyse von Kontextdaten im zeitlichen Verlauf sollte durch Batch-Offline-Lernen unterstützt werden ↑. Dies ist aber nicht problematisch, da Methoden zum Online-Adaptionslernen prinzipiell auch offline, also zur Entwicklungszeit in mehreren Schritten größere Mengen an Trainingsdaten verarbeiten können. Auch das direkte, manuelle Einbringen von Vorwissen durch Experten (deduktives Lernen) sollte unterstützt werden. Diese beiden Arten des *Einbringens von a priori vorhandenem Wissen* können die Menge für gute Prognosen zur Laufzeit zu lernender Trainingsdaten verringern [Wit02, S. 68-69].

Bei der Nutzung von Expertenwissen spielen subjektive Einschätzungen eine Rolle.[4]Auch beim Batch-Lernen zur Entwicklungszeit können subjektive Einschätzungen eingehen, da anders als beim Online-Lernen keine Unsichtbarkeit zu wahren ist und somit Messungen durch das Befragen von Nutzern möglich sind. Diese Subjektivität findet in der Unschärfe der menschlichen Sprache [LSR03, S. VI] Ausdruck. Das Wort „warm“ kann z.B. sehr unterschiedlich interpretiert werden. Wenn man sich für das Wärmeempfinden eines Nutzers interessiert, stellt ein Thermometer jedoch keine Alternative dar. Daher bietet sich für das Einbringen von a priori vorhandenem Wissen eine spezielle Behandlung solcher Unschärfe an.

Um besser zwischen Vorwissen, das zur Laufzeit gelernt wird, und Vorwissen, das zur Entwicklungszeit entsteht, unterscheiden zu können, wird der Begriff des Prognosemodells eingeführt.

Definition 2.13 (Prognosemodell):
Ein *Prognosemodell* ist genau der Aspekt von Vorwissen, der sich auf der Modellebene im Sinne der OMG [Obj09a, S. 16-20] befindet. Der Begriff des Modells wird dabei so interpretiert, dass Modelle zur Entwicklungszeit entstehen.

Wenn man beispielsweise neuronale Netze zur Prognose einsetzen würde, könnte das Prognosemodell die Anzahl der Schichten des Netzes angeben. Die Definition greift die Begriffsbildung in Abschnitt 2.3.4 über Kontextmodelle auf, der ebenso zwischen Instanzebene (zur Laufzeit entstandene Kontextdaten) und Modellebene (zur Entwicklungszeit erstelltes Kontextmodell) unterscheidet. Prognosemodelle sind dadurch von Kontextmodellen abgegrenzt, dass sie Zusammenhänge, also den Aspekt des Verhaltens und der Gesetzmäßigkeiten hinter Veränderungen des Kontextes beschreiben.

Durch ein Prognosemodell kann sowohl die Repräsentation von Vorwissen auf der Instanzebene als auch, welche Arten von Zusammenhängen das Vorwissen enthalten soll, festgelegt werden. Mit der *Erstellung eines Prognosemodells* wird das Festlegen eines solchen Rahmens für Vorwissen auf der Instanzebene bezeichnet. Möglich ist auch das Einbringen von a priori vorhandenem Vorwissen, das zur Laufzeit auf der Instanzebene existieren soll, in ein Prognosemodell. Das ist vergleichbar mit statischen Variablen in der objektorientierten Programmierung. Deren Existenz auf der Instanzebene wird auch schon zur Entwicklungszeit festgelegt. Das *Einbringen von a priori vorhandenem Vor-*

[4]Dank an Dr. Magdalena Kotnis von der Universität Stettin für Hinweise in diese Richtung!

wissen wird begrifflich vom *Erstellen von Prognosemodellen* getrennt. Dennoch ist a priori vorhandenes, in ein Prognosemodell eingebrachtes Vorwissen gemäß der Definition von Prognosemodellen Teil des Modells.

Es ergibt sich die Anforderung der Unterstützung bei der Erstellung von Prognosemodellen. Unterstützungsbedarf besteht z.B., wenn historische Kontextdaten analysiert werden sollen.

Eine sich im Folgenden noch häufiger stellende, aber nicht pauschal beantwortbare Frage besteht darin, wie bedeutend die *Rolle eines Prognosemodells* als Teil des zur Prognose verwendeten Vorwissens sein soll.

2.5.5. Integration in Anwendungsszenarien

Die vorhergehenden Abschnitte haben sich mit der Anpassung an Dynamik und Nutzer beschäftigt, die überwiegend zur Laufzeit erfolgt. In diesem Abschnitt geht es um die grundsätzliche Eignung des zu entwickelnden Prognosesystems für unterschiedliche Anwendungsszenarien und die *Anpassbarkeit daran zur Entwicklungszeit*, die aufgrund der angestrebten Generik des Prognosesystems ↑ gefordert wird.

Auf niedriger technischer Ebene ist zunächst die Kompatibilität mit den verwendeten Geräten sicherzustellen. Um dies trotz der Heterogenität der Geräte zu erreichen, sollte das Prognosesystem möglichst plattformunabhängig sein ↑.

Zudem ist die *Kompatibilität mit unterschiedlichen Kontextsystemen* erforderlich. Verbindungen zwischen Prognosesystem und Kontextsystem bestehen beim Prognostizieren und beim Lernen, für das vom Kontextsystem Trainingsdaten bereitgestellt werden ↑. Das Prognosesystem muss dafür Schnittstellen bereitstellen, die universell genug sind, um ohne große Probleme von verschiedenen Kontextsystemen genutzt werden zu können. Das betrifft z.B. die Unterstützung der verbreiteten Benachrichtigungsmechanismen.

Für die Kompatibilität mit verschiedenen Kontextsystemen sind insbesondere auch Unterschiede zwischen Metamodellen zur *Kontextmodellierung* zu berücksichtigen. Aufgrund der unterschiedlichen Metamodelle von Kontextsystemen benötigt das Prognosesystem zur Gewährleistung der Kompatibilität mit beliebigen Kontextsystemen ein eigenes, von konkreten Kontextsystemen unabhängiges, universelles Metamodell für die Übermittlung von Kontextdaten zwischen Kontext- und Prognosesystem. Ein Kontextmodell, das diesem Metamodell entspricht, ist unabhängig vom Prognosemodell, das nur intern zum Lernen und zur Prognose verwendet wird. Das Kontext-Metamodell sollte so allgemein sein, dass Kontextdaten aus Kontextmodellen möglichst beliebiger Kontextsysteme damit dargestellt werden können. Die Definition historischer Daten stellt ein Beispiel für ein solches Metamodell dar. Ein Modell ist in diesem Fall festgelegt durch die Wahl der Variablen und des Zeitintervalls. Eine Instanz des Modells entspricht den historischen Daten selbst.

Anwendungsszenarien unterscheiden sich auch in der Art ihrer Nutzung von Kontextdatenprognose. Das Prognosesystem sollte Anwendungen ein möglichst breites Spektrum an *Arten und Darstellungsweisen gesuchter Informationen* bieten, denn für unterschiedliche Anwendungen eignen sich unterschiedliche Arten und Darstellungsweisen ↑.

2.5.6. Genauigkeit

Neben der prinzipiellen Funktionsfähigkeit des Prognosesystems und dessen Eignung für Kontextdaten kann man erwarten, dass die Prognosen möglichst gut die Zukunft widerspiegeln, also genau sind. Die besondere Bedeutung der *Genauigkeit* ist hervorzuheben. Eine geringe Zuverlässigkeit wird von Sigg als entscheidend [SHD06, S. 1] und als gravierendster Nachteil von Kontextdatenprognose [SHD07, S. 32] angesehen. Auch Mayrhofer zeigt die Bedeutung von Genauigkeit auf [May05, S. 32]. Die Genauigkeit von Prognosen entspricht der nicht-prinzipiellen Unsicherheit, mit der sie verbunden sind. Unsicherheit ist von entscheidender Bedeutung bei Entscheidungen ubiquitärer Systeme ↑. Ebenso wie bei Unsicherheit im Fall der Ermittlung von High-Level-Kontext aus Low-Level-Kontext ↑ ist ein bewusster Umgang mit Unsicherheit ↑ erforderlich. Beispiel 2.4 zeigt, wie ein Navigationssystem bewusst mit Unsicherheit, die ihr vom Prognosesystem signalisiert wird, umgehen kann ↑. Der bewusste Umgang mit Unsicherheit beugt schweren Schäden beim Agieren ubiquitärer Systeme im Kontext vor ↑. Eine Anwendung kann bei Unsicherheit den Nutzer einbeziehen ↑ [May05, S. 32] oder auch bei Billigung des Nutzers und unter Berücksichtigung der Unsicherheit selbstständig Entscheidungen fällen. Wenn z.B. die Prognose über das Telefonierverhalten eines Mobiltelefonbesitzers ↑ hohe Unsicherheit aufweist, wobei der Nutzer sein zukünftiges Telefonierverhalten evtl. selbst auch nicht besser abschätzen kann, könnte das System sinnvollerweise entscheiden, sparsam mit Energie umzugehen und so Risiken zu verringern. Bei der Prognose der Verfügbarkeit von Diensten für das DEMAC-Projekt ↑ wird das System versuchen, sich so zu konfigurieren, dass Dienste beim Aufruf voraussichtlich mit hoher Sicherheit verfügbar sind. Man kann sich auch Szenarien vorstellen, bei denen Unsicherheit sogar begrüßt wird. Wenn ein Navigationssystem z.B. zwischen zwei alternativen Routen entscheiden muss und für beide zur Fahrtzeit Stau prognostiziert wird, stellt die Route mit der größeren Unsicherheit der Prognose bei ansonsten gleichen Bedingungen die bessere Wahl dar, weil eine größere Chance auf Abwesenheit von Stau besteht.

Anzustreben ist die *Verringerung* von Schätz-, Variations- und Methodenunsicherheit, die die Prognosegenauigkeit beeinträchtigen ↑. Bei prinzipieller Unsicherheit sollte die bestehende Möglichkeit genutzt werden, die Unsicherheit, mit der eine konkrete Prognose behaftet ist, fundiert quantitativ anzugeben, z.B. durch Wahrscheinlichkeiten ↑. Das ist natürlich auch für die anderen Arten von Unsicherheit wichtig. Unsicherheit kann zusammen mit den gesuchten Informationen als Prognoseergebnis angegeben werden ↑. Je nach Anwendung kann es auch erforderlich sein, Informationen über mehrere mögliche Zukunftsszenarien anzugeben ↑. Die grundsätzlichen Möglichkeiten zur Angabe und Verringerung von Unsicherheit sollten genutzt werden.

Zur Verringerung der *Schätzunsicherheit* sollten ausreichend Trainingsdaten gelernt werden ↑. Wittig hebt für den Bereich nutzeradaptiver Systeme die Knappheit von Trainingsdaten hervor [Wit02, S. 63]. Diese wird auch verschärft durch das Einbeziehen zusätzlicher Informationen / Variablen zur Verringerung der Variationsunsicherheit ↑. Außerdem werden bei adaptivem Online-Lernen erst mit der Zeit Trainingsdaten gelernt. Deshalb sollten auch bei geringer Menge bereits gelernter Trainingsdaten schon möglichst genaue Prognosen möglich sein. Dabei ist eine möglichst gute Generalisierungs-

fähigkeit für gute Prognosen in möglichst vielen Situationen anzustreben, so dass Overfitting vermieden wird [Wit02, S. 63-64]. Auch das Einbringen von a priori bekanntem Vorwissen kann helfen, wenn noch nicht genug Trainingsdaten gelernt werden konnten [Wit02, S. 68-69], und sollte deshalb unterstützt werden. Wie bei prinzipiellen Unsicherheit sollten die Möglichkeiten der fundierten Angabe von Schätzunsicherheit genutzt werden ↑. Das ist besonders dann wichtig, wenn beim adaptiven Lernen noch nicht viele Trainingsdaten gelernt werden konnten ↑.

Zufallsexperimente und deren Durchführungen sowie die damit verbundenen Zeitintervalle sind sorgfältig zu wählen, so dass stabile Inseln von Zusammenhängen gelernt werden und die *Variationsunsicherheit* minimiert wird ↑. Dabei sollte die besondere Bedeutung räumlich und zeitlich nahen Kontextes ↑ ↑ berücksichtigt werden. Es ist eine Anpassung des Prognosesystems daran erforderlich, wo die stabilen Inseln liegen. Inwieweit diese Anpassung automatisch zur Laufzeit erfolgen oder zur Entwicklungszeit von den Anwendungsentwicklern durchgeführt werden sollte, bleibt offen. Zur Wahl von Zufallsexperimenten zählt auch die Wahl einbezogener Informationen. Dies schließt an die Anforderung nach dem Einbeziehen zusätzlicher Informationen zur Verringerung der Variationsunsicherheit an ↑.

Neben Möglichkeiten zur Verringerung von Variationsunsicherheit existieren Ansätze zur *Anpassung an Änderungen von Zusammenhängen* ↑. Solche Ansätze sollten angeboten werden. Ein einfacher Fall besteht darin, dass der Wert einer abhängigen Variablen beeinflusst wird durch einen langfristigen Trend als Zusammenhangsart ↑, der aufgrund seiner Länge meist nicht über Durchführungen von Zufallsexperimenten erfasst werden kann und dadurch zu Variationsunsicherheit führt. In diesem Fall kann eine solche Anpassung z.B. schon erreicht werden, indem man den Einfluss des Trends als separaten Anteil der abhängigen Variablen betrachtet ↑ und den Trend durch ein Polynom nähert. Außerdem kann man durch die Erkennung von Zusammenhangsänderungen, die offensichtlich implizit oder explizit bei diesen Ansätzen stattfinden muss, versuchen Variationsunsicherheit anzugeben.

Die *Methodenunsicherheit* sollte möglichst gering gehalten werden, sofern es die verfügbaren Ressourcen ermöglichen, und sie sollte angegeben werden.

Ein weiterer Unsicherheitsfaktor liegt außerhalb der Prognose und des Lernens. Trainingsdaten können aufgrund ungenauer oder grob fehlerhafter Messwerte unsicher sein ↑ [Bra07a, S. 15-17] [May05, S. 33] [Wit02, S. 68]. Es können auch Werte fehlen [Bra07a, S. 17] [May05, S. 33], so dass unsicher ist, welches die tatsächlichen Werte sind. Es ist außerdem möglich, dass Daten über logische Sensoren von Personen bezogen werden, die nur subjektive Einschätzungen [Wit02, S. 91] abgeben können. Unsicherheit durch *mangelhafte Trainingsdaten* sollte minimiert und angegeben werden.

Eine weitere Forderung liegt in der Konsistenz mehrerer, möglicherweise unterschiedlicher Prognosen, die zu nah beieinander liegenden Zeitpunkten erstellt werden. Bei weiter auseinander liegenden Zeitpunkten kann Konsistenz nicht im Allgemeinen gewährleistet werden, weil sich das Vorwissen durch Adaptionslernen zwischen den Prognosen ändern kann, ebenso wie die darin zum Ausdruck kommenden Zusammenhänge.

2.5.7. Effizienz

Eine hohe Prognosegenauigkeit kann einen hohen Aufwand an *Ressourcen* verursachen, die von den Basistechniken wie Geräten und Netzen bereitgestellt werden müssen ↑. Mobile Geräte sind jedoch eingeschränkt in der Nutzung von Ressourcen im Bereich Rechenleistung und Speicherkapazität ↑. Unter Effizienz soll in dieser Arbeit das Verhältnis zwischen Genauigkeit, anderen Gütekriterien wie Reaktionszeiten und der Menge an Prognosen auf der einen Seite und dem Bedarf an Ressourcen wie Rechenleistung und Speicherkapazität auf der anderen Seite verstanden werden. Eine hohe Effizienz kann dazu beitragen, den Ressourcenaufwand zu verringern oder bei vorgegebenem Ressourcenaufwand die Genauigkeit zu verbessern. Nur durch Effizienz kann das Prognosesystem sowohl der Anforderung nach hoher Genauigkeit, als auch der nach Schonung von Ressourcen gerecht werden ↑. Zur Sicherstellung der Effizienz sollten einige Anforderungen erfüllt werden.

Wegen des beschränkten *Speicherplatzes* auf mobilen Geräten sollte Online-Lernen adaptiv erfolgen. Beim Einsatz von Batch-Lernen würde eine Archivierung aller zu lernender Kontextdaten bis zum einmaligen Lernen notwendig werden, die sich für mobile Geräte nur bedingt eignet. Durch Abstraktion beim adaptiven Lernen werden die Datenmengen neuer Trainingsdaten dagegen kontinuierlich verringert ↑. Abstraktion ist zumindest insoweit einzusetzen, dass die Prognosegenauigkeit nicht wesentlich leidet. Zusätzlich oder alternativ ist auch eine separate Aggregation von Daten möglich. Zur Vergrößerung des Spielraums der Abstraktion sollte eine Anpassung des Prognosesystems daran erfolgen, welche Prognosen oder Arten von Prognosen benötigt werden ↑. Gefordert ist außerdem eine derartige Anpassung der Verteilung von Berechnungen auf Lernen und Prognose an das Anwendungsszenario und den Nutzer, die Speicherplatzbedarf und Rechenaufwand minimiert ↑ ↑.

Um mit Speicherplatz und Rechenleistung planen zu können, sollte es möglich sein, den Speicherbedarf und die Rechenzeit zu begrenzen. Selbst wenn keine *Begrenzung* festgelegt werden kann, sollte eine Begrenzung existieren, denn der verfügbare Speicherplatz und die verfügbare Rechenleistung sind immer endlich.

Die Effizienz beim *Lernen* sollte gefördert werden durch Anpassung an die Zusammenhänge in den Trainingsdaten, indem die stabilen Inseln von Zusammenhängen gelernt werden . Zudem sollte das Ausmaß des Lernens daran angepasst werden, wie stark die Notwendigkeit zum Lernen für die Prognosegenauigkeit ist. Wenn z.B. im Kontext selten Veränderungen stattfinden, besteht keine Notwendigkeit, mit hoher Frequenz gemessene Trainingsdaten ohne zeitliche Vergröberung zu lernen. Wenn schon sehr große Mengen an Trainingsdaten gelernt wurden, so dass die Schätzunsicherheit bereits minimiert wurde, und kaum Variationsunsicherheit vorliegt, lohnt es sich zunächst nicht, in bisherigem Umfang weiter zu lernen. Der notwendige Umfang an Trainingsdaten zum Lernen kann durch das Einbringen von a priori bekanntem Vorwissen verringert werden [Wit02, S. 68-69], weshalb dies ermöglicht werden sollte.

Bei *Prognosen* sollten ausreichend kurze Reaktionszeiten sichergestellt werden ↑. Dabei sollten natürlich auch keine starken Schwankungen der Reaktionszeiten auftreten. Das ist insbesondere bei der Verteilung des Rechenaufwands auf Lernen und Prognose zu

berücksichtigen ↑.

Das *Einbeziehen zusätzlicher Informationen* zur Verringerung der Variationsunsicherheit kann sowohl im Bereich des Lernens, als auch der Prognose zu hohem Ressourcenaufwand führen ↑. Deshalb ist ein effizienter Umgang mit großen Mengen an Variablen gefordert.

Maßnahmen zur Optimierung der Effizienz können die Methodenunsicherheit erhöhen ↑. Das führt aber letztendlich nicht notwendigerweise zu einer Verschlechterung der Prognosegenauigkeit, wenn durch die bei der Ressourcenverbesserung frei werdenden Ressourcen an anderer Stelle wiederum zur Verringerung von Unsicherheit eingesetzt werden.

Neben der theoretischen Effizienz von Prognose spielt auch der absolute Ressourcenkonsum eine Rolle. Nur bei einer an die Möglichkeiten des Gerätes angepassten Ressourcennutzung und einer geeigneten Zuteilung der Ressourcen zu den Prozessen auf dem Gerät kann für den Nutzer letztendlich ein hoher Nutzen und eine hohe Effizienz im Sinne des Verhältnisses von diesem Nutzen zum damit verbundenen Ressourcenaufwand erreicht werden. Deshalb und aufgrund der Diversifizierung ↑ und Heterogenität ↑ mobiler Geräte ist eine Anpassung an die Art von mobilen Geräten, auf denen das Prognosesystem eingesetzt werden soll, notwendig ↑. *Skalierbarkeit* kann sowohl die Anpassbarkeit an die Leistungsfähigkeit von Geräten, als auch an Qualitätsansprüche z.B. bezüglich der Genauigkeit bedeuten. Eine solche Anpassung kann auch zur Laufzeit erfolgen, z.B. bei hoher Auslastung des Gerätes.

2.5.8. Verteilte Prognose

Vernetzung spielt eine wichtige Rolle für mobile Geräte. Im Ubiquitous Computing werden die Paradigmen der Dezentralisierung und Konnektivität propagiert ↑. Das zu entwickelnde Prognosesystem arbeitet also in der Regel in einer verteilten Umgebung. Aufgrund der Heterogenität mobiler Geräte ↑ und ihrer Mobilität ↑ muss man davon ausgehen, dass ein Gerät über Kontextdaten verfügen kann, zu denen ein anderes Gerät keinen direkten Zugang hat. Deshalb ist es wichtig, auch Kontextdatenprognose verteilt durchführen zu können. Coulouris et al. sehen die Motivation für verteilte Systeme begründet im Wunsch des Teilens von Ressourcen [CDK05, S. 2]. Im Fall der verteilten Kontextdatenprognose besteht die Motivation im *Teilen von Kontextdaten* als Ressourcen. Es bieten sich bei mobilen Geräten wegen ihrer Ressourcenarmut ↑ *Rechenleistung*, *Speicherkapazität* und *Netzwerkverbindungen* als weitere Ressourcen zum Teilen an. Das Teilen von Rechenleistung, Speicherkapazität, Netzwerkverbindungen, Kontextdaten bzw. der Ergebnisse einer Verarbeitung von Kontextdaten stellt das Ziel einer *verteilten Kontextdatenprognose* dar.

Unter einem verteilten System wird nach Coulouris et al. ein System verstanden, in dem Komponenten auf vernetzten Computern durch Nachrichtenaustausch kommunizieren und sich koordinieren [CDK05, S. 1]. Schill et al. nennen als weitere wichtige Merkmale eines verteilten Systems, dass die Komponenten des Systems unabhängig sind und gemeinsam durch Kooperation eine Funktionalität bereitstellen, die keine der Komponenten allein bereitstellen könnte [SS07, S. 5]. Nachrichtenaustausch im Fall der ver-

teilten Kontextdatenprognose beinhaltet den oben als Aufgabe zur Context-Awareness dargestellten Austausch von Daten. Die kooperativ bereitgestellte Funktionalität besteht in der Prognose von Kontextdaten in Verbindung mit anderen Aktivitäten zur Context-Awareness.

Mit der Realisierung eines verteilten Systems sind einige Herausforderungen verbunden wie Nebenläufigkeit, separate Adressräume, Heterogenität, Fehlerbehandlung, Offenheit, Sicherheit, Skalierbarkeit und Transparenz [CDK05, S. 16-26] [Bra07b, S. 2-6]. Es wird vorausgesetzt, dass diese Probleme zu großem Teil bereits auf der Ebene der Anwendung oder des Kontextsystems gelöst sind und diese Lösungen beim Einbinden des Prognosesystems auch so weit wie möglich für das Prognosesystem verwendet werden, da die Probleme überwiegend auf allgemeiner Ebene, unabhängig von der Aufgabe der Kontextdatenprognose gelöst werden können. Zudem fügt sich das Prognosesystem so besser in die Anwendungslandschaft. Die Wiederverwendung bestehender Lösungen betrifft beispielsweise das Anbieten von Methoden über RPC oder Webservices und das Veröffentlichen von Diensten in einem Dienstverzeichnis.

Weitere Herausforderungen bezüglich der verteilten Kontextdatenprognose ergeben sich daraus, dass das Prognosesystem auf mobilen Geräten eingesetzt werden soll. Zu diesen Herausforderungen zählen Heterogenität, instabile Vernetzung und Adaptivität. *Adaptivität* bezeichnet gerade das oben als Ziel der Verteilung genannte Teilen von Ressourcen durch Zuteilung von Funktionen zu Geräten ↑ unter dynamischen Bedingungen bei ungleicher Verteilung von Ressourcen. Die Güte einer Zuteilung von Funktionen hängt davon ab, welche Funktionen für die Kontextdatenprognose zu erbringen sind und welche Ressourcen dafür benötigt werden.

Die Güte der Zuteilung von Funktionen hängt damit von der Architektur des Prognosesystems ab. Es kommt darauf an, welche Funktionen überhaupt durch Schnittstellen gekapselt und damit auf ein anderes Gerät verlagerbar sind. In jedem Fall kann aber die Gesamtfunktion des Prognosesystems verlagert werden. Es kann Verteilung entstehen, indem die Gesamtfunktion des Prognosesystems einem anderen Gerät zugeteilt wird als die Funktion, die die Trainingsdaten bereitstellt. So könnte z.B. ein mobiles Gerät zum Lernen Daten von einem Sensornetz beziehen. Diese Möglichkeit der Verteilung hat allerdings zur Folge, dass zu lernende Kontextdaten nicht immer vollständig und in aufeinander abgestimmten Intervallen gemessen werden können, sondern insgesamt heterogen sind ↑. Damit muss beim Lernen umgegangen werden können. Die andere Möglichkeit für Verteilung besteht darin, dass die Funktion, die Prognosen bezieht, auf einem anderen Gerät ausgeführt wird als das Prognosesystem. Das könnte in Beispiel 2.2 über den Versand von Daten mittels eines lokalen Netzes auf einem Flughafen bei hoher Netzauslastung und Zeitknappheit dazu genutzt werden, vom Smart Space des Flughafens eine Prognose darüber abzufragen, wie sich die Netzauslastung entwickeln wird. Geräte, die anderen Geräten Prognosen anbieten, sollten Informationen über ihre Fähigkeiten veröffentlichen, so dass erkennbar wird, ob es sich um ein Gerät mit beschränkten Prognosefähigkeiten und eingeschränkter Genauigkeit wie z.B. ein Mobiltelefon oder um ein leistungsfähiges Gerät wie einen Server handelt. Neben diesen beiden Optionen der Verteilung sind je nach Architektur des Prognosesystems weitere Optionen möglich. Das

Prognosesystem sollte die Wahl dieser Optionen zur Entwicklungs- und zur Laufzeit erlauben, um Adaptivität zu ermöglichen.

Es wird vorausgesetzt, dass die Anwendung bzw. das Kontextsystem die Wahl der Optionen übernimmt, also entscheidet, welche Funktionen welchem Gerät zugeordnet werden. Es obliegt also auch der Anwendung bzw. dem Kontextsystem zu entscheiden, inwieweit diese Zuteilung nur einmalig zur Entwicklungszeit erfolgt und inwieweit zur Laufzeit eine Anpassung an die Verfügbarkeit von Geräten, die Auslastung von Netzen usw., also den technikbezogenen Kontext stattfindet. Es können so die bestehenden Lösungsansätze von Anwendung bzw. Kontextsystem verwendet werden. Kontextsysteme mit einem zentralen Kontextserver können z.B. zur Entwicklungszeit den Grad der Zentralität der verteilten Prognose an das restliche System anpassen.

Ein Problem bei mobilen Geräten liegt in der instabilen Vernetzung. Mögliche Lösungsansätze sollten insoweit vorgesehen sein, wie sie direkt im Prognosesystem berücksichtigt werden müssen, so dass auch in diesem Bereich eine möglichst große Wahlfreiheit für die Anwendung bzw. das Kontextsystem besteht. Zur asynchronen Kommunikation können z.B. Callback-Methoden an der Schnittstelle des Prognosesystems notwendig sein.

Eine hohe Verbreitung im Bereich des Mobile Computing, der Context-Awareness und der Softwaretechnik hat das Paradigma der *Publish-Subscribe*-Kommunikation erreicht ↑ ↑ ↑. Publish-Subscribe-Mechanismen können auch im Bereich der Prognose die Netzlast verringern. Ein Navigationssystem könnte sich z.B. von einem Server über einen in der Zukunft zu erwartenden Stau benachrichtigen lassen, anstatt regelmäßig eine Prognose abzufragen. Ein Publish-Subscribe-Mechanismus kann in Verbindung mit bestehenden Lösungen realisiert werden, indem eine zusätzliche Softwareeinheit regelmäßig Prognosen initiiert und Benachrichtigungen verschickt. Vorteilhaft wäre es zudem, wenn das Prognosesystem ausnutzen könnte, dass wiederholt gleichartige Prognosen angefordert werden, um die Effizienz zu steigern.

2.5.9. Benutzbarkeit

Die Benutzbarkeit des Prognosesystems wird unterteilt in die Benutzbarkeit für Anwendungsentwickler und die Benutzbarkeit für Anwendungsnutzer. Anwendungsnutzer benutzen das Prognosesystem zwar nicht direkt, aber das Prognosesystem beeinflusst die Benutzbarkeit der Anwendung. Die Benutzbarkeit für Nutzer ist zu berücksichtigen als direkter Einflussfaktor bezüglich des Nutzens für den Nutzer, der laut Zielsetzung der Arbeit anzustreben ist.

Was gute Benutzbarkeit für den Nutzer bedeutet, hängt mit der Vision des Ubiquitous Computing zusammen. Unsichtbarkeit sollte gewährleistet sein. Das Prognosesystem sollte also z.B. möglichst wenig Nutzereingaben erfordern. Das betrifft sowohl den laufenden Betrieb als auch mögliche Wartungsaufgaben und die Verwaltung des Prognosesystems. Mayrhofer fordert eine unaufdringliche Operation [May05, S. 33]. Trotzdem sollten Entscheidungen ubiquitärer Systeme auf Wunsch auch nachvollziehbar vom System erklärt werden können ↑, was erfordert, dass auch Prognosen nachvollzogen werden können. Wittig stellt für benutzeradaptive Systeme eine ähnliche Anforderung, die

er Interpretierbarkeit nennt [Wit02, S. 66-67].

Eine gute Benutzbarkeit für Anwendungsentwickler impliziert, dass sie das Prognosesystem einfach und mit wenig Aufwand verwenden können. Es wird unterschieden zwischen Vorbereitungsaufwand (z.B. Lernaufwand und Erstellung von Prognosemodellen) und Aufwand beim Einsatz ↑. Beide sind zu minimieren.

2.6. Anforderungskatalog

In Abschnitt 2.5 über Kontextdatenprognose wurden bereits zahlreiche Anforderungen an Kontextdatenprognose auf mobilen Geräten herausgearbeitet. Dieser Abschnitt fasst die Anforderungen zur Übersicht in Form eines Anforderungskatalogs zusammen. Der Anforderungskatalog folgt in seinem *Aufbau* der Gliederung von Abschnitt 2.5. Es werden die Unterabschnitte „*Kontextdaten als historische Daten*“, „*Dynamik von Kontext*“, „*Anpassung an Nutzer*“, „*Integration in Anwendungsszenarien*“, „*Genauigkeit*“, „*Effizienz*“, „*Verteilte Prognose*“ und „*Benutzbarkeit*“ als Gruppen von Anforderungen übernommen.

Der Aufbau orientiert sich auch grob am Standard ISO 9126, Teil 1a zur Softwarequalität, der die verbreitete Unterscheidung zwischen externer Qualität (*Funktionalität*, *Effizienz*, *Benutzbarkeit* und *Zuverlässigkeit*) und interner Qualität (*Wartbarkeit* und *Portabilität*) beinhaltet [SX08, S. 110-111]. Die Bereiche „*Kontextdaten als historische Daten*“, „*Dynamik von Kontext*“, „*Anpassung an Nutzer*“ und „*Integration in Anwendungsszenarien*“ enthalten überwiegend funktionale Anforderungen und die Bereiche „*Genauigkeit*“, „*Effizienz*“, „*Verteilte Prognose*“ und „*Benutzbarkeit*“ überwiegend nicht-funktionale Anforderungen. Abbildung 2.10 zeigt die Abhängigkeiten der Anforderungsgruppen von Abschnitten dieses Kapitels über das Problemfeld und verdeutlicht so den Gedankengang des Kapitels.

Der Anforderungskatalog ordnet jede Anforderung der Gruppe zu, zu der sie den größten inhaltlichen Bezug aufzuweisen scheint, da Anforderungen bei einer Anwendung der Anforderungen zur Bewertung von Verfahren nicht mehrfach eingehen sollten. Es existiert eine *Vielzahl an Anforderungen*, die eine Idealvorstellung von Kontextdatenprognose auf mobilen Geräten ausdrücken und möglichst geeignete Bewertungen und Vergleiche von Verfahren erlauben sollen.

Um Prioritäten zu setzen, sind im Anforderungskatalog die Nummern der Anforderungen entsprechend der Wichtigkeit der Anforderung *unterstrichen*, *gestrichelt unterstrichen*, *nicht unterstrichen* oder *eingeklammert*. Unterstrichen sind die Nummern der Anforderungen, die essentiell wichtig erscheinen. Eingeklammert sind die Nummern der Anforderungen mit eher optionalem Charakter.

Die Wichtigkeit einer Anforderung kann sich z.B. aus der Notwendigkeit ihrer Erfüllung für die Funktionsfähigkeit des Prognosesystems ergeben. Anforderungen ohne Unterstreichung der Nummer lassen sich teilweise ohne Probleme durch Erweiterungen ohne Änderung des Grundansatzes des Prognoseverfahrens erfüllen. Anforderungen mit eingeklammerter Nummer sind in der Regel schon in ihrer Beschreibung in Abschnitt 2.5 als optional gekennzeichnet. Die Prioritäten sind auch ein Ausdruck der Schwerpunkte dieser Arbeit.

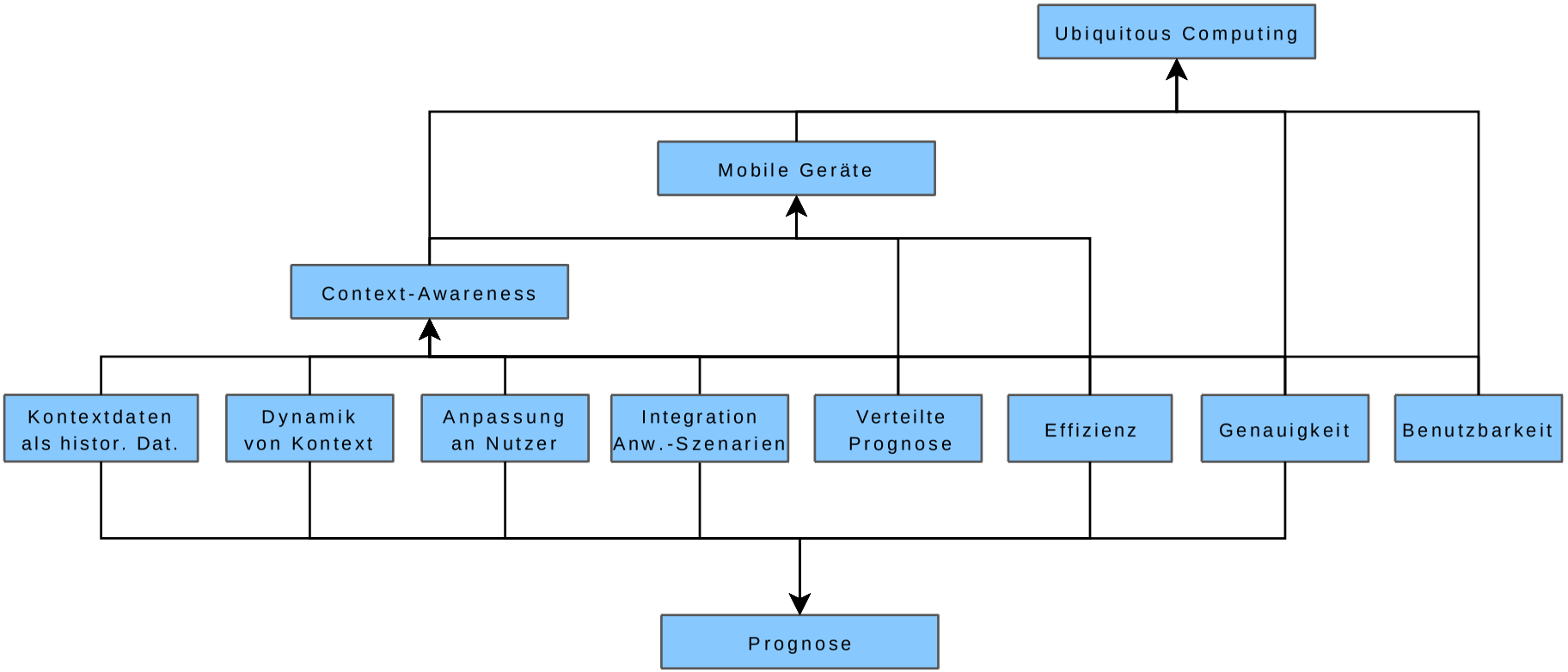

Abbildung 2.10.: vereinfachte Darstellung der Verweise zwischen den Abschnitten 2.5.2, 2.5.3, 2.5.4, 2.5.5, 2.5.6, 2.5.7, 2.5.8, 2.5.9 als Repräsentanten der Anforderungsgruppen (Mitte) und den Abschnitten 2.1, 2.2, 2.3, 2.4 (oben und unten)

Umgang mit Kontextdaten als historische Daten:

Nr.	Anforderung	Verweise
1.1	Unterstützung von Zusammenhangsarten: 1.1.1 Trends 1.1.2 Sequentielle Muster 1.1.3 Periodische Muster 1.1.4 Lineare Zusammenhänge 1.1.5 Nichtlineare Zusammenhänge 1.1.6 Gemischte Zusammenhänge	Abschnitt 2.5.2, Abschnitt 2.4.6
1.2	Direkte Unterstützung von Skalenniveaus: 1.2.1 Nominal 1.2.2 Ordinal 1.2.3 Metrisch	Abschnitt 2.5.2, Abschnitt 2.4.4
1.3	Unterstützung von Entitätsarten: 1.3.1 Menschen (z.B. Nutzer, soziale Umgebung) 1.3.2 Technik (z.B. Geräte, Netzwerke)	Abschnitt 2.5.2, Abschnitt 2.3.3
1.4	Speziell optimierte Unterstützung von primärem Kontext	Abschnitt 2.5.2, Abschnitt 2.3.3
1.5	Berücksichtigung der Abhängigkeit des sekundären Kontextes von primärem	Abschnitt 2.5.2
1.6	Unterstützung von horizontaler Verarbeitung auf unterschiedlichen Abstraktionsebenen: 1.6.1 Low-Level-Kontext 1.6.2 High-Level-Kontext	Abschnitt 2.5.2
(1.7)	Vertikale Verarbeitung	Abschnitt 2.5.2

Umgang mit Dynamik von Kontext:

Nr.	Anforderung	Verweise
2.1	Unterstützung von Instanzen (z.B. von Entitäten und Beziehungen)	Abschnitt 2.5.3
2.2	Unterstützung von aktiven Elementen	Abschnitt 2.5.3, Abschnitt 2.5.5

Anpassung an Nutzer:

Nr.	Anforderung	Verweise
3.1	Unterstützung von adaptivem Online-Lernen	Abschnitt 2.5.4, Abschnitt 2.5.7
3.2	Unterstützung bei Erstellung von Prognosemodellen	Abschnitt 2.5.4
3.3	Möglichkeit des Einbringens von A Priori - Wissen 3.3.1 Durch Offline-Batch-Lernen 3.3.2 Durch deduktives Lernen	Abschnitt 2.5.4, Abschnitt 2.5.6, Abschnitt 2.5.7

Integration in Anwendungsszenarien:

Nr.	Anforderung	Verweise
4.1	Plattformunabhängigkeit	Abschnitt 2.5.5
4.2	Universelle Schnittstellen für Anwendungen und bestehende Kontextsysteme	Abschnitt 2.5.5
4.3	Eigenes, universelles Kontext-Metamodell	Abschnitt 2.5.5
4.4	Unterstützung von Rollen der Zeit in gesuchten Informationen: 4.4.1 Inform. über Variablen/Ereignisse gesucht 4.4.2 Zeitpunkt gesucht	Abschnitt 2.5.5, Abschnitt 2.4.11
4.5	Unterstützung von Informationen über Variablen: 4.5.1 Wahrscheinlichkeitsverteilung 4.5.2 Modalwert 4.5.3 Wahrscheinlichkeit zu logischer Formel 4.5.4 Erwartungswert und Varianz	Abschnitt 2.5.5, Abschnitt 2.4.11

Genauigkeit:

Nr.	Anforderung	Verweise
5.1	Fundierte Angabe prinzipieller Unsicherheit	Abschnitt 2.5.6
5.2	Hohe Generalisierungsfähigkeit	Abschnitt 2.5.6
5.3	Genug Trainingsdaten lernen	Abschnitt 2.5.6
5.4	Fundierte Angabe von Schätzunsicherheit	Abschnitt 2.5.6, Abschnitt 2.5.4
5.5	Lernen stabiler Inseln von Zusammenhängen	Abschnitt 2.5.6, Abschnitt 2.5.3, Abschnitt 2.5.7
5.6	Berücksichtigung der Bedeutung räumlich und zeitlich nahen Kontextes	Abschnitt 2.5.6
5.7	Einbeziehen zusätzlicher Informationen	Abschnitt 2.5.6
5.8	Versuch der Angabe von Variationsunsicherheit	Abschnitt 2.5.6
5.9	Versuch der Anpassung an Änderungen von Zusammenhängen	Abschnitt 2.5.6
5.10	Minimierung von Methodenunsicherheit	Abschnitt 2.5.6
5.11	Angabe von Methodenunsicherheit	Abschnitt 2.5.6
5.12	Umgang mit Mängeln in Trainingsdaten: 5.12.1 Fehlerhafte Werte 5.12.2 Fehlende Werte 5.12.3 Subjektive Einschätzungen	Abschnitt 2.5.6, Abschnitt 2.5.8, Abschnitt 2.5.4
5.13	Konsistenz zu ähnlicher Zeit erstellter Prognosen	Abschnitt 2.5.6

Effizienz:

Nr.	Anforderung	Verweise
6.1	Abstraktion beim Lernen / Aggregation	Abschnitt 2.5.7
6.2	Berücksichtigung, welche Prognosen insgesamt von Anwendung benötigt werden	Abschnitt 2.5.7
6.3	Angepasste Aufwandsverteilung auf Lernen und Prognose für Minimierung von Ressourcenaufwand	Abschnitt 2.5.7
6.4	Begrenzbarkeit verwendeter Ressourcen	Abschnitt 2.5.7
6.5	Vermeidung unnötigen Lernens	Abschnitt 2.5.7
6.6	Prognosen mit stabil geringen Reaktionszeiten	Abschnitt 2.5.7
6.7	Effizienter Umgang mit großen Mengen an Variablen	Abschnitt 2.5.7
6.8	Skalierbarkeit	Abschnitt 2.5.7

Verteilte Prognose:

Nr.	Anforderung	Verweise
7.1	Umgang mit heterogenen Trainingsdaten aus Messungen ohne abgestimmte Messzeitpunkte	Abschnitt 2.5.8
7.2	Angabe der Fähigkeiten des Gerätes	Abschnitt 2.5.8
7.3	Ermöglichung der Wahl zwischen verschiedenen Verteilungsoptionen	Abschnitt 2.5.8
7.4	Vorsehen von Lösungsansätzen für instabile Vernetzung (z.B. asynchrone Kommunikation)	Abschnitt 2.5.8
(7.5)	Effizienzoptimierungen für durch Publish-Subscribe abonnierte Prognosen	Abschnitt 2.5.8, Abschnitt 2.5.5

Benutzbarkeit:

Nr.	Anforderung	Verweise
8.1	Geringer Vorbereitungsaufwand für Anwendungsentwickler	Abschnitt 2.5.9
8.2	Geringer Aufwand beim Einsatz für Anwendungsentwickler	Abschnitt 2.5.9
8.3	Unaufdringlichkeit für Nutzer	Abschnitt 2.5.9
8.4	Nachvollziehbarkeit für Nutzer	Abschnitt 2.5.9

Die Anbindung des Prognosesystems an bestehende Kontextsysteme durch konkrete Schnittstellen und die verteilte Prognose sollen zwar im Zuge dieser Arbeit realisiert werden, liegen aber außerhalb des *Schwerpunktes dieser Arbeit*. Im weiteren Verlauf der Arbeit werden Methoden zur Prognose den Ausgangspunkt bilden, da sie den Kern von Kontextdatenprognose ausmachen und nicht als trivial anzusehen sind. Dabei stehen Kontextdaten als historische Daten, die Dynamik von Kontext, die Anpassung an den

Nutzer, Genauigkeit, Effizienz, teilweise die Integration in Anwendungsszenarien und die Generik als wichtiges übergeordnetes Ziel ↑ im Mittelpunkt.

Die konkreten Anforderungen des Anforderungskatalogs wurden im bisherigen Verlauf der Arbeit von der Zielsetzung der Arbeit unter Zuhilfenahme der Literatur abgeleitet. Sie sollen eine nachvollziehbare Prüfung von Prognosesystemen auf deren Eignung für Kontextdatenprognose auf mobilen Geräten und auf deren Erreichung der Ziele dieser Arbeit ermöglichen. Die geforderte Genauigkeit und die geforderte Effizienz werden in der Evaluation der Ergebnisse dieser Arbeit ergänzend zum Anforderungskatalog auch quantitativ geprüft.

Die konkreten Anforderungen zu Genauigkeit und Effizienz im Anforderungskatalog stützen sich insbesondere auf die in Abschnitt 2.4 entwickelte universelle Sichtweise auf Prognose, die eine von konkreten Verfahren unabhängige Betrachtung ermöglichen soll, um zu harte konkrete Anforderungen und das Verwerfen guter Verfahren durch vorzeitige Festlegung auf konkrete Optionen der Realisierung von Prognose zu vermeiden. Es wird außerdem davon ausgegangen, dass dieses Kapitel über das Problemfeld, aus dem die konkreten Anforderungen hervorgehen, die wesentlichen Bereiche abdeckt, die in der Literatur zur Kontextdatenprognose Erwähnung finden, wobei allerdings der Einsatz auf mobilen Geräten im Allgemeinen nicht in der Literatur vorausgesetzt wird. Mayrhofer hat in [May05, S. 33-34] „Issues“ zur Kontextdatenprognose, aus denen sich Anforderungsbereiche ablesen lassen, aus der Forschung zusammengetragen[5]. Sie werden alle in diesem Kapitel berücksichtigt. Zudem wurde nach dem Verfassen des Anforderungskatalogs festgestellt, dass er alle sechs Bewertungskriterien von Michalski et al. für Methoden des maschinellen Lernens[6] als Anforderungsgruppen oder konkrete Anforderungen berücksichtigt [MBK98a, S. 29-31]. Damit sollte gewährleistet sein, dass möglichst wenig konkrete Anforderungen im Anforderungskatalog fehlen und dadurch keine unzureichenden Verfahren als geeignet eingestuft werden.

Der Anforderungskatalog entkoppelt die folgenden Kapitel von weiten Teilen dieses Kapitels über das Problemfeld. Sie können auf den Anforderungskatalog als zentrales Ergebnis der Problemanalyse zurückgreifen.

[5] Die „Issues“ lauten: accuracy, fault tolerance (meint fehlende und falsche Messwerte), unobtrusive operation, user acceptance, privacy, supervised vs. unsupervised, automatic vs. manually assisted (meint a priori bekanntes Wissen), problem complexity, uncertainty, online processing, heterogeneity (von Messwerten).

[6] Die Bewertungskriterien lauten: accuracy, efficiency, robustness against noise and incompleteness, special requirements (incrementality [meint adaptives Lernen], concept drift), concept complexity (representational issues, meint Ausdrucksmächtigkeit von Modellen/Metamodellen), transparency (comprehensibility for the human user).

3

Bestehende Lösungsmöglichkeiten

In Kapitel 2 wurde das Problem der Realisierung einer generischen Kontextdatenprognose auf mobilen Geräten dargestellt und es wurden Anforderungen an eine Lösung des Problems abgeleitet. Dieses Kapitel untersucht bestehende Lösungsmöglichkeiten und vergleicht diese anhand der Anforderungen des Anforderungskatalogs. Dazu wird zunächst ein Überblick über relevante Wissenschaftszweige und deren Herangehensweisen geschaffen, die Lösungsmöglichkeiten zugrunde liegen. Es wird unterschieden zwischen konkret ausgearbeiteten, umfassenden Lösungsmöglichkeiten zur Kontextdatenprognose (*Verfahren*) und allgemeineren, oft formalisierten oder als Algorithmen formulierten Lösungsmöglichkeiten zur Zukunfts- oder Folgerungsprognose wie z.B. neuronale Netze oder Entscheidungsbäume (*Methoden*). Eine Methode beinhaltet ein Prognosemetamodell, das zu einem wesentlichen Teil das Prognosemodell bestimmt, und bietet die Möglichkeit zur Prognose und meist auch des Lernens. Es wird zunächst überprüft, welche kompletten, fertig ausgearbeiteten Verfahren zur Kontextdatenprognose existieren, inwieweit Teile davon verwendet werden können und welche Konsequenzen aus den Resultaten der wesentlichen Prinzipien der Verfahren gezogen werden können. Aufgrund der Defizite der Verfahren erfolgt ein Vergleich und eine Auswahl von Methoden als weniger weit ausgearbeitete Lösungsmöglichkeiten. Diese sollen im Zuge der wichtigsten aus der Betrachtung kompletter Verfahren gezogenen Konsequenz hybrid zusammen eingesetzt werden, was zu der Erschließung von entsprechenden Ansätzen führt.

3.1. Relevante Wissenschaftszweige

Dieser Abschnitt stellt für die Arbeit relevante Wissenschaftszweige vor. Dazu werden grundlegende Herangehensweisen der Wissenschaften und übliche Klassifizierungen ihrer Methoden präsentiert. Die spezifische Terminologie der Wissenschaftszweige wird nur verwendet, wenn in dieser Arbeit noch keine äquivalenten Begriffe eingeführt wurden.

3.1.1. Data Mining

Data Mining ist motiviert durch die wachsenden Mengen an erhobenen und gespeicherten Daten heutzutage. Das Ziel besteht darin, wertvolles Wissen aus massiven Datenmengen zu extrahieren [Bra07a, S. 2-3] [SM05, S. 11]. Es geht also um das Lernen von Zusammenhängen in Trainingsdaten.

Insgesamt kann man Data Mining eher als *anwendungsorientiert* bezeichnen [LOL+05, S. V] [SM05, S. 17]. Data Mining ist also eher für praktische Aspekte von Nutzen. Die Anwendungsorientierung findet auch darin Ausdruck, dass *Prozesse* existieren, die den Einsatz von Data Mining beschreiben [Lar06, S. xiii]. Abbildung 3.1 bildet einen solchen Prozess auf in Kapitel 2 identifizierte prinzipielle Aktivitäten für Kontextdatenprognose

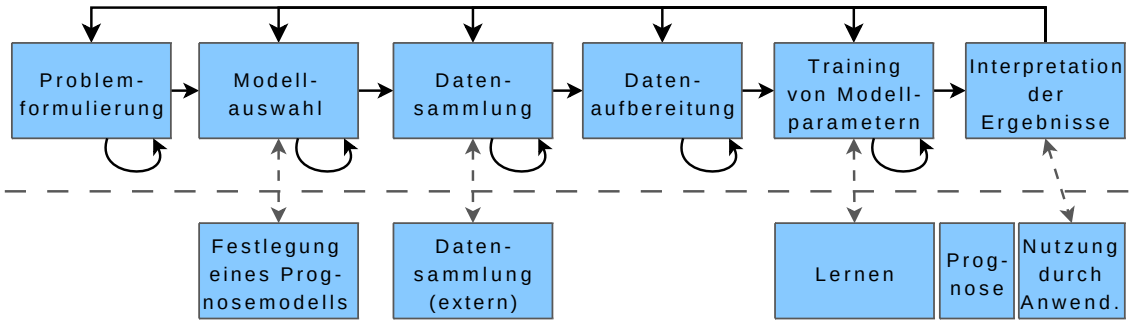

Abbildung 3.1.: Data Mining - Prozess nach Menzel [Men07, Teil 2a, S. 11] (oben) und in Kapitel 2 eingeführte Äquivalente (unten)

ab. Die Datensammlung erfolgt bei der Kontextdatenprognose durch die Anwendung bzw. das Kontextsystem ↑. Eine Prognose ist im allgemeinen Data Mining - Prozess nicht explizit vorgesehen.

Data Mining bietet unter Anderem Techniken zur *Datenaufbereitung* wie z.B. zur Dimensionsreduktion, Verkleinerung von Wertemengen und zur Reduzierung der Anzahl an Datensätzen [Bra07a, S. 14-19] [SM05, S. 18-21]. Eine Datenaufbereitung wird nicht explizit als Aktivität zur Kontextdatenprognose berücksichtigt, da die Zielsetzung der Arbeit schon in gewissem Maße vorverarbeitete Daten als Eingabe des Prognosesystems voraussetzt. Der Anforderungskatalog fordert jedoch das Behandeln von Mängeln in Trainingsdaten ↑ und ihrer Heterogenität ↑. Fraglich ist, inwieweit eine Datenaufbereitung sinnvoll zur Laufzeit eingesetzt werden kann und inwieweit sie zur Entwicklungszeit erfolgen sollte [Bra07a, S. 3] [SM05, S. 12-13]. Was Data Mining - Methoden betrifft, wird in der Literatur in der Regel nur die Möglichkeit des Batch-Lernens dargestellt.

Data Mining liegt eine allgemeine *Sichtweise auf Daten* zugrunde, die Daten als Werte von Variablen betrachtet und den in dieser Arbeit definierten historischen Daten ähnelt [Bra07a, S. 4]. Es kann auf die große Mehrheit an Datenorganisationsschemata angewendet werden [SM05, S. 15].

Data Mining beinhaltet induktives Lernen, bei dem Beispiele für Variablenwerte als Datensätze von Trainingsdaten gelernt werden [Bra07a, S. 4]. Ein Datensatz könnte z.B. die Qualität einer von vielen gerade produzierten Photovoltaik-Anlagen ↑ beschreiben oder Kontextdaten zu einem bestimmten Kontextmodell darstellen. Ziel ist das Lernen der Zusammenhänge im konkreteren Sinne des Wortes, die zwischen Variablen bestehen.

Die wahrscheinlich verbreitetste *Unterteilung von Data Mining - Methoden* unterscheidet auf oberster Ebene nach der Art des Lernens [Bra07a, S. 4]:

- überwacht lernend
  - Klassifizierung
  - Regression
- unüberwacht lernend
  - Assoziationsregeln
  - Clustering

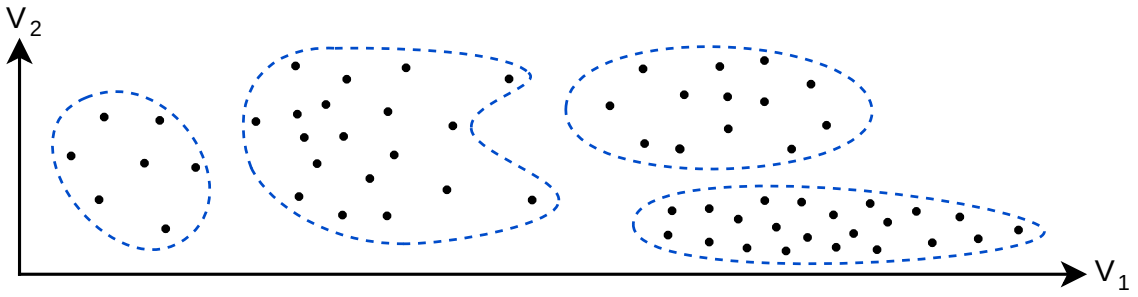

Abbildung 3.2.: typische Darstellung von Clustering wie z.B. in [SM05, S. 26] und [Bra07a, S. 222]

Bramer verwendet die in Abschnitt 2.4.3 eingeführten Begriffe des überwachten und unüberwachten Lernens. Bei Klassifikation und Regression ist also vorgegeben, welche Variablenwerte bei der Prognose aus anderen bestimmt werden können sollen, bei Assoziationsregeln und Clustering nicht [Bra07a, S. 4]. Klassifikation und Regression unterscheiden sich im Skalenniveau der abhängigen Variablen, dessen Bestimmung erlernt werden soll. Bei nominalem Skalenniveau handelt sich um *Klassifizierung*, bei metrischem um *Regression* [Bra07a, S. 4, 13]. Im Fall von Klassifizierung können die möglichen Werte der abhängigen Variablen als Klassen aufgefasst werden. Regression kann z.B. angewendet werden, um die Dauer eines Telefonats vorherzusagen ↑. Durch Klassifizierung könnte man vorhersagen, ob in einer bestimmten Situation ein Dienst verfügbar ist ↑.

Beim *Clustering* werden wie bei der Klassifizierung unabhängigen Variablen Gruppen zugeordnet, die als Cluster bezeichnet werden. Der Unterschied zur Klassifizierung liegt darin, dass bei Klassifizierung die korrekten Klassen in den Trainingsdaten im Zuge des überwachten Lernens vorgegeben sind. Beim Clustering gibt es gar keine korrekten Cluster. Es wird erst von der Clustering-Methode festgelegt, welche Cluster es überhaupt gibt. Das Ziel von Clustering liegt darin, Datensätze genau dann gleichen Clustern zuzuordnen, wenn sie ähnliche Variablenwerte besitzen [Bra07a, S. 221]. Da die Anzahl an Clustern normalerweise erheblich geringer ist als die Anzahl an Wertekombinationen der unabhängigen Variablen, kann man Clustering zur Aggregation von Daten nutzen, indem man die Angabe eines Clusters als Ersatz zu den Werten der unabhängigen Variablen ansieht. Aggregation kann entsprechend dem Anforderungskatalog beim Lernen als eine Möglichkeit zur Verringerung des Speicherbedarfs verwendet werden ↑. Clustering kann veranschaulicht werden durch die Vorstellung eines Datenwürfels mit den Variablen als Dimensionen und den Datensätzen als Datenpunkten im Würfel [Bra07a, S. 222]. Abbildung 3.2 zeigt dies für den Fall von zwei Variablen. Es wurden vier Cluster gebildet und jeder Datenpunkt einem Cluster zugeordnet. Die vier Cluster bilden die Werte einer neuen Variablen.

Assoziationsregeln als zweite Form unüberwacht lernender Methoden geben beliebige Zusammenhänge zwischen Variablen an [Bra07a, S. 7]. Es ist nicht festgelegt, zwischen welchen Variablen nach Zusammenhängen gesucht wird.

Bei den bisher vorgestellten Arten von Methoden handelt es sich um Methoden der Folgerungsprognose. Nicht zuletzt befasst sich Data Mining auch mit Daten, die die Zeit

berücksichtigen [SM05, S. 16] [MBK98b, S. 85].

Die vorgestellte Klassifizierung bietet einen guten Überblick über zur Prognose einsetzbare Methoden. Die Sichtweise des Data Mining steht in enger Verbindung zu dieser Arbeit. Neben den an der Praxis orientierten Prognosemethoden kommen unter Anderem auch Ansätze zur Datenaufbereitung zur Nutzung in Frage.

3.1.2. Maschinelles Lernen

Maschinelles Lernen (*Machine Learning*) ist im Bereich der künstlichen Intelligenz angesiedelt und steht im Bezug zur Tradition von Expertensystemen. Expertensysteme ersetzen den Vorgang des Programmierens durch die Aufgabe der Wissensaquisition und der Wissenskodierung [MBK98a, S. 4]. Die Idee des (induktiven) Lernens ersetzt diese Aufgabe wiederum durch das Bereitstellen von Trainingsdaten [MBK98a, S. 4] (vgl. auch Abschnitt 2.4.3). In Anlehnung an Michalski et al. unterscheidet diese Arbeit zwischen wissensorientierten Methoden, die das gewonnene Vorwissen in einer prinzipiell für den Nutzer verständlichen Form darstellen können und der symbolischen künstlichen Intelligenz zugeordnet sind, und nicht wissensorientierten Methoden [MBK98a, S. 5].

Wissensorientierte Methoden kommen der geforderten Nachvollziehbarkeit ↑ von Prognose entgegen. Das Ziel *wissensorientierter Methoden* liegt häufig in der Bestimmung von *Konzeptbeschreibungen* aus den Trainingsdaten [MBK98a, S. 5]. Ein Konzept beschreibt eine Mengen von Objekten mit bestimmten gemeinsamen Eigenschaften [MBK98a, S. 6]. Die Bestimmung der Konzeptzugehörigkeit als abhängige Variable mit nominalem Skalenniveau entspricht einer Klassifizierung, wobei ein Objekt aber auch mehreren Konzepten angehören kann, so dass man eine binäre abhängige Variable für jedes mögliche Konzept benötigt [MBK98a, S. 5]. Zudem existieren im Bereich des *Artificial Discovery* unüberwacht lernende Methoden, die als spezielle Varianten von Clustering und Assoziationsregeln angesehen werden können [MBK98a, S. 42-53].

Der *nicht wissensorientierte Zweig* des maschinellen Lernens weist z.B. bei der Sichtweise auf die Daten und der möglichen Datenaufbereitungsphase große Ähnlichkeiten zum Data Mining auf [Bis06, S. 2]. Es wird sogar bei Bishop zwischen überwachtem und unüberwachtem Lernen sowie Klassifizierung und Regression unterschieden [Bis06, S. 3]. Aufgrund ihrer Verbreitung werden insbesondere die Begriffe der Klassifizierung und Regression im Folgenden unabhängig von Wissenschaftszweigen verwendet werden.

Im Bereich des maschinellen Lernens und auch in anderen Wissenschaftszweigen wird (induktives) Lernen häufig als Problem angesehen, das durch *Suche* nach dem besten Vorwissen gelöst werden kann [MBK98a, S. 14]. Die Motivation für eine Suche liegt darin, dass das Vorwissen in der Praxis meist nicht wie bei der universellen Sichtweise in Abschnitt 2.4.9 durch Schätzen eines Wahrscheinlichkeitsmaßes direkt bestimmt werden kann ↑. Das Ermitteln von Vorwissen durch Suche löst das Problem der Ermittlung von gutem Vorwissen aus Trainingsdaten, indem es unterschiedliches Vorwissen mehr oder weniger systematisch wählt und dann durch eine Bewertungskriterium auf seine Güte prüft [MBK98a, S. 14-16]. Die Güte von Vorwissen kann durch testweise Prognose der schon bekannten Werte der abhängigen Variablen in den Trainingsdaten bestimmt werden.

Zusammenfassend kann festgestellt werden, dass maschinelles Lernen über Data Mining hinausgehend auch wissensorientierte Methoden anbietet. Zudem spielt die Sichtweise der Suche für das Lernen eine Rolle.

3.1.3. Mustererkennung

Mustererkennung (*Pattern Recognition*) ist vor einigen Jahrzehnten aus der Statistik erwachsen und mittlerweile eng mit technischen Anwendungen und der künstlichen Intelligenz verbunden [TK06, S. 1]. Mustererkennung findet beispielsweise bei der Erkennung geschriebener Buchstaben, medizinischer Diagnose und der Vorhersage von Erdbeben Anwendung [TK06, S. 1,3]. Mustererkennung beschäftigt sich primär mit Klassifizierung [DGL97, S. 1] und besitzt abgesehen vom zeitlichen Bezug eine ähnliche Sichtweise auf Daten wie die Definition historischer Daten in dieser Arbeit.

Der Prozess der Mustererkennung bezieht wie Data Mining eine Datenaufbereitungsphase, insbesondere eine Dimensionsreduktion ein [TK06, S. 6], wobei man eher noch weiter geht als beim Data Mining. Daher kommt dem Begriff des *Merkmals* (*feature*) eine besondere Bedeutung zu. Ein Merkmal bezeichnet eine aus ursprünglichen Variablen gewonnene neue Variable [TK06, S. 3-5]. Zum Extrahieren von Merkmalen werden Techniken wie die Fourier-Transformation eingesetzt [TK06, S. 263-326]. Es werden jedoch auch einfachere Merkmale verwendet wie z.B. die mittlere Intensität der Pixel eines Flecks auf einem Röntgenbild [TK06, S. 3-5]. Ein Fokus liegt auf der Bestimmung von Merkmalen zu Bild-, und Ton-Daten [TK06, S. 327-396], die auch als mögliche gemessene Daten im Bereich der Context-Awareness eine Rolle spielen. Prognosen auf solchen Daten wie z.B. die Vorhersage für einen Blinden, dass der Zusammenstoß mit einem beweglichen Objekt droht, beinhalten jedoch in erheblichem Maße vertikale Verarbeitung, die im Anforderungskatalog als optional eingestuft ist ↑, weil sie nur bedingt als Teil des Aufgabenbereichs von Zukunftsprognose angesehen wird ↑.

Mustererkennung ist von Nutzen für diese Arbeit, weil sie Klassifizierungsmethoden bereitstellt. Die Ansätze zur Datenaufbereitung sind nur bedingt relevant.

3.1.4. Stochastik

Gegenstand der *Stochastik* sind Vorgänge und Ereignisse, die vom Zufall beeinflusst sind [Hüb03, S. 1]. Ausgehend von diesem elementaren Thema bietet sie ein breites theoretisches Fundament und konnte daher auch in Abschnitt 2.4 als Grundlage für eine universelle Sichtweise auf Prognose und zur Beschreibung von *Unsicherheit* dienen. Im Gegensatz zur Stochastik wird Unsicherheit bei anderen Wissenschaftszweigen häufig nur in eingeschränktem Umfang berücksichtigt, z.B. durch Festlegung von Prognoseergebnissen auf Variablenwerte statt Wahrscheinlichkeitsverteilungen und die Annahme einer unabhängigen, identischen Verteilung (vgl. z.B. [Bra07a, S. 23-24] und [DGL97, S. 1-2]). Neben der Stochastik kommen prinzipiell auch die Dempster-Shafer-Theorie und die Fuzzy Logic mit der darauf aufbauenden *Possibility Theory* zum Umgang mit Unsicherheit in Frage [Wit02, S. 21] [BK02, S. 21]. Auf der Fuzzy Logic und Fuzzy Sets aufsetzend existieren sogenannte linguistische Methoden, die auf den Umgang mit Unschärfe, die

in der menschlichen Sprache Ausdruck findet, spezialisiert sind [LSR03].[1] Diese Theorien werden jedoch in dieser Arbeit nicht näher untersucht, da Wahrscheinlichkeiten ein sehr etabliertes Konzept darstellen und dadurch im Rahmen vieler Methoden eingesetzt werden können und der Anforderung nach geringem Lernaufwand für Anwendungsentwickler gerecht werden. Zudem wird bei Wittig nicht erkennbar, dass eine dieser beiden Theorien eindeutig gegenüber der Stochastik zu bevorzugen ist [Wit02, S. 21-23]. Der Einsatz linguistischer Methoden könnte jedoch weiterführend untersucht werden, da das Konzept der Unschärfe (in der menschlichen Sprache) das Konzept der Unsicherheit ergänzt.

Neben allgemeiner Theorie bietet die Stochastik auch Methoden, die allerdings häufig noch relativ viel Gestaltungsspielraum lassen [Hüb03, S. 4-5]. Auf Basis der Prognosemetamodelle von Methoden können Prognosemodelle für konkrete Anwendungen erstellt werden [Hüb03, S. 4-5]. Hübner stellt das Vorgehen in der Stochastik als das Bilden von (Prognose-)Modellen dar [Hüb03, S. 3-5]. Die Definition von Wahrscheinlichkeitsmodellen beschreibt ein Prognosemetametamodell [Hüb03, S. 11], dem Prognosemetamodelle von Methoden der Stochastik wie z.B. Markovketten und Bediennetzen entsprechen [Hüb03, S. 124-131, 153-157]. Markovketten und Bediennetze sind auch Beispiele dafür, dass die Stochastik die Bildung mehrstufiger Prognosemodelle ermöglicht, die eine Struktur aufweisen [Hüb03, S. 47-57]. Von besonderem Interesse sind stochastische Prozesse als eine Klasse von Prognosemetamodellen, die Zustände im zeitlichen Verlauf beschreiben [Hüb03, S. 123-132].

Zur Prognose stehen analytische Ansätze und Simulation zur Verfügung [Hüb03, S. 3]. Die Anwendung der Stochastik erfolgt leider ähnlich wie beim Data Mining in vielen Fällen offline, also ohne Lernen zur Laufzeit bei einem Anwendungsnutzer. Stattdessen werden z.B. bei der Qualitätskontrolle von Photovoltaik-Anlagen ↑ oder der Analyse der Transaktionen eines Unternehmens beim Data Mining die Prognosemodelle vom gleichen Personenkreis angewendet, von dem sie auch erstellt wurden. Daher muss man damit rechnen, dass adaptives Online-Lernen nicht mit allen Methoden (einfach) möglich ist. Letztendlich ist daher in erster Linie der theoretische Aspekt der Stochastik zum Umgang mit Unsicherheit von Interesse.

3.1.5. Statistik

Statistik befasst sich mit beobachteten Daten [Hüb03, S. 173], die z.B. aus Befragungen, Zählungen oder Messungen stammen können. Es wird unterschieden zwischen *deskriptiver Statistik* und *schließender Statistik* [Ste07, S. 1, 135]. Die schließende Statistik umfasst die beiden Teilbereiche *Testen* und *Schätzen* (vgl. auch Abschnitt 2.4.9) [Dul08, S. 12].

Die *deskriptive Statistik* dient der Beschreibung von Daten durch Tabellen, Grafiken und Kenngrößen sowie dem Erkennen möglicher Zusammenhänge in Daten [Ste07, S. 1] und stellt daher ein Hilfsmittel bei der Erstellung von Prognosemodellen dar. Dieses Thema wird in der Arbeit jedoch nicht fokussiert. Die *schließende Statistik* führt schließlich die Erstellung von (stochastischen) Prognosemodellen durch und füllt Prognosemodelle durch Lernen aus den Daten als Trainingsdaten mit weiterem Vorwissen auf der Instanzebene

[1] Dank an Dr. Magdalena Kotnis von der Universität Stettin für Hinweise in diese Richtung!

[Ste07, S. 135]. Die (schließende) Statistik beschäftigt sich also ergänzend zur Stochastik mit dem Lernaspekt von Methoden [Hüb03, S. 173], während sich die Stochastik eher mit den Prognosemetamodellen und der Prognosefähigkeit von Methoden befasst. Das Lernen von Vorwissen auf der Instanzebene kann sich entweder auf bestimmte Parameter des Prognosemodells wie z.B. die Varianz einer Normalverteilung beschränken (*parametrische Statistik*) oder ganze Wahrscheinlichkeitsverteilungen umfassen (*nichtparametrische Statistik*) [Ste07, S. 136]. Nichtparametrische Prognosemodelle sind von Interesse, wenn eher wenig Vorwissen durch das Prognosemodell festgelegt und viel Vorwissen beim Lernen gebildet werden soll. Leider erfolgt das Lernen bei vielen Anwendungen der Statistik offline (z.B. bei wissenschaftlichen Studien).

Ein besonderer Vorzug der schließenden Statistik und ihrer theoretischen Fundierung liegt in der Berücksichtigung von Schätzunsicherheit. Variationsunsicherheit wird leider in der Statistik in der Regel von der Betrachtung ausgeschlossen, indem eine unabhängige, identische Verteilung vorausgesetzt wird (z.B. in [Ste07, S. 137]).

Es existieren zwei große Denkschulen in der Statistik, die den Begriff der Wahrscheinlichkeit in ihrer Bedeutung im Bezug auf die reale Welt unterschiedlich interpretieren [Wit02, S. 91]. Die in Abschnitt 2.4 zugrunde gelegte *frequentistische Denkschule*, die in streng (natur-)wissenschaftlichen Bereichen ihre Bedeutung hat, interpretiert Wahrscheinlichkeiten als physikalische Eigenschaft [Wit02, S. 91]. Der Weg zur Ermittlung von Wahrscheinlichkeiten führt nach dieser Sichtweise über das Wiederholen von Zufallsexperimenten [Wit02, S. 91]. Der *bayes'sche Ansatz* nach Thomas Bayes sieht Wahrscheinlichkeiten als Maß für subjektive Einschätzungen einer Person an [Wit02, S. 91]. Dadurch wird die Anforderung der Unterstützung subjektiver Einschätzungen erfüllt ↑. Da Wahrscheinlichkeiten von Ereignissen im Sinne der Frequentisten von Personen als deren subjektive Einschätzung übernommen werden können [Wit02, S. 91], umfasst die bayes'sche Interpretation von Wahrscheinlichkeit die Möglichkeiten der frequentistischen. Wahrscheinlichkeitstheoretische Definitionen, Formeln und Theoreme gelten nach dem Wissen des Autors jedoch unabhängig von diesen beiden Sichtweisen, selbst wenn das Wort „Bayes" im Namen enthalten ist wie z.B. beim Satz von Bayes. Solange durch die bayes'sche Sichtweise kein deutlicher Vorteil entsteht, wird diese Arbeit bei der frequentistischen Sichtweise verbleiben.

Eine wesentliche praktische Bedeutung der beiden Denkschulen liegt in ihren *Lernansätzen* [Wit02, S. 92], die explizit zum Lernen angewendet werden oder implizit in Lernstrategien konkreter Methoden aufgegangen sein können [Dul08, S. 13]. Der frequentistische Ansatz, der nach Davison üblicher bei Anwendungen ist [Dav03, S. 52], besteht in der Maximierung der sogenannten *Likelihood-Funktion* [Wit02, S. 92], die in angepasster Schreibweise $L(D;\vartheta)$ lautet [Dul08, S. 13] [Wit02, S. 92] [Ste07, S. 140]. Mit D werden die Trainingsdaten bezeichnet. ϑ ist ein als Teil des Vorwissens zu lernender Parameter [Dul08, S. 12-13]. Der Wert der Likelihood-Funktion kann vereinfacht gesagt (genau genommen bei diskreten Daten) als Wahrscheinlichkeit des Auftretens der Trainingsdaten bei einem bestimmten Parameterwert ϑ interpretiert werden [Dul08, S. 13]. Der Parameterwert ϑ wird nun nach dem *Maximum-Likelihood-Prinzip* so gewählt, dass der Wert der Likelihood-Funktion bei gegebenen Trainingsdaten maximal wird, so dass Prognosen

mit den Trainingsdaten so gut wie möglich werden. Durch die Optimierung in Bezug auf die Trainingsdaten wird allerdings nicht die Anforderung der Generalisierungsfähigkeit berücksichtigt.

Das Maximum-Likelihood-Prinzip spielt bei allgemeinen Lernalgorithmen wie dem *Expectation-Maximization-Algorithmus* als prominentem Vertreter eine Rolle. Solche Algorithmen erfüllen die Anforderung des Umgangs mit fehlenden Werten in Trainingsdaten [Wit02, S. 98] [BK02, S. 137]. Es ist jedoch unklar, ob der Expectation Maximization - Algorithmus die Forderungen nach Skalierbarkeit und Effizienz in ausreichendem Maße erfüllt [BK02, S. 140-141]. Zudem stuft Wittig ihn nicht als adaptiv ein [Wit02, S. 98, 104]. Daher muss bei der Erarbeitung eines geeigneten Verfahrens wahrscheinlich auf einfachere Lösungen zurückgegriffen werden.

Abschließend kann festgestellt werden, dass die Statistik insbesondere im Zusammenhang mit Lernen nützliche, präzise Mittel zum Umgang mit Unsicherheit bereitstellt. Von den beiden Denkschulen der Statistik wird auf die frequentistische zurückgegriffen.

3.1.6. Zeitreihenanalyse

Zeitreihenanalyse (*Time Series Analysis*) beschäftigt sich im Gegensatz zu den vorhergehenden Wissenschaftszweigen mit Zukunftsprognose statt Folgerungsprognose. Sie hat sich als Anwendungsgebiet der Stochastik etabliert und kommt beispielsweise in den Bereichen Physik, Technik und den Wirtschaftswissenschaften zur Analyse und zur Prognose zeitlicher Verläufe zum Einsatz [ABH85, S. 7].

Die übliche Darstellung der Trainingsdaten in der Zeitreihenanalyse dient als Grundlage für die Definition historischer Daten in Abschnitt 2.4.4. Die Trainingsdaten liegen nicht in Form mehrerer Datensätze vor, die durch die Wiederholung eines Zufallsexperiments entstanden sind, sondern als historische Daten über einen möglicherweise langen Zeitraum. Eine Aufteilung in Zufallsexperimente muss implizit oder explizit beim Lernen erfolgen. Man unterscheidet zwischen multivariater Zeitreihenanalyse und univariater Zeitreihenanalyse, die sich mit dem Spezialfall nur einer Variablen befasst [ABH85, S. 7]. In Abschnitt 2.4.6 wurden bereits eine Reihe von möglichen Zusammenhängen in historischen Daten genannt, von denen in der Literatur zur Zeitreihenanalyse zumindest Trends [ABH85, S. 8-9], periodische Muster [ABH85, S. 9, „Saisonkomponente“], deterministische Zusammenhänge [And71, S. 2], gemischte Zusammenhänge [And71, S. 2] und lineare Zusammenhänge [KS97] thematisiert werden.

Eine abhängige Variable X kann als Summe mehrerer Anteile mit unterschiedlichen Zusammenhangsarten dargestellt werden, z.B. als Summe einer direkt vom Zeitpunkt abhängigen deterministischen Komponente f und einer nichtdeterministischen Komponente u [And71, S. 2]. Eine einfache, klassische Sorte von Prognosemetamodellen beschreibt eine Zeitreihe durch $x_j = f(j)+u_j$, wobei die nichtdeterministische Komponente u zufällige Abweichungen von der deterministischen Komponente f darstellt und eher als Störung interpretiert wird [And71, S. 3, 4]. Die deterministische Komponente spielt in solchen Prognosemetamodellen die wesentliche Rolle und kann zur Beschreibung von Trends und periodischen Zusammenhängen benutzt werden. Kompliziertere Prognosemetamodelle erlauben, dass sich nichtdeterministische Vorgänge auch auf die Zukunft

auswirken und so fortpflanzen können. Im Fall von nichtlinearen Zusammenhängen und entsprechenden Prognosemetamodellen kann dies zum bereits erwähnten Chaos führen [KS97, S. 3].

Die Zeitreihenanalyse setzt teilweise Stationarität voraus. Im Fall von Trends berücksichtigt sie aber Variationsunsicherheit [And71, S. vii, 1, 3]. Zeitreihenprognose ist insgesamt für diese Arbeit von Bedeutung, weil sie sich speziell mit Methoden der Zukunftsprognose beschäftigt.

3.1.7. Systemdynamik

Die *Systemdynamik* (*System Dynamics*) wird angewandt zur Beschreibung von z.B. sozialen und politischen Prozessen und des Verhaltens komplexer Systeme wie Unternehmen, deren Management durch die Systemdynamik unterstützt werden soll [Kir98, S. 1] [Ric99, S. 440]. Nach Kirkwood kann jeder Geschäftsprozess mit Mitteln der Systemdynamik charakterisiert werden [Kir98, S. 17]. Die genannten Anwendungen sprechen dafür, dass Ansätze der Systemdynamik der Anforderung der Unterstützung von Mensch und Technik als Kontextarten ↑ prinzipiell gerecht werden.

Die Systemdynamik steht im Bezug zur Systemtheorie [Kir98, S. 48-49]. Bei Kirkwood ist die Rede von Elementen und Struktur [Kir98, S. 18]. Struktur kann mit Prognosemetamodellen grafisch dargestellt werden. Es wurden leider keine Hinweise in der Literatur darauf gefunden, dass Unsicherheit in der Systemdynamik eine wesentliche Rolle spielt [Kir98, S. 22].

3.2. Verfahren

Es wurden in den vorhergehenden Abschnitten grundlegende Herangehensweisen von Wissenschaftszweigen vorgestellt, deren Gegenstand Prognose beinhaltet. Aufbauend auf derartigen Herangehensweisen wurden bereits komplette Verfahren zur Kontextdatenprognose ausgearbeitet. Es existieren eine Reihe von Projekten zum Thema Context-Awareness, die auch Verfahren zur Kontextdatenprognose beinhalten [May04, S. 19-32] [Sig08, S. 60-69] [Pet05, S. 10-12]. Sie wählen jedoch überwiegend eher spezialisierte Ansätze zur Kontextdatenprognose, die z.B. nur die Position prognostizieren können [May04, S. 19-32] [Sig08, S. 19-20] [Pet05, S. 11]. Solche Ansätze kommen wegen der generischen Ausrichtung dieser Arbeit ↑ nicht in Frage. Es existieren jedoch drei Dissertationen, die sich ausschließlich und in relativ allgemeiner Weise mit Kontextdatenprognose befassen und die entsprechende Verfahren entwickeln. Daher werden in diesem Abschnitt die Verfahren dieser drei Dissertationen von Mayrhofer [May04], Petzold [Pet05] und Sigg [Sig08] untersucht. Eine weitere Dissertation von Wittig [Wit02] wird an dieser Stelle nicht aufgegriffen, da sie sich nicht mit Kontextdatenprognose sondern mit dem ähnlichen Thema des Lernens für benutzeradaptive Systeme beschäftigt. Aus der Dissertation von Wittig fließen jedoch an verschiedenen Stellen Aspekte in diese Arbeit ein.

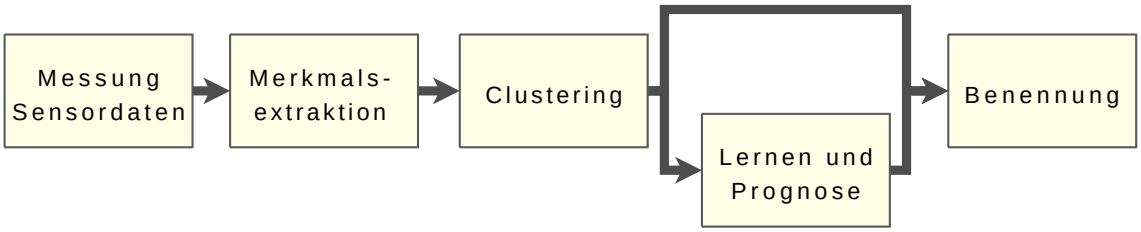

Abbildung 3.3.: Architektur des Verfahrens von Mayrhofer [May04, S. 38]

3.2.1. Verfahren von Mayrhofer

Die Dissertation von Mayrhofer stammt von 2004 und beschreibt ein Verfahren zur Kontextdatenprognose, dessen Implementierung in Zusammenarbeit mit Radi erfolgt ist. Mayrhofer verbindet Prognose mit vertikaler Verarbeitung, also der Ermittlung von *High-Level-Kontext* aus Low-Level-Kontext [May04, S. 5, 33, 62]. High-Level-Kontext wird im Verfahren von Mayrhofer aus den Werten möglichst vieler Sensoren ermittelt und als Zustand aufgefasst, der den gesamten Kontext abstrakt beschreibt und Bedeutungen tragen kann wie „zu Hause“, „in einer Besprechung“ und „etwas präsentierend“ [May04, S. 5, 33, 62]. Die Prognose besteht in der Vorhersage zukünftiger derartiger Zustände [May04, S. 37, 39-40].

Das Verfahren von Mayrhofer arbeitet in *fünf Schritten* (vgl. Abbildung 3.3). Bei der *Messung von Sensordaten* wird direkt auf Sensoren zugegriffen [May04, S. 78]. Es folgt die *Extraktion von Merkmalen* aus dem gemessenen Low-Level-Kontext [May04, S. 38]. Die Ermittlung von High-Level-Kontext ist durch *Clustering* realisiert [May04, S. 37]. Die *Benennung* dient der Zuordnung von Bezeichnern zu Clustern durch den Nutzer [May04, S. 40]. Bei der Prognose wird ein zukünftiger Cluster prognostiziert [May04, S. 40, 66, 76, 87]. Prinzipiell vorgesehen ist auch die Bestimmung der Wahrscheinlichkeit zukünftiger Cluster, die jedoch zum Zeitpunkt der Veröffentlichung der Dissertation nicht implementiert ist [May04, S. 40, 66, 76, 87]. Die gerade vorgestellten fünf Schritte bilden die mehrschichtige Architektur des Verfahrens von Mayrhofer [May04, S. 36, 38, 130]. Die Schichten fungieren als Filter, durch die die Daten nur in eine Richtung fließen können [May04, S. 37, 128]. Die *Methoden* für Merkmalsextraktion, Clustering und Prognose sind austauschbar und es wurden mehrere implementiert, aus denen eine gewählt werden kann [May04, S. 37, 77-78, 82, 86].

Das Verfahren ist auf einen *lokalen Betrieb* ausgelegt und berücksichtigt die *limitierten Ressourcen* eingebetteter und mobiler Geräte [May04, S. 30, 31, 37, 39, 130]. Es existieren jedoch auch erste Ansätze zum *verteilten Einsatz* [May04, S. 80, 87-88, 129]. Besonderes Augenmerk wurde gelegt auf die automatische *Anpassung* an Nutzer und Eignung für beliebige Anwendungen bei gleichzeitiger *Unaufdringlichkeit* und ohne Notwendigkeit umfangreicher Konfiguration [May04, S. 31, 39, 64, 128, 131]. Es wird insbesondere *adaptives Online-Lernen* unterstützt und die Menge der Cluster, die Low-Level-Kontext zugeordnet werden können, wird zur Laufzeit dynamisch an die Gewohnheiten des Nutzers angepasst [May04, S. 31, 37, 39, 40, 64].

Die folgende Tabelle zeigt, inwieweit das Verfahren von Mayrhofer die im Anforderungskatalog festgelegten Anforderungen erfüllt. Informationen über das Verfahren

werden ausschließlich aus der Dissertation von Mayrhofer [May04] bezogen. Eine Anforderung kann erfüllt sein („✔“), mit Einschränkungen erfüllt sein („(✔)“) oder nicht erfüllt sein („✘“). Ein Fragezeichen wird bei fehlenden Informationen gesetzt und deutet in vielen Fällen darauf hin, dass die Anforderung nicht bedacht wurde.

Nr.	Anforderung		Begründung / Kommentar
Kontextdaten als historische Daten			
1.1	Unterstützung von Zusammenhangsarten:		
1.1.1	Trends	(✔)	vorgesehen, nicht als Prognosemethode realisiert [May04, S. 68]
1.1.2	Sequentielle Muster	✔	Standard bei bestehenden Prognosemethoden [May04, S. 31, 72, 86-87]
1.1.3	Periodische Muster	✘	durch neue Prognosemethoden realisierbar [May04, S. 68, 76, 132-133]
1.1.4	Lineare Zusammenhänge	-	nicht relevant, da Cluster ohne metrisches Skalenniveau
1.1.5	Nichtlineare Zusammenhänge	-	nicht relevant, da Cluster ohne metrisches Skalenniveau
1.1.6	Gemischte Zusammenhänge	✘	wäre durch Methoden möglich, die dies unterstützen [May04, S. 70-75, 86-87]
1.2	Direkte Unterstützung von Skalenniveaus:		
1.2.1	Nominal	?	vermutlich unterstützt [May04, S. 37, 46, 48, 128]
1.2.2	Ordinal	✔	[May04, S. 37, 46, 48, 128]
1.2.3	Metrisch	✔	[May04, S. 37, 46, 48, 128]
1.3	Unterstützung von Entitätsarten:		
1.3.1	Menschen	✔	Situation des Nutzers als Cluster, Menschen über Umgebungsgeräusche erfasst [May04, S. 83]
1.3.2	Technik	✔	entsprechende Messungen und Merkmalsextraktionen [May04, S. 82-84]
1.4	Optimierungen primärer Kontext	✘	primärer Kontext nur implizit in Clustern
1.5	Nutzung des Zusammenhangs prim. Kont. → sekund. Kont.	✘	primärer und sekundärer Kontext nur implizit in Clustern
1.6	Horizontale Verarbeitung auf unterschiedlichen Abstraktionsebenen:		
1.6.1	Low-Level-Kontext	✘	[May04, S. 5, 33, 62]
1.6.2	High-Level-Kontext	✔	[May04, S. 5, 33, 62]
1.7	Vertikale Verarbeitung	✔	Clustering als vertikale Verarbeitung
Umgang mit Dynamik von Kontext			
2.1	Instanzen	(✔)	zur Laufzeit angepasste Menge verfügbarer Cluster [May04, S. 40]
2.2	Aktive Elemente	✘	[May04, S. 132]
Anpassung an Nutzer			
3.1	Adaptives Online-Lernen	✔	[May04, S. 31, 39, 64]
3.2	Unterstützung bei Erstellung von Prognosemodell	-	kein Prognosemodell vorhanden
3.3	Einbringen von A Priori - Wissen	✘	[May04, S. 64]
Integration in Anwendungsszenarien			
4.1	Plattformunabhängigkeit	(✔)	überwiegend plattformunabhängig, Compiler für Zielplattform nötig [May04, S. 79-80]

Nr.	Anforderung		Begründung / Kommentar
4.2	Universelle Schnittstellen	(✔)	kein Publish-Subscribe, Betrieb mit separatem Kontextsystem nur angedacht [May04, S. 78-79]
4.3	Universelles Kontext-Metamodell	✘	nur High-Level-Kontext [May04, S. 5, 33, 62]
4.4	Unterstützung von Rollen der Zeit in gesuchten Informationen:		
4.4.1	Informationen über Variablen/Ereignisse gesucht	✔	[May04, S. 35, 40, 65]
4.4.2	Zeitpunkt gesucht	(✔)	noch experimentell [May04, S. 75]
4.5	Unterstützung von Informationen über Variablen:		
4.5.1	Wahrscheinlichkeitsverteilung	(✔)	vorgesehen, nicht als Prognosemethode realisiert [May04, S. 31, 36, 65, 66, 76, 87]
4.5.2	Modalwert	✔	Standard bei bestehenden Methoden[May04, S. 66, 76, 87]
4.5.3	Wahrscheinlichkeit zu logischer Formel	✘	[May04, S. 40]
4.5.4	Erwartungswert und Varianz	✘	[May04, S. 40]
Genauigkeit			
5.1	Fundierte Angabe prinzipieller Unsicherheit	(✔)	vorgesehen, nicht durch Prognosemethoden realisiert [May04, S. 31, 36, 65, 66, 76, 87]
5.2	Hohe Generalisierungsfähigkeit	?	abhängig von Prognosemethode
5.3	Genug Trainingsdaten lernen	✔	wenig benötigt durch Clustering
5.4	Fundierte Angabe Schätzunsicherheit	✘	Unsicherheitsarten nicht unterschieden bei Prognoseergebnis [May04, S. 36, 40, 65]
5.5	Lernen stabiler Inseln von Zusammenhängen	(✔)	vermutlich gewisse Stabilität in High-Level-Kontext
5.6	Berücksichtigung räumlich und zeitlich nahen Kontextes	(✔)	vermutlich feineres Clustering häufiger auftretenden Kontextes, nach Clustering jedoch räuml.-zeitl. Inform. überwiegend verloren
5.7	Einbeziehen zusätzl. Inform.	(✔)	einige Sensoren nutzbar [May04, S. 82-84]
5.8	Angabe Variationsunsicherheit	✘	Unsicherheitsarten nicht unterschieden bei Prognoseergebnis [May04, S. 36, 40, 65]
5.9	Anpassung an Änderungen von Zusammenhängen	(✔)	Berücksichtigung von Trends vorgesehen, nicht als Prognosemethode realisiert [May04, S. 68]
5.10	Geringe Methodenunsicherheit	(✔)	Prognosemethoden unterstützen keine prinzipielle Unsicherheit [May04, S. 31, 36, 65, 66, 76, 87]
5.11	Angabe Methodenunsicherheit	✘	Unsicherheitsarten nicht unterschieden bei Prognoseergebnis [May04, S. 36, 40, 65]
5.12	Umgang mit Mängeln in Trainingsdaten:		
5.12.1	Fehlerhafte Werte	✘	[May04, S. 130]
5.12.2	Fehlende Werte	✔	[May04, S. 130]
5.12.3	Subjektive Einschätzungen	?	keine Angaben gefunden
5.13	Konsistenz mehrerer Prognosen	✔	keine Gefahr da durch Clustering nur eine Variable
Effizienz			
6.1	Abstraktion beim Lernen	✔	durch Clustering, zudem keine Speicherung kompletter Historien [May04, S. 5, 14, 33, 62]
6.2	Prognosebedarf berücksichtigen	?	keine Angaben gefunden

Nr.	Anforderung		Begründung / Kommentar
6.3	Angepasste Aufwandsverteilung auf Lernen und Prognose	✔	durch Wählbarkeit von Methoden [May04, S. 37, 77-78]
6.4	Begrenzbarkeit Ressourcennutzung	✘	[May04, S. 129]
6.5	Kein unnötiges Lernen	?	unklar [May04, S. 39]
6.6	Stabil geringe Reaktionszeiten	✔	zumindest angestrebt [May04, S. 36]
6.7	Effizienz bei vielen Variablen	✔	Clustering reduziert Variablen auf eine [May04, S. 5, 33, 62]
6.8	Skalierbarkeit	(✔)	Eignung Ressourcenknappheit, Wahlmöglichkeit Methoden [May04, S. 31, 37, 39, 77-78, 130]
Verteilte Prognose			
7.1	Umgang mit heterogenen Trainingsdaten	(✔)	Messmöglichkeit zu jedem Zeitpunkt vorausgesetzt [May04, S. 33, 39, 129, 130]
7.2	Angabe von Fähigkeiten	?	vermutlich nicht erfüllt [May04, S. 88, 130]
7.3	Wahlmöglichkeiten Verteilung	(✔)	möglich: möglicherweise entferntes Messen, entfernter Anwendungszugriff [May04, S. 79, 87-88, 129]
7.4	Ansätze gegen instabile Vernetzung	?	vermutlich nicht erfüllt [May04, S. 88, 130]
7.5	Effizienzoptimierungen abonnierte Prognosen	?	unklar [May04, S. 79]
Benutzbarkeit			
8.1	Geringer Vorbereitungsaufwand für Anwendungsentwickler	✔	praktisch keine Modellierung nötig da kompletter Kontext zu Cluster zusammengefasst
8.2	Geringer Aufwand beim Einsatz für Anwendungsentwickler	?	keine Angaben gefunden
8.3	Unaufdringlichkeit für Nutzer	(✔)	adaptives Online-Lernen, jedoch Benennung von Clustern durch Nutzer [May04, S. 31, 39, 64]
8.4	Nachvollziehbarkeit für Nutzer	?	keine Angaben gefunden

Der Ansatz von Mayrhofer weist in den meisten Anforderungsgruppen sowohl Stärken als auch Schwächen auf. Dabei ist das Potential des Ansatzes noch nicht ausgeschöpft. Die Erfüllung von Anforderungen wie z.B. der Unterstützung von Zusammenhangsarten, der Angabe von Unsicherheit und der Möglichkeit der verteilten Prognose können durch Weiterentwicklung des Verfahrens und neuer Prognosemethoden noch verbessert werden.

Einige Eigenschaften des Verfahrens liegen dagegen schon in dessen Ansatz begründet. Durch die Austauschbarkeit von Methoden kann eine gute Anpassbarkeit und leichte Erweiterbarkeit erreicht werden. Der Anspruch, trotz Anpassbarkeit Anwendungsentwickler nicht zu belasten, führt auf der einen Seite zu einem sehr geringen Vorbereitungsaufwand für Anwendungsentwickler, ist aber auf der anderen Seite mit Einschränkungen der Anpassbarkeit wie z.B. der fehlenden Möglichkeit des Einbringens von a priori vorhandenem Wissen verbunden.

Das Erreichen eines so geringen Vorbereitungsaufwands wird ermöglicht durch das Clustering, also die *Vergröberung* von Low-Level-Kontext zu High-Level-Kontext. Das ganze Problem der Kontextdatenprognose wird dadurch vereinfacht, denn die der Pro-

gnose zugrunde liegenden historischen Daten enthalten nur eine Variable, die den aktuellen Cluster angibt. So kann ein geringer Ressourcenaufwand bei der Prognose erreicht werden, was zur Verbesserung der Eignung für mobile Geräte beiträgt. Die Menge benötigter Trainingsdaten wird reduziert. Es wird allerdings ein Teil der Herausforderungen der Prognose zum Clustering verlagert. Den Erleichterungen durch die Vergröberung steht außerdem entgegen, dass die Bedeutung primären Kontextes, der nur implizit durch Cluster ausgedrückt wird, nicht zur Verbesserung der Genauigkeit genutzt werden kann. Außerdem ergibt sich für den Nutzer eine Belastung durch die Aufgabe der Benennung.

Für die *Nutzung von Prognosen durch Anwendungen* leistet das Clustering den Zusatznutzen der Aggregation als eine Art vertikaler Verarbeitung. Die Beschränkung der Prognose auf High-Level-Kontext stellt jedoch für Anwendungen eine gravierende Einschränkung dar. Es ist zwar möglich, beispielsweise morgens den Kontext „Aufstehen“ zu erkennen, den Kontext „Frühstück“ zu prognostizieren und daraufhin die neuesten Nachrichten zum Lesen beim Frühstück herunterzuladen [May04, S. 16]. Die in dieser Arbeit umzusetzende Prognose der Verfügbarkeit von Diensten für das DEMAC-Projekt (vgl. Beispiel 1.1) und andere Einsatzbereiche wie z.B. die Prognose des Telefonierverhaltens eines Mobiltelefonbesitzers (vgl. Beispiel 1.2, 2.8) scheint der Ansatz von Mayrhofer jedoch nicht praktikabel ermöglichen zu können, da Cluster keinen Low-Level-Kontext wie die Verfügbarkeit eines Dienstes oder die Angabe, ob der Nutzer gerade telefoniert, enthalten.

Aufgrund der Beschränkung des Ansatzes von Mayrhofer auf High-Level-Kontext kann diese Arbeit, die sich gerade Generik zum Ziel gesetzt hat, nicht einfach auf diesem Ansatz aufbauen. Die Nutzung einer Abwandlung des Ansatzes, die auch die Prognose auf der Ebene von Low-Level-Kontext unterstützt, wäre evtl. möglich, wenn man Wissen darüber voraussetzt, dass der Wert einer bestimmten Variablen X die gesuchten Informationen darstellt. Dann könnte man das Clustering so anpassen, dass für alle Cluster gilt, dass X bei allen Kombinationen von Merkmalswerten, die jemals diesem Cluster zugeordnet werden, den gleichen Wert besitzt, so dass zu jedem Cluster eindeutig ein Wert für X festgelegt wäre. Da die Bildung von Clustern aber grundsätzlich alle Variablen einbezieht statt nur der, die überhaupt mit der zu prognostizierenden Variablen zusammenhängen, würde man sich nicht auf stabile Inseln von Zusammenhängen beschränken. Man würde potentiell irrelevante Informationen einbeziehen und diese dann zusammen mit den relevanten Informationen vergröbern und so Genauigkeit und Effizienz unnötig beeinträchtigen (vgl. auch [Sig08, S. 20-21, 53, 88, 179]).

Daher wird die Möglichkeit der Nutzung einer Abwandlung des Ansatzes von Mayrhofer für diese Arbeit nicht weiterverfolgt. Einzelne Aspekte des Verfahrens kommen jedoch für die Verwendung in dieser Arbeit in Frage. So kann Clustering z.B. nur auf bestimmte Variablen angewendet werden. Das ist z.B. nützlich bei der Prognose der Position des Nutzers, um häufige Aufenthaltsorte des Nutzers zu erkennen und beim Lernen und der Prognose verwenden zu können.

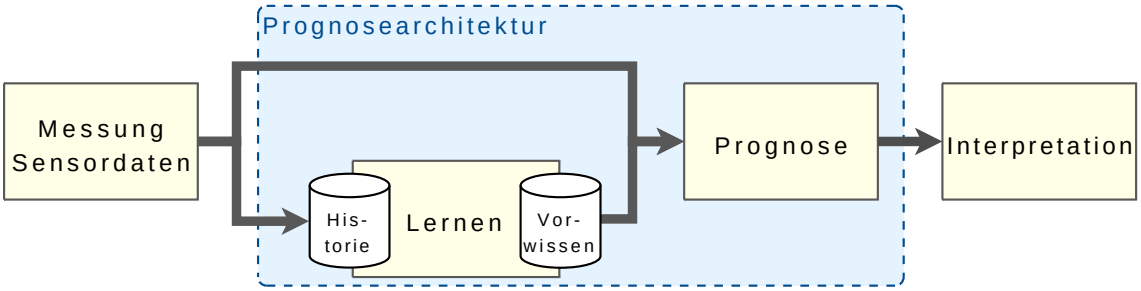

Abbildung 3.4.: Architektur von Sigg für den Fall von Prognose auf Basis von Low-Level-Kontext [Sig08, S. 93, 101, 103]

3.2.2. Verfahren von Sigg

Sigg beschäftigt sich in seiner Dissertation aus dem Jahr 2008 mit dem Vergleich von Methoden und Herangehensweisen zur Kontextdatenprognose [Sig08, S. 6, 19, 20, 44-45, 249-254]. Durch theoretische Analysen und Simulationen versucht er, in erster Linie eine Optimierung der *Prognosegenauigkeit* zu erreichen [Sig08, S. 6, 19, 20, 44-45, 249-254]. Er entwickelt eine Architektur, die Prognose für Kontextdaten auf unterschiedlichen *Abstraktionsebenen* erlaubt [Sig08, S. 6, 20, 22]. Im Rahmen dieser Architektur implementiert Sigg drei Methoden [Sig08, S. 107, 249]. Von der Implementierung eines ganzen Prognosesystems ist jedoch nicht die Rede, so dass man nur eingeschränkt von einem vollständigen Verfahren sprechen kann.

Sigg erarbeitet eine schlanke, modulare *Architektur*, die Messung und Nutzung von Kontext sowie Prognose und Interpretation als Verarbeitung von Kontext beinhaltet, wobei Prognose als horizontale Verarbeitung verstanden wird [Sig08, S. 52, 93, 95] (vgl. Abbildung 3.4). Die Prognose als Teil der Architektur wird durch eine Prognosearchitektur genauer beschrieben und andere Module der Gesamtarchitektur werden als existierend vorausgesetzt [Sig08, S. 97]. Die Prognose kann sowohl mit Low-Level-Kontext als Ein- und Ausgabe vor einer Interpretation, als auch mit High-Level-Kontext als Ein- und Ausgabe nach einer Interpretation erfolgen, wobei Sigg die erste Variante bevorzugt [Sig08, S. 89, 92, 101, 103, 104]. Die Prognosearchitektur enthält ein Lern- und ein Prognosemodul [Sig08, S. 102-103]. Das Prognosemodul bezieht die Trainingsdaten aus einer Historie, die alle Kontextdaten bis zu einem bestimmten Alter speichert [Sig08, S. 75, 95, 104-105, 237-238]. Das Prognosemodul greift zum einen auf das beim Lernen gewonnene Vorwissen zu und bezieht zum anderen die gegebenen Informationen der Prognosen aus der Historie [Sig08, S. 95, 105-107].

Durch die modulare Architektur ist die Prognosemethode austauschbar [Sig08, S. 91, 102]. Auch die *Verteilung* von Modulen auf unterschiedliche Geräte ist möglich [Sig08, S. 20, 22, 95-96, 102]. Die Einsetzbarkeit auf *mobilen Geräten* und ein entsprechender Umgang mit den Ressourcen der Geräte sieht Sigg zwar als wichtig an und berücksichtigt er beim Vergleich von Methoden, aber sein Fokus liegt auf der Prognosegenauigkeit [Sig08, S. 88, 203-205].

Es folgt wie im vorhergehenden Abschnitt 3.2.1 eine Analyse des Verfahrens im Hin-

blick auf die Erfüllung der Anforderungen des Anforderungskataloges. Da Sigg kein vollständiges Prognosesystem implementiert hat, ist das Symbol „**?**“ bei vielen Anforderungen so zu interpretieren, dass sich erst bei einer kompletten Realisierung des Verfahrens entscheidet, ob die jeweilige Anforderung erfüllt wird.

Nr.	Anforderung		Begründung / Kommentar
Kontextdaten als historische Daten			
1.1	Unterstützung von Zusammenhangsarten:		
1.1.1	Trends	✔	geeignete Methode implementiert [Sig08, S. 97, 107]
1.1.2	Sequentielle Muster	✔	unterstützt von allen implementierten Methoden [Sig08, S. 97, 107]
1.1.3	Periodische Muster	?	nirgendwo festgelegt
1.1.4	Lineare Zusammenhänge	✔	geeignete Methode implementiert [Sig08, S. 107, 227-228]
1.1.5	Nichtlineare Zusammenhänge	✔	geeignete Methoden implementiert [Sig08, S. 107, 213, 221]
1.1.6	Gemischte Zusammenhänge	(✔)	geeignete Methoden für bestimmte Kombinationen implementiert [Sig08, S. 97, 107]
1.2	Direkte Unterstützung von Skalenniveaus:		
1.2.1	Nominal	✔	geeignete Methoden implementiert [Sig08, S. 54, 231]
1.2.2	Ordinal	?	nirgendwo festgelegt
1.2.3	Metrisch	✔	geeignete Methoden implementiert [Sig08, S. 54, 231]
1.3	Unterstützung von Entitätsarten:		
1.3.1	Menschen	(✔)	keine besondere Optimierung [Sig08, S. 97, 102]
1.3.2	Technik	(✔)	keine besondere Optimierung [Sig08, S. 97, 102]
1.4	Optimierungen primärer Kontext	✘	[Sig08, S. 47]
1.5	Nutzung des Zusammenhangs prim. Kont. → sekund. Kont.	✘	[Sig08, S. 47]
1.6	Horizontale Verarbeitung auf unterschiedlichen Abstraktionsebenen:		
1.6.1	Low-Level-Kontext	✔	[Sig08, S. 6, 20, 22, 92]
1.6.2	High-Level-Kontext	✔	[Sig08, S. 6, 20, 22, 92]
1.7	Vertikale Verarbeitung	✘	als vorhanden vorausgesetzt [Sig08, S. 97]
Umgang mit Dynamik von Kontext			
2.1	Instanzen	(✔)	Instanzen von Kontextquellen unterstützt [Sig08, S. 95, 96, 98, 99]
2.2	Aktive Elemente	?	nirgendwo festgelegt
Anpassung an Nutzer			
3.1	Adaptives Online-Lernen	(✔)	eine implementierte Methode lernt gar nicht [Sig08, S. 75, 77, 82, 95-96, 104-106, 231, 237-238]
3.2	Unterstützung bei Erstellung von Prognosemodell	-	kaum Modellierung nötig
3.3	Einbringen von A Priori - Wissen	?	nirgendwo festgelegt
Integration in Anwendungsszenarien			
4.1	Plattformunabhängigkeit	?	vermutlich in Java implementiert [Sig08, S. 233]
4.2	Universelle Schnittstellen	?	nirgendwo festgelegt

Nr.	Anforderung		Begründung / Kommentar
4.3	Universelles Kontext-Metamodell	✔	[Sig08, S. 55-57]
4.4	Unterstützung von Rollen der Zeit in gesuchten Informationen:		
4.4.1	Informationen über Variablen/Ereignisse gesucht	✔	[Sig08, S. 80-81]
4.4.2	Zeitpunkt gesucht	✔	geeignete Algorithmen implementiert [Sig08, S. 96, 207, 231]
4.5	Unterstützung von Informationen über Variablen:		
4.5.1	Wahrscheinlichkeitsverteilung	✘	[Sig08, S. 80-82, 95, 109, 112, 235]
4.5.2	Modalwert	✔	[Sig08, S. 80-82, 95]
4.5.3	Wahrscheinlichkeit zu logischer Formel	✘	da keine Wahrscheinlichkeiten als Prognoseergebnis [Sig08, S. 80-82, 95]
4.5.4	Erwartungswert und Varianz	?	nirgendwo festgelegt
Genauigkeit			
5.1	Fundierte Angabe prinzipieller Unsicherheit	✘	[Sig08, S. 80-82, 95, 109, 112, 235]
5.2	Hohe Generalisierungsfähigkeit[2]	(✔)	bei einer implementierten Methode erfüllt [Sig08, S. 207, 231]
5.3	Genug Trainingsdaten lernen	(✔)	je nach Methode hoher Trainingsdatenbedarf bei Prognose auf Low-Level-Ebene [Sig08, S. 21, 206, 252]
5.4	Fundierte Angabe Schätzunsicherheit	?	nirgendwo festgelegt
5.5	Lernen stabiler Inseln von Zusammenhängen	(✔)	eine Methode implementiert, die nur Inseln lernt [Sig08, S. 105, 107, 108]
5.6	Berücksichtigung räumlich und zeitlich nahen Kontextes	(✔)	manuell möglich durch Variablenwahl
5.7	Einbeziehen zusätzl. Inform.	(✔)	manuell möglich durch Variablenwahl
5.8	Angabe Variationsunsicherheit	?	nirgendwo festgelegt
5.9	Anpassung an Änderungen von Zusammenhängen	(✔)	Trends unterstützt, ansonsten nur andauerndes Lernen [Sig08, S. 82, 88-89, 97, 104, 106]
5.10	Geringe Methodenunsicherheit	(✔)	nicht bei allen implementierten Methoden, keine Unterstützung prinzipieller Unsicherheit [Sig08, S. 80-82, 95, 109, 110, 112, 235 238]
5.11	Angabe Methodenunsicherheit	?	nirgendwo festgelegt
5.12	Umgang mit Mängeln in Trainingsdaten:		
5.12.1	Fehlerhafte Werte	?	zumindest Fehlertoleranz als wichtig angesehen [Sig08, S. 204]
5.12.2	Fehlende Werte	✔	[Sig08, S. 73-74]
5.12.3	Subjektive Einschätzungen	?	nirgendwo festgelegt
5.13	Konsistenz mehrerer Prognosen	?	methodenabhängig
Effizienz			
6.1	Abstraktion beim Lernen	(✔)	eher weniger bei Prognose auf Low-Level-Ebene, methodenabhängig, Datenhistorie begrenzbarer Länge gespeichert [Sig08, S. 95, 104-105, 110, 237-238, 252]
6.2	Prognosebedarf berücksichtigen	?	nirgendwo festgelegt

[2] Bei Sigg ist die Rede von „approximate pattern matching“.

Nr.	Anforderung		Begründung / Kommentar
6.3	Angepasste Aufwandsverteilung auf Lernen und Prognose	✔	durch Wählbarkeit von Methoden
6.4	Begrenzbarkeit Ressourcennutzung	?	unklar [Sig08, S. 205-206]
6.5	Kein unnötiges Lernen	?	unklar [Sig08, S. 106]
6.6	Stabil geringe Reaktionszeiten	(✔)	je nach Methode bei weiter in Zukunft reichenden Prognosen oder vielen Variablen stark erhöht [Sig08, S. 231]
6.7	Effizienz bei vielen Variablen	(✔)	je nach Methode sehr unterschiedlich [Sig08, S. 110, 214, 228]
6.8	Skalierbarkeit	(✔)	Speicherbedarf je nach Methode anpassbar, Rechenaufwand kann je nach Methode groß werden [Sig08, S. 110, 231]
Verteilte Prognose			
7.1	Umgang mit heterogenen Trainingsdaten	?	nirgendwo festgelegt
7.2	Angabe von Fähigkeiten	?	nirgendwo festgelegt
7.3	Wahlmöglichkeiten Verteilung	✔	[Sig08, S. 20, 22, 95-96, 102, 105]
7.4	Ansätze gegen instabile Vernetzung	?	nirgendwo festgelegt
7.5	Effizienzoptimierungen abonnierte Prognosen	?	nirgendwo festgelegt
Benutzbarkeit			
8.1	Geringer Vorbereitungsaufwand für Anwendungsentwickler	✔	da eine Methode auf komplette Daten angewendet
8.2	Geringer Aufwand beim Einsatz für Anwendungsentwickler	?	nirgendwo festgelegt
8.3	Unaufdringlichkeit für Nutzer	✔	keine explizite Nutzerinteraktion nötig
8.4	Nachvollziehbarkeit für Nutzer	?	nirgendwo festgelegt

Das Verfahren von Sigg berücksichtigt ähnlich wie das Verfahren von Mayrhofer ↑ den Umgang mit Unsicherheit und insbesondere dessen Angabe leider nicht im anzustrebenden Maß. Positiv hervorzuheben ist jedoch die relativ gute Abdeckung von Zusammenhangsarten. Die Anpassbarkeit an die Zusammenhangsarten und andere für den Anwendungsbereich spezifische Gegebenheiten werden durch die *Austauschbarkeit der Prognosemethode* erreicht.

Im Gegensatz zum Verfahren von Mayrhofer ist das Verfahren von Sigg nicht auf Prognose auf der *Abstraktionsebene* von High-Level-Kontext beschränkt. Dass die Reihenfolge der Verarbeitungsschritte nicht vorgegeben ist, stellt eine Neuerung dar [Sig08, S. 19-20]. Sigg geht zwar davon aus, dass für nicht-triviale Anwendungen nur High-Level-Kontext nützlich ist [Sig08, S. 53], hält aber aufgrund seiner Untersuchungen das Gesamtergebnis aller Verarbeitungen bei einer Prognose auf der Ebene von Low-Level-Kontext, die vor einer Interpretation stattfindet, in den meisten Fällen für genauer [Sig08, S. 89, 17]. Eine Rolle spielt dabei, dass eine Prognose im Fall des Einsatzes nach einer Interpretation erstens schon auf mehr oder weniger fehlerbehafteten Ergebnissen der Interpretation aufbaut und sich zweitens mit dem verringerten Informationsgehalt von High-

Level-Kontext begnügen muss [Sig08, S. 20-21, 53, 84, 88, 179, 201]. Im vorhergehenden Abschnitt 3.2.1 wurde zudem anhand von Beispielen (im Widerspruch zu Sigg) auch die Bedeutung von Prognose auf der Ebene von Low-Level-Kontext ohne eine darauf folgende Interpretation herausgestellt.

Unabhängig davon, ob Prognose (und Lernen) auf der Ebene von Low-Level- oder High-Level-Kontext stattfinden, stellt sie ähnlich wie bei Mayrhofer eine durch die Wahl einer Methode festgelegte *große Einheit* dar, die bei den von Sigg implementierten Methoden eine *Black Box* bilden [Sig08, S. 102-103, 206, 213-217, 221-229]. Dieser Ansatz hat weitreichende Konsequenzen. Auf der einen Seite ergibt sich ein geringer Vorbereitungsaufwand für die Anwendungsentwickler. Auf der anderen Seite sind die Möglichkeiten der Optimierung der Prognose durch ein anwendungsspezifisches Prognosemodell kaum gegeben.

Eine wichtige Konsequenz des Einsatzes allein einer einzelnen Methode für das komplette Lernen und die komplette Prognose in Kombination mit dem Einsatz auf der Ebene von Low-Level-Kontext liegt darin, dass die Methode eine potentiell hohe Anzahl an Variablen, die zusammen mehrdimensionale historische Daten bilden, bewältigen muss [Sig08, S. 206]. Im Vergleich zu Mayrhofer entfällt die Entlastung der Prognosemethode durch das Clustering ↑. Dadurch hängt die Erfüllung einer Vielzahl von Anforderungen wie der folgenden in hohem Maße von der Methode ab. Die Nutzung des hohen Informationsgehalts von Low-Level-Kontext erfordert das Lernen einer großen Menge an Trainingsdaten. Solange der hohe Bedarf an Trainingsdaten nicht gedeckt werden kann, muss dies durch eine gute Generalisierungsfähigkeit der Methode kompensiert werden. Um den Speicherbedarf in einem akzeptablen Rahmen zu halten, muss die Methode beim Lernen abstrahieren. Sigg selbst schreibt Prognose auf der Ebene von Low-Level-Kontext einen höheren Speicherbedarf als auf der Ebene von High-Level-Kontext zu [Sig08, S. 252]. Trotz des hohen Informationsgehaltes und der möglichen Vielzahl an Variablen und Zusammenhängen müssen auch der Rechenaufwand und die Reaktionszeiten von der Methode gering gehalten werden. Der Ressourcenaufwand sollte im Idealfall begrenzbar und skalierbar sein und in keinem Fall die Möglichkeiten mobiler Geräte überschreiten, die bei Sigg weniger im Vordergrund stehen. Trotz dieser schwierigen Bedingungen sollte die Prognosegenauigkeit nicht zu sehr leiden.

Die von Sigg favorisierte *Alignment-Methode* stellt den wahrscheinlich aussichtsreichsten Kandidaten der drei implementierten Methoden im Bezug auf diese Anforderungen dar. Sie verfügt über Generalisierungsfähigkeit [Sig08, S. 207, 231] und kann beim Lernen durch Beschränkung auf Inseln von Zusammenhängen abstrahieren [Sig08, S. 110]. Die Reaktionszeit und der Rechenaufwand bei einer Prognose sind jedoch bedenklich, da die Zeitkomplexität bei Sigg mit $O(k^3)$ angegeben ist, wobei k ausdrückt, wie weit die Prognose in die Zukunft reicht [Sig08, S. 231].

Da keine geeignete Methode ohne ernsthafte Probleme im Hinblick auf die Zielsetzung dieser Arbeit bekannt ist, wird das Verfahren von Sigg nicht direkt verwendet. Die Verwendung einzelner Aspekte des Verfahrens kommt jedoch in Frage.

3.2.3. Verfahren von Petzold

Petzold schlägt in seiner Dissertation von 2005 eine Klasse von Prognosemethoden und Ansätze für eine Reihe von Aspekten von Kontextdatenprognose vor und evaluiert sie [Pet05, S. 141-144]. Eine Architektur wird nicht explizit ausgearbeitet [Pet05]. Das Verfahren von Petzold ist ähnlich spezialisiert wie das von Mayrhofer und noch weniger konkret ausgearbeitet als das von Sigg [Pet05]. Daher wird auf eine Prüfung bezüglich der Erfüllung jeder einzelnen Anforderung des Anforderungskatalogs verzichtet.

Die Dissertation von Petzold geht davon aus, dass Kontextdaten von der Anwendung in einer Historie gespeichert werden und dass auf Basis dieser Daten Prognosen für die Anwendung durchgeführt werden [Pet05, S. 12]. Es wird auch die Möglichkeit des Einsatzes auf mobilen und ressourcenarmen Geräten berücksichtigt [Pet05, S. 12, 16].

Die Prognose beschränkt sich auf primären Kontext [Pet05, S. 9, 141]. Petzold setzt voraus, dass zukünftiger sekundärer Kontext bei Bedarf aus prognostiziertem primärem Kontext bestimmt wird [Pet05, S. 9]. Wie sekundärer Kontext aus primärem bestimmt werden kann, wird nicht von der Dissertation thematisiert. Stattdessen fokussiert sie sich im Rahmen von primärem Kontext noch auf die Position [Pet05, S. 8, 13, 15, 17, 21].

Kontextdatenprognose besteht bei Petzold in der Vorhersage des wahrscheinlichsten nächsten Elements einer Zeitreihe von Kontextdaten mit diskreten Werten bzw. des Wertes einer entsprechenden Variablen [Pet05, S. 2, 7, 107]. Hinzu kommen Angaben über den Zeitpunkt, zu dem die Variable den nächsten Wert annimmt, und die Zuverlässigkeit der Prognose [Pet05, S. 2, 7, 107]. Es wird adaptives Lernen unterstützt und der Bedarf an Trainingsdaten und die Generalisierungsfähigkeit sowie die Anpassung an gewisse Änderungen von Zusammenhängen werden als „Anlernen“ und „Umlernen“ bedacht [Pet05, S. 1, 9, 25-26].

Eine Verfeinerung liegt im hybriden, parallelen Einsatz von Methoden [Pet05, S. 2, 9, 25, 87]. Dies führt zu einem erhöhten Speicherbedarf, wird jedoch von Petzold erfolgreich zur Verbesserung der Genauigkeit genutzt [Pet05, S. 142]. Insgesamt können durch *Hybridität* die Vorzüge unterschiedlicher Methoden kombiniert werden [Pet05, S. 142].

Der Ansatz von Petzold eignet sich wegen der Beschränkung auf primären Kontext für diese Arbeit, die eine hohe Generik anstrebt ↑, in dieser Form nicht. Die nötige Software zur Bestimmung von sekundärem aus primärem Kontext ist nicht immer vorhanden. Hinzu kommt, dass das Konzept der Beschränkung der Prognose auf primären Kontext und der Bestimmung von sekundärem Kontext aus dem prognostizierten primären Kontext trotz Eleganz in der Theorie in manchen Fällen in der Praxis nur schwer umsetzbar ist. Die Position als primärer Kontext kann z.B. nicht von jedem mobilen Gerät gemessen werden. Da oft beliebige unbekannte Entitäten im Kontext erscheinen können ↑, ist das Sammeln von genügend Vorwissen über die Identität von Entitäten und deren Aktivitäten nicht immer möglich. In Beispiel 2.2 über den Versand von Daten mittels eines stark ausgelasteten drahtlosen lokalen Netzes am Flughafen wäre eine Prognose der Auslastung des Netzes durch Einbeziehen der Aktivitäten aller Nutzer z.B. ziemlich schwierig. Stattdessen bietet es sich an, zunächst die vielversprechende Alternative der direkten Prognose der Netzauslastung durch Analyse von Trends und periodischen Mustern im Verlauf der Netzauslastung und Zusammenhängen mit dem Flugplan zu erproben, da so

die Privatsphäre der Nutzer besser geschützt bleibt und wegen der Vermeidung der Prognose der Aktivitäten jedes einzelnen Nutzers eine höhere Effizienz zu erwarten ist. Eine auf primären Kontext beschränkte Prognose ist also nicht in allen Fällen gut geeignet.

Dennoch stellt die Möglichkeit der speziellen Berücksichtigung von primärem Kontext gegenüber den Verfahren von Mayrhofer und Sigg einen Vorteil dar. Problematisch ist die Beschränkung darauf. Wichtig wäre für den bestehenden Ansatz auch, dass die Bestimmung sekundären Kontextes vom Prognosesystem durchgeführt werden kann und als Teil der Prognose angesehen wird. Der Grundgedanke der Prognose von sekundärem Kontext durch Prognose von primärem Kontext und anschließender Bestimmung des sekundären Kontextes aus dem primären beinhaltet die interessante Idee der Aufteilung der Prognose in aufeinander folgende Schritte, die durch unterschiedliche Methoden durchgeführt werden können. Die von Petzold untersuchten Möglichkeiten der Hybridität erlauben zusätzlich eine Aufteilung auf parallel eingesetzte Methoden. Der Einsatz mehrerer Methoden bietet insgesamt die Chance, die Schwierigkeiten zu vermeiden, die sich bei dem Verfahren von Sigg daraus ergeben, dass Lernen und Prognose komplett einer einzelnen Methode auferlegt werden ↑.

3.2.4. Weiterführende Diskussion

In den Abschnitten 3.2.1 bis 3.2.3 wurden die vorgestellten Verfahren bereits diskutiert. An dieser Stelle erfolgt ausgehend davon auf einer allgemeineren Ebene eine Diskussion grundlegender Entscheidungen, die bei der Entwicklung eines Verfahrens zur Kontextdatenprognose getroffen werden müssen. Die folgenden vier Entscheidungsfelder werden in diesem Abschnitt diskutiert.

- Prognose auf der Ebene von High-Level-Kontext oder auf der Ebene von Low-Level-Kontext
- Einsatz einer Methode oder Einsatz mehrerer Methoden
- Art des Einbringens von Wissen über die Anwendungsdomäne
- Konkrete Variablenwerte oder Wahrscheinlichkeitsverteilungen zur Berücksichtigung prinzipieller Unsicherheit

Eine *Aggregation von Low-Level-Kontext zu High-Level-Kontext* vor der Prognose erleichtert die Prognose in vielen Aspekten deutlich ↑. Die Aggregation selbst bietet zudem schon einen Nutzwert ↑. Eine Beschränkung auf High-Level-Kontext allein kommt jedoch für diese Arbeit aufgrund der fehlenden Eignung für viele Anwendungsbereiche nicht in Frage ↑. Eine Prognose auf der Ebene von Low-Level-Kontext ist deutlich schwieriger ↑, bietet aber in den meisten, jedoch nicht allen Fällen eine höhere Genauigkeit ↑. Die Unterstützung beider Varianten ist wichtig für die Anpassung an das jeweilige Anwendungsszenario. Sigg geht noch weiter und schlägt für zukünftige Forschung das genauere Verfolgen der Idee vor, die Prognose für unterschiedliche Teile der Kontextdaten auf unterschiedlicher Abstraktionsebene durchzuführen [Sig08, S. 116, 251-252, 255]. Man könnte z.B. GPS-Koordinaten zu Werten wie „auf der Arbeit“ und „zu Hause“ aggregie-

ren. Die Verfolgung des Vorschlages von Sigg ist erstrebenswert, um ganz spezifisch die beste Abstraktionsebene wählen zu können.

Insbesondere im Fall von Prognose auf der Ebene von Low-Level-Kontext stellt der Ansatz des *Einsatzes einer einzigen Methode* zum Lernen und zur Prognose hohe Anforderungen an diese Methode ↑. Mayrhofer und Sigg erreichen durch die Austauschbarkeit der Methode, dass für Kontextdatenprognose verfügbare Methoden nicht allen Anwendungsbereichen gerecht werden müssen, sondern sich auf die spezifischen Anforderungsschwerpunkte bestimmter Anwendungsbereiche beschränken können. Die von Petzold aufgegriffene Hybridität, also der sequentielle und parallele Einsatz mehrerer Methoden zur Laufzeit ermöglicht außerdem, die Vorzüge unterschiedlicher Methoden beim Einsatz in einem bestimmten Anwendungsbereich zu kombinieren [Pet05, S. 142]. Durch eine Beschränkung der einzelnen Methoden auf Inseln von Zusammenhängen mit wenigen unabhängigen Variablen können die Methoden entlastet werden. Das betrifft die Erfüllung methodenabhängiger Anforderungen wie Genauigkeit, Effizienz und die Unterstützung von Zusammenhangsarten, Skalenniveaus und Arten gesuchter Informationen. Auch Mayrhofer schlägt den parallelen Einsatz mehrerer Methoden zum Zeitpunkt seiner Dissertation als zukünftige Aufgabe vor [May04, S. 65, 76]. Für diese Arbeit bietet sich der Einsatz mehrerer, austauschbarer Methoden aufgrund der angestrebten Generik ↑ ganz besonders an, denn Mayrhofer, Sigg und Petzold kommen alle zu dem Ergebnis, dass es keine universelle, bezüglich aller Anforderungen optimale Methode gibt [May04, S. 86, 91] [Sig08, S. 203, 204] [Pet05, S. 142]. Lenič et al. stellen dies auch für den Bereich des Data Mining fest [LKZ+05, S. 306-307]. Daher werden beim zu entwickelnden Verfahren mehrere, austauschbare Methoden eingesetzt werden.

Unabhängig von den Methoden ist das *Einbeziehen von Wissen über die Anwendungsdomäne* wichtig, weil es Lernen und Prognose erleichtert. Aufgrund der angestrebten Generik ↑ und Unterstützung möglichst vieler Anwendungsszenarien ist der Bedarf an Wissen über die Anwendungsdomäne, um an eine Anpassung an die spezifischen Gegebenheiten des Anwendungsszenarien erreichen zu können, relativ groß. Wenn z.B. nur zwischen bestimmten Variablen klare Zusammenhänge bestehen, die für die von der jeweiligen Anwendung benötigten Prognosen relevant sind, kann sich ein Prognosesystem mit entsprechendem Wissen über die Anwendungsdomäne beim Lernen und der Prognose auf diese Zusammenhänge beschränken und so weniger Ressourcen verbrauchen oder bei gleichem Ressourcenaufwand die Genauigkeit verbessern. Dies könnte eine wichtige Rolle spielen beim Umgang mit dem Spannungsfeld zwischen Genauigkeit und hohen Anforderungen durch Low-Level-Kontext auf der einen Seite und leistungsschwachen, mobilen Geräten auf der anderen Seite. Die in den vorhergehenden Abschnitten vorgestellten Verfahren nutzen leider kaum Wissen über die Anwendungsdomäne, was Sigg auch im Bezug auf bestehende Methoden feststellt [Sig08, S. 20]. Mayrhofer geht auf die Vorteile des Einbeziehens von Wissen über die Anwendungsdomäne ein, nutzt diese Möglichkeit aber selbst nicht, da er sich das Ziel einer hohen Unaufdringlichkeit gesetzt hat, die dadurch gestört würde [May04, S. 128, 131-132].

Für das in dieser Arbeit zu entwickelnde Verfahren besteht die Möglichkeit der Formulierung von Wissen über die Anwendungsdomäne zur Entwicklungszeit im Prog-

	bekannte Anwendungen	**unbekannte Anwendungen**
bekannte Nutzer	• Erstellung des Prognosemodells mit Wissen über Anwendungsdomäne zur Entwicklungszeit • Einbringen von a priori bekanntem, für alle Nutzer gültigem Vorwissen in Prognosemodell zur Entwicklungszeit	• Automatisches Sammeln von Wissen über Anwendungsdomäne zur Laufzeit • Einbringen von a priori bekanntem, für alle Nutzer gültigem Vorwissen in Prognosemodell zur Entwicklungszeit
unbekannte Nutzer	• Erstellung des Prognosemodells mit Wissen über Anwendungsdomäne zur Entwicklungszeit • Adaptives Online-Lernen von nutzerspezifischen Zusammenhängen zur Laufzeit	• Automatisches Sammeln von Wissen über Anwendungsdomäne zur Laufzeit • Adaptives Online-Lernen von nutzerspezifischen Zusammenhängen zur Laufzeit

Tabelle 3.3.: verschiedene Stufen von Generik (nur bekannte oder auch unbekannte Anwendungen sowie nur bekannte oder auch unbekannte Nutzer) und mögliche Umsetzungen (gelb unterlegt: diese Arbeit)

nosemodell, da die Zielsetzung voraussetzt, dass nicht mehrere, unbekannte Anwendungen auf das Prognosesystem zugreifen. Diese Voraussetzung schränkt die Generik in gewissem Maße ein. Ohne Verzicht auf Wissen über die Anwendungsdomäne wäre die Alternative zu dieser Voraussetzung eine automatische Ermittlung solches Wissens zur Laufzeit. Eine ähnliche Wahlmöglichkeit bestand in Abschnitt 2.5.4 im Bezug auf die Unterstützung von Nutzern, deren individuelle Eigenschaften zur Entwicklungszeit unbekannt sind (vgl. Tabelle 3.3). Es war zu wählen zwischen der Anpassung an Nutzer zur Entwicklungszeit (Einbringen von a priori vorhandenem Vorwissen in das Prognosemodell) und der Anpassung an Nutzer zur Laufzeit (adaptives Online-Lernen) und es wurde die Bereitstellung der Möglichkeit des adaptiven Online-Lernens gefordert, da die Berücksichtigung der individuellen Eigenschaften jedes einzelnen Nutzers zur Entwicklungszeit nicht möglich ist. Die Berücksichtigung der Eigenschaften der Anwendungsdomäne im Prognosemodell zur Entwicklungszeit ist dagegen möglich und in gewissem Rahmen akzeptabel. Die Alternative des automatischen Sammelns von Wissen über die Anwendungsdomäne zur Laufzeit ohne Beteiligung von Menschen erscheint zu gewagt für diese Arbeit, denn mit Prognose verwandte Vorgänge wie der Prozess des Data Mining werden zum Teil als interaktiv und sogar als Kunst bezeichnet [Bra07a, S. 3] [SM05, S. 13], was die Bedeutung der Beteiligung von Menschen für gute Ergebnisse hervorhebt. Die explizite Erstellung eines Prognosemodells kann einer interaktiven Abstimmung und Optimierung der Prognose durch die Anwendungsentwickler dienen. Dies ermöglicht z.B. auch, den Umfang der Prognose zu bestimmen und das Prognosesystem so im Sinne von Skalierbarkeit an bestimmte Geräte anzupassen.

Aus diesen Gründen wird in dieser Arbeit trotz Vorbereitungsaufwand für Anwendungsentwickler nicht auf die Formulierung von Wissen über die Anwendungsdomäne im Prognosemodell verzichtet. In Abhängigkeit vom Prognosemetamodell kann der Vorbereitungsaufwand jedoch mehr oder weniger stark reduziert werden durch den weiterführenden Ansatz der Unterstützung durch Werkzeuge bei der Erstellung von Prognosemodellen.

Wenig beachtet wurde in den vorgestellten Verfahren die Möglichkeit der *Beschreibung prinzipieller Unsicherheit durch Wahrscheinlichkeitsverteilungen*. Bei der Bestimmung und dem Umgang mit Wahrscheinlichkeitsverteilungen ist mit einem höheren Ressourcenaufwand zu rechnen als bei konkreten Variablenwerten, da aus einer Wahrscheinlichkeitsverteilung mit relativ geringem Aufwand auch der Modalwert bestimmt werden kann. Der Unterschied im Ressourcenaufwand ist aber geringer, als man vermuten könnte, denn bei der Angabe eines konkreten Variablenwertes besteht wegen der Möglichkeit einer hohen prinzipiellen Unsicherheit Bedarf nach Genauigkeitsangaben, die prinzipielle Unsicherheit ausdrücken. Der Ressourcenaufwand für gute Genauigkeitsangaben ist im allgemeinen Fall nicht unerheblich, da verschiedene Kontexte unterschieden werden müssen. Wenn der Nutzer z.B. gerade auf dem Weg zur Arbeit ist, besteht bei der Prognose der Position des Nutzers für die nächsten fünf Minuten eine viel geringere Unsicherheit als z.B. bei einem spontanen Spaziergang ohne festes Ziel. Wahrscheinlichkeitsverteilungen bieten gegenüber einem konkreten Variablenwert bzw. dem Modalwert außerdem zusätzliche Informationen, so dass eine Anwendung in manchen Fällen trotz hoher prinzipieller Unsicherheit, also einer relativ gleichmäßigen Verteilung der Wahrscheinlichkeit auf verschiedene Variablenwerte sinnvolle Entscheidungen treffen kann ↑. Der interne Umgang mit Wahrscheinlichkeitsverteilungen ermöglicht jedoch auch die Angabe von Prognoseergebnissen als Modalwert, Erwartungswert und Varianz, da diese Werte in der Wahrscheinlichkeitsverteilung inbegriffen sind. Insgesamt trägt die Verwendung von Wahrscheinlichkeitsverteilungen zur Genauigkeit bei, die als wichtiger Faktor für den Einsatz von Kontextdatenprognose anzusehen ist [Sig08, S. 249-250].

3.3. Methoden

In vorhergehenden Abschnitt 3.2.4 wurde die Entscheidung getroffen, im Rahmen des zu entwickelnden Verfahrens mehrere, austauschbare Methoden einzusetzen. Daher wird eine Suche nach geeigneten Methoden erforderlich. Wegen der Austauschbarkeit der Methoden können die ausgewählten Methoden als eine Art Referenzkonfiguration angesehen werden, die in möglichen weiterführenden Arbeiten oder durch die Anwendungsentwickler erweitert werden kann. In dieser Arbeit wird angestrebt, zunächst zwei bis drei Methoden zu wählen, deren Implementierung im Zeitrahmen dieser Arbeit möglich ist und die sich so ergänzen, dass die wesentlichen Anforderungen erfüllt werden und der Einsatz in wichtigen Anwendungsbereichen ermöglicht wird. Trotz der Auswahl von nur zwei bis drei Methoden wird ein breites Spektrum an Methoden betrachtet, um erstens eine gute Auswahl treffen und zweitens den eigenen Ansatz kompatibel zu möglichst vielen Methoden gestalten zu können.

Dieser Abschnitt gibt einen Überblick über die in Frage kommenden Methoden. Dabei schließt er gleich Methoden mit gravierenden Nachteilen oder der ausschließlichen Einsetzbarkeit für weiterführende Funktionalitäten aus. Aufgrund der Vielzahl an Methoden wird kein detaillierter Vergleich durchgeführt, sondern anhand einfacher Kriterien eine Auswahl für die Referenzkonfiguration getroffen, deren Güte daraufhin anhand der Erfüllung der Anforderungen genauer geprüft wird. Nicht in die Referenzkonfiguration

aufgenommene Methoden können von Anwendungsentwicklern implementiert werden. Bei der Suche nach geeigneten Methoden wird zum einen auf die Dissertationen von Mayrhofer und Sigg zurückgegriffen [May04] [Sig08], die eine Reihe an Methoden unter ähnlichen Gesichtspunkten wie diese Arbeit betrachten. Zum anderen wird Literatur über die vorgestellten relevanten Wissenschaftszweige verwendet, in der in stärkerem Maße auch Informationen über Methoden der Folgerungsprognose verfügbar sind. Folgerungsprognose wird durch den hybriden Einsatz mehrerer Methoden attraktiv, da man Zukunftsprognose auf bestimmte Variablen (z.B. primären Kontext) beschränken und durch Folgerungsprognose die Werte anderer Variablen bestimmen kann (vgl. Abschnitt 3.2.3).

3.3.1. Sichtung und Auswahl

Methoden bilden Informationen über unabhängige Variablen (z.B. X und Y) auf Informationen ab, die sich auf eine abhängige Variable (z.B. Z) beziehen. Im Fall von Zukunftsprognose existieren unterschiedliche Variablen (z.B. X_i) für unterschiedliche Zeitpunkte (vgl. Abschnitt 2.4.8). Es wird unterschieden zwischen dem Lernen von Variablenwerten in den Datensätzen der Trainingsdaten und der Prognose.

Abbildung 3.5 gibt einen Überblick über in Frage kommende Methoden. Es wird dort unterschieden zwischen Methoden der Zukunftsprognose und Methoden der Folgerungsprognose, zwischen denen teilweise jedoch auch Korrespondenzen bestehen. Viele Methoden der Zeitreihenanalyse (rechts unten) bauen z.B. auf *Regression* auf (rechts oben) [And71, S. 8].

Data Mining, Maschinelles Lernen und Mustererkennung überschneiden sich recht stark in ihren Methoden. Am stärksten losgelöst ist die Systemdynamik. Auch die Methoden der Zeitreihenanalyse sind der verwendeten Literatur nach relativ klar von anderen Wissenschaftszweigen abgegrenzt. Zu Stochastik und Statistik wurden in der verwendeten Literatur nicht viele konkrete Methoden identifiziert. Sie bieten jedoch das mathematische Werkzeug zur Konzipierung und Anpassung von Methoden.

Ein grundlegende statistische Methode besteht darin, Zusammenhänge zwischen Variablen durch *Wahrscheinlichkeitsverteilungen* zu beschreiben (links oben). Dazu muss eine Verteilung aus einer Verteilungsfamilie gewählt werden. [Dav03, S. 166]

Eine ganze Klasse statistischer Methoden stellt *Regression* dar (rechts oben), die in Abschnitt 3.1.1 schon als Art von Methoden eingeführt wurde. Nach Bishop zielt Regression darauf ab, den Wert einer abhängigen Variablen mit metrischem Skalenniveau zu bestimmen [Bis06, S. 137]. Lineare Regressionsmethoden ermitteln den Wert der abhängigen Variablen als Funktion der unabhängigen Variablen [Bis06, S. 137-138]. Die Funktion ergibt sich als Linearkombination von Basisfunktionen ϕ_i, die auf die Werte der unabhängigen Variablen angewendet werden, z.B. $Z(X,Y) = \sum_{i=0}^{n-1} w_i \phi_i(X,Y)$ [Bis06, S. 137-138]. Die Linearität liegt in der Abhängigkeit von den Parametern w_i [Bis06, S. 137-138].

Im Gegensatz zu Regressionsmethoden werden Nearest Neighbour Classifier, die Histogramm-Regel, Entscheidungsbäume und Diskriminanten-Funktionen zur Klassifizierung eingesetzt [Bra07a, S. 31] [DGL97, S. 94-96] [Bra07a, S. 41] [Bis06, S. 42-43].

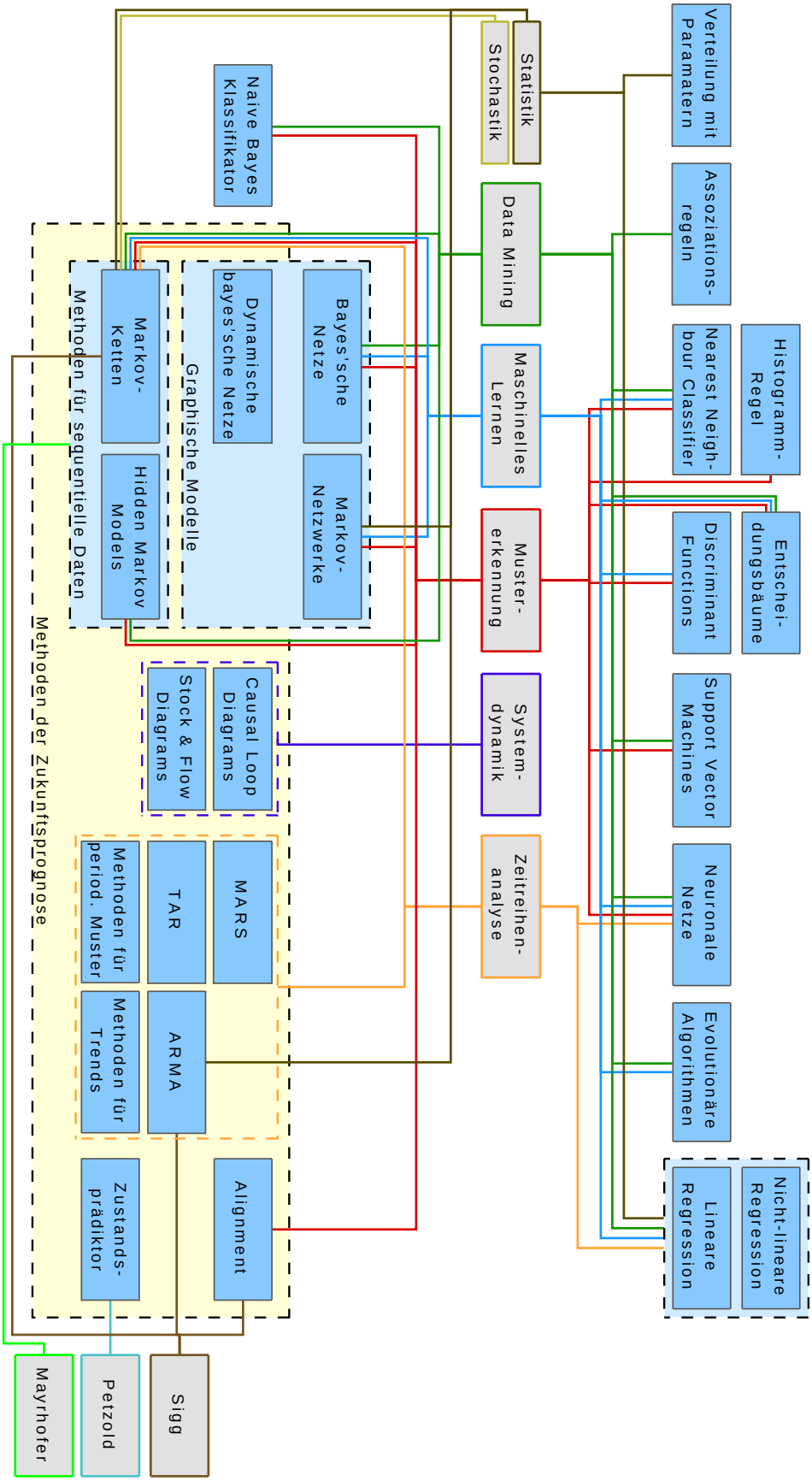

Abbildung 3.5.: möglicherweise für das zu entwickelnde Prognosesystem einsetzbare Methoden, ihre Nutzung bei den Verfahren von Mayrhofer, Sigg und Petzold sowie Beziehungen der Methoden zu den relevanten Wissenschaftszweigen entsprechend den Inhalten zugehöriger Literatur [SM05] [Lar06] [Bra07a] [LOL+05] [Rud08] [MBK98a] [Bis06] [TK06] [DGL97] [Hüb03] [Dav03] [ABH85] [And71] [Kri99] [KS97] [Kir98]

Bei der *Histogramm-Regel* und dem *Nearest Neighbour Classifier* wird von einem Datenwürfel ausgegangen, dessen Datenpunkte Wertekombinationen der unabhängigen Variablen darstellen [DGL97, S. 94-96]. Bei der Prognose wird direkt auf Datensätze der Trainingsdaten zurückgegriffen, deren Wertekombinationen der unabhängigen Variablen in der Nähe der Wertekombination liegt, die bei der Prognose für die unabhängigen Variablen gegeben ist [DGL97, S. 61] [DGL97, S. 94-96]. Die Histogramm-Regel und der Nearest Neighbour Classifier eignen sich zur Verwendung in dieser Arbeit nicht, da wegen des direkten Zugriffs auf die Trainingsdaten kein abstrahierendes Lernen stattfinden kann.

Bei der *Entscheidungsbaum*-Methode werden Zusammenhänge auf sogenannte Entscheidungsbäume als Vorwissen abgebildet. Entscheidungsbäume stehen in Zusammenhang mit regelbasierten Systemen [MBK98a, S. 18] [Bra07a, S. 41]. Prognose mit Entscheidungsbäumen geschieht durch Prüfung von Bedingungen bezüglich der unabhängigen Variablen [Bra07a, S. 43-44]. Die Bedingungen sind über den Baum verteilt und in Abhängigkeit von den Werten der unabhängigen Variablen wird ein bestimmter Pfad von der Wurzel zu einem Blatt durchlaufen [Bra07a, S. 43-44]. Die Blätter sind jeweils mit einem Wert der abhängigen Variablen versehen [Bra07a, S. 43-44].

Diskriminanten-Funktionen sind Funktionen, die die Werte der unabhängigen Variablen auf einen Wert der abhängigen Variablen abbilden [Bra07a, S. 43]. Es wird zwischen linearen und nicht-linearen Funktionen unterschieden [Bra07a, S. 181]. Im Gegensatz zu Regressions-Methoden handelt es sich um eine Methode zur Klassifizierung, so dass die abhängige Variable nominales Skalenniveau aufweist. Man kann sich vorstellen, dass der Datenwürfel zu den unabhängigen Variablen in Teile unterteilt wird, die den Werten der abhängigen Variablen entsprechen.

Support Vector Machines unterteilen wie Diskriminanten-Funktionen den Datenwürfel. Support Vector Machines erreichen durch Transformation der Daten und Erhöhung der Dimension die Unterteilbarkeit der Daten durch lineare Hyperebenen, die sonst nicht immer möglich ist [May04, S. 70] [Sig08, S. 211]. Es ist sowohl der Einsatz zur Klassifizierung als auch zur Regression möglich. Mayrhofer bezeichnet Support Vector Machines als viel versprechend, kommt jedoch auch auf den „extremen Bedarf an Rechenleistung“ aktueller Lernalgorithmen zu sprechen [May04, S. 70, 76]. Daher wird von Support Vector Machines in dieser Arbeit Abstand genommen.

Neuronale Netze können für verschiedenste Zwecke eingesetzt werden. Sie bestehen aus mehreren Neuronen, die jeweils lineare gewichtete Summen berechnen [May04, S. 69]. Online-Lernen ist jedoch mit neuronalen Netzen kaum möglich [May04, S. 69] [CR99, S. 115]. Daher werden neuronale Netze in dieser Arbeit nicht verwendet.

Ähnlich universell wie neuronale Netze sind *evolutionäre Algorithmen*. Sie nutzen einen Ansatz zur Lösung von Optimierungsproblemen, der an der Vererbung von Lebewesen orientiert ist [SM05, S. 35]. Es findet eine Suche nach einer möglichst optimalen Lösung statt [SM05, S. 35], die beim Lernen möglichst gutem Vorwissen entspricht (vgl. auch Abschnitt 3.1.2). Nach Sigg verursachen evolutionäre Algorithmen jedoch einen zu hohen Rechenaufwand für mobile, ubiquitäre Szenarien [Sig08, S. 232] (vgl. auch [LKZ+05, S. 307]). Daher werden sie nicht weiterverfolgt.

Assoziationsregeln drücken Zusammenhänge zwischen Variablen als Vorwissen aus [Bra07a, S. 187]. Sie werden durch unüberwachtes Lernen ermittelt, also ohne Angabe der Variablen, zwischen denen Zusammenhänge zu lernen sind [Bra07a, S. 187]. Dies ist allerdings nicht im Sinne des Ansatzes, Wissen über die Anwendungsdomäne wie z.B. über die Variablen, zwischen denen Zusammenhänge bestehen, einzubeziehen ↑, weshalb diese Methoden nicht genutzt wird.

Die Methoden in der oberen Hälfte von Abbildung 3.5 zielen überwiegend auf die Bestimmung des Wertes der abhängigen Variablen ab. *Bayes'sche Netze*, *dynamische bayes'sche Netze*, *Markov-Netzwerke*, *Markov-Ketten* und *Hidden Markov Models* (links unten) werden dagegen zur Bestimmung von Wahrscheinlichkeiten für das Eintreten von Werten der abhängigen Variablen verwendet und können so prinzipielle Unsicherheit ausdrücken (z.B. $P(Z{=}z \mid X{=}x, Y{=}y)$). Diese Methoden stehen in engem Zusammenhang miteinander.

Recht allgemeine Methoden stellen *Markov-Netzwerke* (auch *Markov Random Fields* genannt) und *Bayes'sche Netze* dar. Sie können eingesetzt werden zur Darstellung der Wahrscheinlichkeitsverteilung mehrerer Variablen als Vorwissen (z.B. $P(X{=}x, Y{=}y, Z{=}z)$) [Bis06, S. 359-360]. Dabei wird berücksichtigt, zwischen welchen Variablen direkte Zusammenhänge bestehen und zwischen welchen nicht [Bis06, S. 359-360]. Variablen werden als Knoten in einem Graphen repräsentiert und Zusammenhänge als Kanten [Bis06, S. 360]. Daher spricht man auch von *graphischen Modellen*. Bei bayes'schen Netzen werden gerichtete Graphen verwendet und bei Markov-Netzwerken ungerichtete [Bis06, S. 360]. *Dynamische bayes'sche Netze* sind ein Spezialfall von bayes'schen Netzen und beziehen die Zeit explizit ein [Wit02, S. 38]. Daher eignen sie sich speziell zur Zukunftsprognose.

Markov-Ketten eignen sich zur Zukunftsprognose für eine einzelne Variable und weisen im Gegensatz zu dynamischen bayes'schen Netzen in ihrer Darstellung als graphisches Modell eine feste Struktur, also feste Annahmen über Zusammenhänge auf [Bis06, S. 605-608]. Es wird für eine Markov-Kette k-ter Ordnung angenommen, dass die Wahrscheinlichkeitsverteilung einer Variablen für einen bestimmten Zeitpunkt (X_j) sich durch zusätzliche Kenntnis der Werte x_1, ..., x_{j-k-1} nicht verändert, wenn die k jüngsten Werte x_{j-k}, ..., x_{j-1} einbezogen sind [Bis06, S. 607-609]. X_j ist also nach dieser Annahme nur indirekt von älteren Werten abhängig.

Die mächtigere Methode der *Hidden Markov Models* ist geeignet für Fälle, in denen diese Annahme zu restriktiv ist [Bis06, S. 609]. Sie führen zusätzliche sogenannte *verborgene Variablen* Y_j ein, deren Werte nicht gemessen werden können [Bis06, S. 609]. Der Wert von X_j hängt jeweils vom Wert von Y_j ab und die Y_j bilden eine Markov-Kette [Bis06, S. 609]. Die Nutzung verborgener Variablen hat jedoch Konsequenzen. Es ist ein entsprechender Lernalgorithmus nötig. Nach Mayrhofer existiert kein Lernalgorithmus für adaptives Online-Lernen [May04, S. 73], so dass Hidden Markov Models für diese Arbeit nicht in Frage kommen.

Der *Naive Bayes Klassifikator* lässt sich wie Markovketten als graphisches Modell mit fester Struktur darstellen [Bis06, S. 380]. Es werden ausschließlich die unabhängigen Variablen, die abhängige Variable und direkte Zusammenhänge zwischen der abhängigen Variablen und jeder einzelnen unabhängigen Variablen dargestellt [Bis06, S. 380]. Dabei

liegt die Annahme zugrunde, dass die unabhängigen Variablen nicht voneinander abhängen, wenn der Wert der abhängigen Variablen bekannt ist [Bis06, S. 46]. Trotz dieser restriktiven Annahme führt der Naive Bayes Klassifikator in der Praxis zu guten Ergebnissen [Zha04, S. 562]. Allerdings ermittelt er nur einen Wert für die abhängige Variable und keine Wahrscheinlichkeitsverteilung [Zha04, S. 562] [Bra07a, S. 30].

Graphische Modelle berücksichtigen die Struktur von Zusammenhängen und kommen somit eher zur Unterstützung des hybriden Einsatzes von Methoden statt als Methode für den Umgang mit Inseln von Zusammenhängen in Frage (vgl. Abschnitt 3.4). Daher werden hier nur graphische Modelle mit einfacher Struktur betrachtet, bei denen direkte Zusammenhänge zwischen der abhängigen Variablen mit den unabhängigen Variablen angenommen werden. Bayes'sche Netze, dynamische bayes'sche Netze, Markov-Netze, Markov-Ketten und der Naive Bayes Klassifikator ähneln einander unter dieser Voraussetzung so sehr, dass sie relativ leicht durch eine einzige Implementierung unterstützt werden können. Daher werden sie im Folgenden zusammengefasst als eine Methode behandelt. Es erfolgt außerdem eine Einschränkung der trotzdem noch starken Allgemeinheit dieser Methode durch Beschränkung auf den in der Literatur häufig angenommenen Standardfall von diskreten Variablen bei direkter Unterstützung von nominalem Skalenniveau und der Speicherung von Wahrscheinlichkeiten in Tabellen zur Beschreibung der Zusammenhänge (vgl. z.B. [Wit02, S. 16], [Jen01, S. 19], [Hüb03, S. 124], [Bra07a, S. 31] und [May04, S. 74]). Dadurch wird die Methode so stark konkretisiert, dass sie sich überhaupt bewerten lässt, und es werden Überschneidungen mit anderen Methoden vermieden. Die Speicherung von Wahrscheinlichkeitsverteilungen macht deswegen Sinn, weil sie im Vergleich zu parametrischen Methoden wie parametrisierten Wahrscheinlichkeitsverteilungen die Unterstützung von unterschiedlichen Zusammenhängen und die Anpassbarkeit an den Nutzer verbessert. Die so festgelegte Methode wird im Folgenden „*Wahrscheinlichkeitstabellen*“ genannt.

Als nächste Methoden folgen in Abbildung 3.5 in der Mitte unten die Methoden der Systemdynamik, die sich zur Darstellung der Struktur von Zusammenhängen eignen [Kir98, S. 18]. Verbreitet sind als stark verwandte Darstellungsmittel *Causal Loop Diagrams* und für genauere, quantitative Darstellungen *Stock and Flow Diagrams* [Kir98, S. 5-9, 16] [Lan00]. Causal Loop Diagrams werden in dieser Arbeit wegen des Fehlens einer quantitativen Darstellung nicht verwendet für eine konkrete Realisierung. Eine spezielle Eigenschaft von Darstellungsweisen der Systemdynamik liegt in der expliziten (graphischen) Ausdrückbarkeit von Rückkopplungen [Kir98, S. 5,7]. Es können also nicht nur Zusammenhänge zwischen unterschiedlichen Variablen bestehen, sondern der Wert einer Variablen kann den Wert derselben Variablen zu einem späteren Zeitpunkt indirekt über andere Variablen beeinflussen. Mathematisch führen Diagramme wie Stock and Flow Diagrams zu Differentialgleichungen [Kir98, S. 24]. Nach der Darstellung in der Literatur scheint Lernen nicht vorgesehen zu sein [Kir98, S. 43-52] [Doe96, S. 202]. Methoden der Systemdynamik eignen sich deshalb und wegen der graphischen Darstellbarkeit der Struktur von Zusammenhängen wie graphische Modelle in ihrer uneingeschränkten Form eher zur Unterstützung von Hybridität.

Die *ARMA*-Methode (*Autoregressive Moving Average*) spielt eine wichtige Rolle in der

Zeitreihenanalyse und wird in mehreren Varianten eingesetzt [Kri99, S. 6] [KS97, S. 203] [ABH85]. Sie arbeitet mit linearen Mitteln und kann zur Prognose von Daten mit metrischem Skalenniveau eingesetzt werden [May04, S. 68] [Sig08, S. 231]. Für die Zwecke dieser Arbeit eignet sie sich jedoch nicht, da sie kein adaptives Online-Lernen unterstützt [May04, S. 68] [Sig08, S. 231].

Mittlerweile hat die Forschung auch nicht-lineare Methoden zur Zeitreihenanalyse hervorgebracht, die sich jedoch überwiegend auf deterministische Fälle beschränken [Kri99, S. 6] [KS97, S. 204]. Die *TAR*-Methode (*Threshold Autoregressive*) verbindet mehrere Autoregressive-Elemente [KS97, S. 206]. Auch die *MARS*-Methode baut auf Regression auf [Fri91, S. 1].

Methoden wie ARMA erfordern, dass im Fall einer Überlagerung mehrerer Zusammenhangsarten der Trend und periodische Muster vor Anwendung der ARMA-Methode entfernt werden. Daher gibt es spezielle *Methoden für Trends und periodische Muster*. Eine gängige Methode zum Ermitteln von Trends besteht im *Glätten* des Wertverlaufs einer Variablen [May04, S. 68] [And71, S. 46]. Das adaptive Online-Lernen von periodischen Mustern gestaltet sich nach Mayrhofer jedoch schwierig [May04, S. 68, 75, 76], weshalb entsprechende Methoden in dieser Arbeit nicht betrachtet werden. Das Lernen periodischer Muster mit schon zur Entwicklungszeit bekannter Periode ist außerdem auch ohne spezielle Methoden für periodische Muster möglich, indem unabhängige Variablen wie „Tageszeit“, „Wochentag“, „Monat“ usw. eingeführt werden [May04, S. 72-73].

Die *Alignment*-Methode und *Zustandsprädiktoren* stellen Methoden dar, die speziell von Sigg und Petzold zur Zukunftsprognose entwickelt bzw. aufgegriffen und angepasst wurden. Die Alignment-Methode führt bei der Prognose eine Form von Pattern Matching durch. Sie vergleicht die Folge der Werte einer Variablen der jüngeren Vergangenheit mit als Vorwissen gespeicherten typischen Wertfolgen, um bei einer Übereinstimmung die Fortsetzung der entsprechenden typischen Wertfolge zur Bildung des Ergebnisses verwenden zu können [Sig08, S. 107-108]. Durch approximatives Matching [Sig08, S. 221] führen auch Varianten von typischen Wertfolgen zu einem Ergebnis, ohne dass alle Varianten gespeichert werden müssen.

Zustandsprädiktoren verwenden auch den Wertverlauf einer Variablen in der jüngeren Vergangenheit und geben in Abhängigkeit davon den nächsten Wert an [Pet05, S. 37-49]. Sie speichern ähnlich wie Markovketten zu jedem Wertverlauf den in der Vergangenheit beim Lernen beobachteten nächsten Wert [Pet05, S. 37-49]. Anders als bei Markovketten speichern sie jedoch keine Wahrscheinlichkeitsverteilung und ändern den gespeicherten Wert beim Lernen recht schnell, wenn die Trainingsdaten für den Wertverlauf einen anderen nächsten Wert als zuvor enthalten („Umlernen“) [Pet05, S. 37-49]. Dadurch berücksichtigen Zustandsprädiktoren Variationsunsicherheit.

Es sind die folgenden 23 Methoden in diesem Abschnitt vorgestellt worden, von denen noch 13 zur Verwendung für das zu entwickelnde Verfahren in Frage kommen.

- **Nicht oder nur für weiterführende Zwecke einsetzbar**: Histogramm-Regel, Nearest Neighbour Classifier, Support Vector Machines, Neuronale Netze, Evolutionäre Algorithmen, Assoziationsregeln, Hidden Markov Models, Causal Loop Diagrams, ARMA

- **Umsetzung vermutlich problematisch**: Methoden für periodische Muster
- **Noch in Frage kommend**: Lineare Regression, Nicht-lineare Regression, Entscheidungsbäume, Discriminant Functions, Verteilung mit Parametern, Wahrscheinlichkeitstabellen, Stock and Flow Diagrams, TAR, MARS, Methoden für periodische Muster, Methoden für Trends, Alignment, Zustandsprädiktoren

Von den 13 noch in Frage kommenden Methoden bieten sich Wahrscheinlichkeitstabellen an, weil sie sich sowohl für Folgerungs- als auch für Zukunftsprognose eignen. Hinzu kommt, dass Mayrhofer und Sigg Wahrscheinlichkeitstabellen bereits zur Zukunftsprognose erprobt haben [May04, S. 71-74] [Sig08, S. 111] und dadurch erwiesen ist, dass adaptives Online-Lernen möglich ist. Da Wahrscheinlichkeitstabellen durch die Unterstützung von nominalem Skalenniveau Klassifizierung ermöglichen, ist noch der Bereich der Regression offen. Da keine Erfahrungen über Regression für Kontextdatenprognose bekannt sind, soll hier zunächst eine einfache Variante linearer Regression mit simplen Basisfunktionen verwendet werden. Die Auswahl an Methoden für die Referenzkonfiguration des zu entwickelnden Verfahrens umfasst also Wahrscheinlichkeitstabellen und eine einfache Variante linearer Regression.

3.3.2. Wahrscheinlichkeitstabellen

Vor einer genaueren Prüfung der Eignung von Wahrscheinlichkeitstabellen zeigt dieser Abschnitt zur genaueren Vorstellung der Methode, wie diese zur Kontextdatenprognose eingesetzt werden kann.

Wahrscheinlichkeitstabellen dienen der Repräsentation von Wahrscheinlichkeiten $P(V_1 = v_1, ..., V_i = v_i)$ für Kombinationen von Variablenwerten $v_1, ..., v_i$ bzw. Häufigkeiten zur Schätzung der Wahrscheinlichkeiten, was dem frequentistischen Ansatz in der Statistik folgt. Die geschätzten Wahrscheinlichkeiten sind nicht direkt in der Tabelle gespeichert, können aber daraus errechnet werden. Seien $V_1, ..., V_{i-1}$ die unabhängigen Variablen und V_i die abhängige Variable. Eine Wahrscheinlichkeitstabelle könnte z.B. so aufgebaut sein, dass die Zeilen die Kombination der Werte der unabhängigen Variablen $(v_1, ..., v_{i-1})$ und die Spalten den Wert der abhängigen Variablen (v_i) angeben. Konkret könnten die Werte $\#(v_1, ..., v_i)$ in einer solchen Tabelle angeben, in wie vielen Datensätzen der Trainingsdaten die Wertekombination $v_1, ..., v_i$ der Variablen aufgetreten ist.

Beispiel 3.1:
Wahrscheinlichkeitstabellen können z.B. verwendet werden, um zu prognostizieren, ob der Nutzer eines Mobiltelefons telefoniert (vgl. Beispiel 1.2 und 2.8). Tabelle 3.4 zeigt eine entsprechende Wahrscheinlichkeitstabelle. V_1 und V_2 sind in diesem Fall die unabhängigen Variablen und V_3 ist die abhängige Variable. Es handelt sich um Folgerungsprognose und nicht um Zukunftsprognose, da nur ein Zeitpunkt einbezogen wird.

	$V_3 = false$	$V_3 = true$
V_1=**Arbeit,** $V_2 = 0$	138	4
V_1=**Arbeit,** $V_2 = 1$	27	2
V_1=**Arbeit,** $V_2 = 2$	12	3
V_1=**zu Hause,** $V_2 = 0$	97	8
V_1=**zu Hause,** $V_2 = 1$	8	6
V_1=**zu Hause,** $V_2 = 2$	2	4

Tabelle 3.4.: fiktives Beispiel für den Einsatz einer Wahrscheinlichkeitstabelle (V_1: Position, V_2: Anzahl offener verpasster Anrufe, V_3: gibt an ob Nutzer telefonierend)

Die Häufigkeit bzw. die geschätzte Wahrscheinlichkeit $P(V_1 = v_1, ..., V_i = v_i)$ für eine Wertekombination $v_1, ..., v_i$ erhält man, indem man den Wert $\#(v_1, ..., v_i)$ in der Tabelle durch die Summe aller Werte in der Tabelle teilt (vgl. Abschnitt 2.4.9).

Für die Prognose mit Wahrscheinlichkeitstabellen muss $P(V_i = v_i \mid V_1 = v_1, ..., V_{i-1} = v_{i-1})$ ermittelt werden (vgl. Abschnitt 2.4.8). Diese bedingte Wahrscheinlichkeit lässt sich folgendermaßen berechnen.

$$P(V_i = v_i \mid V_1 = v_1, ..., V_{i-1} = v_{i-1}) \overset{\text{(Def. 2.12)}}{=} \frac{P(V_1 = v_1, ..., V_i = v_i)}{P(V_1 = v_1, ..., V_{i-1} = v_{i-1})}$$

$$\overset{\text{(Def. 2.10)}}{=} \frac{P(V_1 = v_1, ..., V_i = v_i)}{\sum_{v_i \in D(V_i)\backslash ?} P(V_1 = v_1, ..., V_i = v_i)} \overset{\text{(Gesetz großer Zahlen)}}{\approx} \frac{\#(v_1, ..., v_i)}{\sum_{v_i \in D(V_i)\backslash ?} \#(v_1, ..., v_i)}$$

$\#(v_1, ..., v_i)$ kann aus der Wahrscheinlichkeitstabelle abgelesen werden. Die Summe ergibt sich durch Aufsummieren aller Werte in der entsprechenden Zeile. Die Summen der Werte der einzelnen Zeilen können zur Verringerung des durch die Prognose verursachten Berechnungsaufwands als redundante Informationen gespeichert werden. Das Lernen eines Datensatzes der Trainingsdaten besteht dann im Inkrementieren des entsprechenden Wertes in der Tabelle und der gespeicherten Summe der Zeile.

Bei Realisierung eines Naive Bayes Klassifikators als Wahrscheinlichkeitstabellen wird die Unabhängigkeitsannahme $P(V_1 = v_1, ..., V_{i-1} = v_{i-1} \mid V_i = v_i) = P(V_1 = v_1 \mid V_i = v_i) \cdot ... \cdot P(V_{i-1} = v_{i-1} \mid V_i = v_i)$ gemacht [Zha04, S. 562], die der stochastischen Unabhängigkeit ähnelt. Wichtige Grundlage sind nach der Darstellung von Zhang die folgenden Gleichungen [Zha04, S. 562].

$$P(V_i = v_i \mid V_1 = v_1, ..., V_{i-1} = v_{i-1})$$

$$\overset{\text{(Def. 2.12)}}{=} \frac{P(V_1 = v_1, ..., V_{i-1} = v_{i-1} \mid V_i = v_i) \cdot P(V_i = v_i)}{P(V_1 = v_1, ..., V_{i-1} = v_{i-1})}$$

$$\overset{\text{(Unabhängigkeit)}}{=} \frac{P(V_1 = v_1 \mid V_i = v_i) \cdot \ ... \ \cdot P(V_{i-1} = v_{i-1} | V_i = v_i) \cdot P(V_i = v_i)}{P(V_1 = v_1, ..., V_{i-1} = v_{i-1})}$$

Da zur Klassifizierung nur der wahrscheinlichste Wert für V_i ermittelt werden muss, genügt es, durch probeweises Einsetzen den Wert $v_i \in D(V_i)\backslash ?$ zu finden, der den unteren Ausdruck maximiert, und dabei den Nenner wegzulassen, da er unabhängig von v_i ist [Bra07a, S. 30]. Die Wahrscheinlichkeiten $P(V_1 = v_1 | V_i = v_i)$, ..., $P(V_{i-1} = v_{i-1} | V_i = v_i)$

und $P(V_i = v_i)$ können jeweils einer entsprechenden Wahrscheinlichkeitstabelle entnommen werden. Sei $s = max(|D(V_1)\backslash?|, ..., |D(V_i)\backslash?|)$ die Kardinalität der größten Wertemenge aller Variablen. Dann verursacht der Naive Bayes Klassifikator durch die Unabhängigkeitsannahme und die Beschränkung auf die Bestimmung des wahrscheinlichsten Wertes einen Speicherbedarf von $O(s^2)$ im Vergleich zur oben dargestellten direkten Verwendung einer einzelnen Wahrscheinlichkeitstabelle zur Prognose mit einem Speicherbedarf von $O(s^i)$.

Im Fall der Realisierung einer Markovkette k-ter Ordnung als Wahrscheinlichkeitstabelle zur Zukunftsprognose können $X_{j-k}, ..., X_{j-1}$ als unabhängige Variablen und X_j als abhängige Variable für die direkte Anwendung einer einzelnen Wahrscheinlichkeitstabelle angesehen werden. Es wird angenommen, dass sich die Wahrscheinlichkeiten nicht mit der Zeit ändern (*homogene Markovkette*). Wie gut diese Voraussetzung erfüllt ist, spiegelt sich in der Stärke der Variationsunsicherheit wider. Weiter in die Zukunft reichende Prognosen können durch das Iterieren der Prognose [Sig08, S. 214] oder in Anlehnung an Mayrhofer durch die Betrachtung des Tupels (X_j, ..., X_{j+h}) als Wert der abhängigen Variablen [May04, S. 75] erreicht werden.

3.3.3. Lineare Regression

Als weitere Methode wurde lineare Regression gewählt, die zur Folgerungsprognose eingesetzt werden kann. Bei linearer Regression mit den üblichen einfachen Basisfunktionen und $V_1, ..., V_{i-1}$ als unabhängige Variablen wird der Wert der abhängigen Variablen V_i bei der Prognose durch Ausrechnen des Wertes von $w_0 + w_1V_1 + ... + w_{i-1}V_{i-1}$ bestimmt [Bis06, S. 138]. Es wird also sowohl für die abhängige, als auch für die unabhängigen Variablen metrisches Skalenniveau gefordert. In dem im Folgenden fokussierten Fall nur einer unabhängigen Variablen X ergibt sich die folgende Gleichung, wobei die abhängige Variable mit Z bezeichnet wird.

$$Z(X) = a + bX$$

Die Parameter a und b bilden das Vorwissen. Das Lernen besteht darin, a und b so zu bestimmen, dass sie möglichst gut die Zusammenhänge in den Trainingsdaten widerspiegeln. Dazu wird in der Regel der sogenannte Ansatz der kleinsten Quadrate verwendet [Ste07, S. 57] [Bis06, S. 140]. Dabei wird für jeden Datensatz der Trainingsdaten die Abweichung zwischen dem tatsächlichen Wert von Z und dem über obige Gleichung bestimmten Wert betrachtet [Ste07, S. 56-57]. Ziel ist die Minimierung der Summe der quadrierten Abweichungen in Abhängigkeit von a und b: $\sum_{l=1}^{n}(z_l - (a + bx_l))^2$, wobei der Index l in diesem Zusammenhang nicht unbedingt einen Zeitpunkt, sondern den Datensatz der Trainingsdaten identifiziert [Ste07, S. 57]. Die Minimierung wird erreicht durch die folgende Wahl für a und b, die z.B. bei Steland zu finden ist [Ste07, S. 25, 57].

$$b = \frac{\sum_{l=1}^{n}(x_l - \overline{x})(z_l - \overline{z})}{\sum_{l=1}^{n}(x_l - \overline{x})^2}, \quad a = \overline{z} - b\overline{x} \quad \text{mit} \quad \overline{x} = \frac{1}{n}\sum_{l=1}^{n} x_l \quad \text{und} \quad \overline{z} = \frac{1}{n}\sum_{l=1}^{n} z_l$$

Diese Gleichungen beziehen alle Datensätze der Trainingsdaten ein, eignen sich also für Batch-Lernen. Nach Bishop ist allgemein für lineare Regression auch das notwendige ad-

aptive Online-Lernen möglich [Bis06, S. 143-144]. Dass adaptives Online-Lernen bei der hier betrachteten Form linearer Regression effizient durchgeführt werden kann, erkennt man an den folgenden Ausdrücken, die durch Ausmultiplizieren und weitere einfache Umformungen aus obigen für a und b gewählten Ausdrücken hervorgehen.

$$b = \frac{\sum_{\mathbf{l=1}}^{\mathbf{n-1}} \mathbf{x_l z_l} + x_n z_n + n\,\overline{x}\,\overline{z} - (\sum_{\mathbf{l=1}}^{\mathbf{n-1}} \mathbf{x_l} + x_n)\overline{z} - (\sum_{\mathbf{l=1}}^{\mathbf{n-1}} \mathbf{z_l} + z_n)\overline{x}}{\sum_{\mathbf{l=1}}^{\mathbf{n-1}} \mathbf{x_l^2} + x_n^2 + n\overline{x}^2 - 2(\sum_{\mathbf{l=1}}^{\mathbf{n-1}} \mathbf{x_l} + x_n)\overline{x}}, \quad a = \overline{z} - b\overline{x}$$

$$\text{mit} \quad \overline{x} = \frac{1}{n}(\sum_{\mathbf{l=1}}^{\mathbf{n-1}} \mathbf{x_l} + x_n) \quad \text{und} \quad \overline{z} = \frac{1}{n}(\sum_{\mathbf{l=1}}^{\mathbf{n-1}} \mathbf{z_l} + z_n)$$

Die mit dem Symbol „$\sum$“ eingeleiteten Summen betreffen nun nur noch die ersten n-1 Datensätze der Trainingsdaten. Daher ist es möglich, beim Lernen jedes Datensatzes die Werte der einzelnen Summen mit $1 \leq l \leq n-1$ zu übernehmen und durch Hinzuaddieren des n-ten Summanden den Wert der Summe mit $1 \leq l \leq n$ für das Lernen des nächsten Datensatzes durch Abspeichern bereitzustellen. Durch dieses rekursive Vorgehen wird adaptives Online-Lernen möglich, denn beim Lernen eines Datensatzes werden ältere Trainingsdaten nicht mehr benötigt. Es wird nur eine feste, kleine Anzahl an Parametern gespeichert und beim Lernen einbezogen.

Lineare Regression in der bisher vorgestellten Form setzt voraus, dass ein linearer, deterministischer Zusammenhang zwischen der unabhängigen und der abhängigen Variablen vorliegt. Es besteht jedoch auch die Möglichkeit, die Gleichung zur Bestimmung der abhängigen Variablen um einen sogenannten Störterm ϵ_l zu erweitern: $Z_l = a + bX_l + \epsilon_l$ [Ste07, S. 182]. ϵ_l ist eine Zufallsvariable für den l-ten Datensatz der Trainingsdaten [Ste07, S. 182]. Es wird angenommen, dass ϵ_1, ..., ϵ_n unabhängig und identisch verteilt sind, einer Normalverteilung folgen und den Erwartungswert 0 und eine Varianz > 0 aufweisen [Ste07, S. 183]. ϵ_l eignet sich z.B. zur Darstellung von Messfehlern [Ste07, S. 182]. Bei der Prognose unter Einbeziehung von ϵ_l ist der Wert von X_l gegeben und Z_l ist eine Zufallsvariable mit einer Wahrscheinlichkeitsverteilung. Durch die ursprüngliche, obige Gleichung $Z(X) = a + bX$ wird der Erwartungswert von Z bestimmt.

Beispiel 3.2:
Abbildung 3.6 zeigt, wie Regression zur Folgerungsprognose der Dauer von Telefonaten eingesetzt werden kann (vgl. auch Beispiel 1.2 und 2.8). Die Gerade entspricht der Gleichung $Z(X) = a + bX$. Aus dem Abstand zwischen den Punkten und der Geraden lässt sich die Genauigkeit ablesen.

Neben Folgerungsprognose kann man Regression auch zur Zukunftsprognose einsetzen, indem man einfach eine abhängige Variable wählt, die einen späteren Zeitpunkt als die unabhängigen Variablen betrifft. Unklar ist jedoch, wie erfolgreich dieser Ansatz ist, weshalb Regression hier im Weiteren nur als Methode zur Folgerungsprognose betrachtet wird.

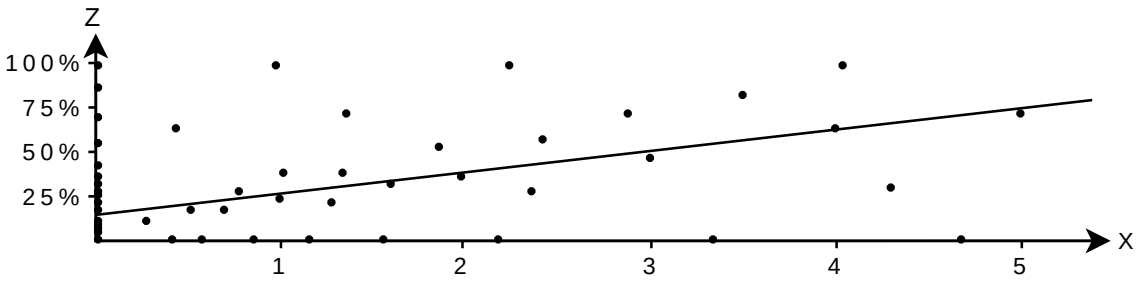

Abbildung 3.6.: fiktive Datensätze von Trainingsdaten für Regression als Punkte und zugehörige Regressionsgerade in einem Graphen (X: mittlere Anzahl offener verpasster Anrufe in einem Zeitraum von 30 Minuten, Z: Anteil der zum Telefonieren verwendeten Zeit in diesen 30 Minuten)

3.3.4. Eignung gewählter Methoden

Um die Güte von Wahrscheinlichkeitstabellen und linearer Regression zur Kontextdatenprognose auf mobilen Geräten genauer zu überprüfen, erfolgt eine Bewertung anhand einiger Anforderungen des Anforderungskatalogs der Arbeit, deren Erfüllung wesentlich von den Methoden abhängig ist. Aus der Literatur werden wenige weitere Anforderungen hinzu genommen. Die folgende Tabelle zeigt, inwieweit die beiden Methoden die Anforderungen erfüllen.

Anforderung	**Wahrscheinlichkeitstabellen**		**Lineare Regression**	
Prognoseart	Folgerungs- und Zukunftsprognose		Folgerungsprognose (Zukunftsprognose nicht betrachtet)	
Unterstützung von Zusammenhangsarten:				
Trends	✘		-	Zukunftsprognose nicht betrachtet
Sequentielle Muster	✔		-	Zukunftsprognose nicht betrachtet
Periodische Muster	(✔)	nur bei üblichen Perioden wie einer Woche	-	Zukunftsprognose nicht betrachtet
Lineare Zusammenhänge	-	kein metrisches Skalenniveau unterstützt	✔	
Nichtlineare Zusammenhänge	-	kein metrisches Skalenniveau unterstützt	✘	
Direkte Unterstützung von Skalenniveaus:				
Nominal	✔		✘	
Ordinal	✘		✘	
Metrisch	✘		✔	
Mehrere unabhängige Variablen [Sig08, S. 231]	✔		-	möglich, aber hier nicht betrachtet
Adaptives Online-Lernen	✔		✔	
Unterstützung von Rollen der Zeit in gesuchten Informationen:				

Anforderung	Wahrscheinlichkeitstabellen		Lineare Regression	
Informationen über Variablen/Ereignisse gesucht	✔		-	Zukunftsprognose nicht betrachtet
Zeitpunkt gesucht	?	realisierungsabhängig	-	Zukunftsprognose nicht betrachtet
Unterstützung von Informationen über Variablen:				
Wahrscheinlichkeits-verteilung	✔		(✔)	durch Störterm
Modalwert	✔	ergibt sich aus Wahrscheinlichkeiten	(✔)	ergibt sich aus Wahrscheinlichkeiten
Erwartungswert und Varianz	✔	ergibt sich aus Wahrscheinlichkeiten	✔	
Fundierte Angabe prinzipieller Unsicherheit	✔		(✔)	durch Störterm
Hohe Generalisierungsfähigkeit	✘	Prognose nur für gelernte Variablenwerte	✔	
Genug Trainingsdaten lernen	(✔)	viele benötigt	✔	eher wenige benötigt
Berücksichtigung räumlich und zeitlich nahen Kontextes	(✔)	erfüllt bei Markovketten	?	realisierungsabhängig
Geringe Methodenunsicherheit	(✔)	einschränkend bei Naive Bayes Classifier: Annahme von Unabhängigkeit, bei Markovketten: Ordnung der Kette	✘	Annahme linearer Zusammenhänge oft nur unzureichend erfüllt
Zukunftsprognose ohne Iterationen [Sig08, S. 235]	(✔)	bei zusätzlichem Speicherbedarf	-	Zukunftsprognose nicht betrachtet
Umgang mit Mängeln in Trainingsdaten:				
Fehlerhafte Werte	(✔)		✘	Problem: extreme Werte [Ste07, S. 60]
Fehlende Werte	?	realisierungsabhängig	?	realisierungsabhängig
Abstraktion beim Lernen	(✔)		✔	
Begrenzbarkeit Ressourcennutzung	(✔)	hoher Speicherbedarf, begrenzt durch Wahl der Variablen etc.	✔	geringer, begrenzter Ressourcenaufwand
Stabil geringe Reaktionszeiten	✔		✔	
Effizienz bei vielen Variablen	(✔)	hoher Speicherbedarf außer bei Naive Bayes	-	nur Fall einer unabh. Variablen betrachtet
Skalierbarkeit	(✔)	hoher Speicherbedarf, Anpassbarkeit durch Anzahl Variablen, Einsatz von Naive Bayes und Ordnung bei Markovketten	(✔)	wenig Ressourcen benötigt, kaum Skalierung zur Verbesserung der Genauigkeit möglich

Wahrscheinlichkeitstabellen und die hier betrachtete Variante linearer Regression zeichnen sich durch ihre einfache Implementierbarkeit und den geringen Berechnungsauf-

wand aus. Zudem ergänzen sich beide Methoden wie angestrebt. Das gilt zum einen im Bereich der Unterstützung von Skalenniveaus und Zusammenhangsarten, wie aus der Tabelle klar wird. Zum anderen ergänzen sie sich auch in ihren Charakteristika bezüglich Genauigkeit und Effizienz.

Wahrscheinlichkeitstabellen bilden Zusammenhänge relativ detailliert ab. Das wirkt sich auf der einen Seite positiv auf die Genauigkeit aus und ermöglicht die Angabe unterschiedlichster Arten gesuchter Informationen. Auf der anderen Seite besteht ein relativ großer Bedarf an Speicherplatz und Trainingsdaten, der mit der Anzahl der Variablen bei direkter Verwendung einer Wahrscheinlichkeitstabelle sogar exponentiell steigt. Hinzu kommt die geringe Generalisierungsfähigkeit. Daher sollte man sich beim Einsatz von Wahrscheinlichkeitstabellen genau überlegen, welche Variablen man einbezieht, und die Wertemengen der Variablen nicht zu groß wählen. Bei drei Bytes pro Wert in der Tabelle sind z.B. drei Variablen mit jeweils 20 möglichen Werten ($20^3 \cdot 3\,\text{Byte} = 24.000\,\text{Byte}$) und vier Variablen mit jeweils zehn möglichen Werten ($10^4 \cdot 3\,\text{Byte} = 30.000\,\text{Byte}$) für die meisten Mobiltelefone und PDAs noch akzeptabel. Die Beschränkung auf Inseln von Zusammenhängen stellt eine gute Voraussetzung für den Speicherbedarf dar. Auch ein hybrider, paralleler Einsatz mehrerer Wahrscheinlichkeitstabellen mit unterschiedlichen unabhängigen Variablen könnte erfolgreich sein. Bei Markovketten eröffnen sich Möglichkeiten der Skalierung durch die Wahl der Ordnung. Eine erheblich geringerer Bedarf an Speicherplatz und Trainingsdaten bietet der Naive Bayes Klassifikator. Der Preis dafür ist eine Erhöhung der Methodenunsicherheit, wenn die restriktiveren Annahmen nicht erfüllt sind, und die Beschränkung auf den Modalwert als gesuchte Informationen.

Lineare Regression nimmt im Gegensatz zu Wahrscheinlichkeitstabellen nur sehr wenig Speicherplatz in Anspruch, benötigt eher wenig Trainingsdaten und generalisiert deutlich besser. Allerdings bildet sie die Zusammenhänge in Trainingsdaten auch viel gröber ab. Genauere gesuchte Informationen als der Erwartungswert und die Varianz können nur unter strengen Annahmen ermittelt werden. Auch ansonsten werden relativ harte Annahmen gemacht, die in vielen Anwendungsszenarien und bei vielen Nutzern nur unzureichend erfüllt sind und deswegen zu einer hohen Methodenunsicherheit führen. Lineare Regression eignet sich somit gut für Geräte mit wenig Ressourcen, wenn die Annahmen näherungsweise erfüllt sind. Auch ein hybrider, paralleler Einsatz neben einer anderen Methode kommt in Frage.

Weiterführend wäre die Prüfung der Eignung zusätzlicher wie Regression stärker generalisierender, jedoch für nominales Skalenniveau geeigneter Methoden wie z.B. von Entscheidungsbäumen sinnvoll. In diesem Bereich besteht noch eine größere Lücke. Zur Prüfung in Frage kommt auch die von Sigg eingesetzte Alignment-Methode zur Zukunftsprognose [Sig08, S. 107-111, 221-227]. Durch das Lernen nur häufiger Folgen von Variablenwerten bietet sie gegenüber Wahrscheinlichkeitstabellen Vorteile beim Bedarf an Speicher und Trainingsdaten und durch approximatives Pattern Matching eine bessere Generalisierungsfähigkeit [Sig08, S. 207, 231]. Zur Verbesserung ihrer Zeitkomplexität könnte ein Ansatz von Shasha et al. helfen [SZ04, S. 96-99, 134-139].

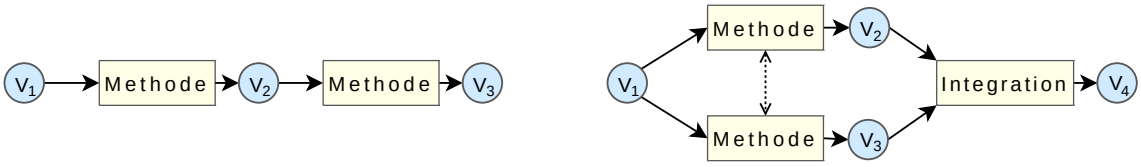

Abbildung 3.7.: einfacher Fall von Chainprocessing (links) und Coprocessing (rechts)

3.4. Hybridität

In Abschnitt 3.2.4 wurde entschieden, im zu entwickelnden Verfahren mehrere, austauschbare Methoden einzusetzen. Die Methoden können sich so in ihrer Funktionalität, z.B. bei der Unterstützung von Zusammenhangsarten ergänzen und Genauigkeit und Effizienz verbessern ↑. Dieser Abschnitt sucht nach Möglichkeiten, wie der gewählte Ansatz der Hybridität realisiert werden kann, wobei die zugrunde liegende Literatur sich auf die Verbesserung von Genauigkeit und Effizienz durch Hybridität konzentriert.

Einen großen Fortschritt für die erwünschte Verringerung der Anforderungen an die Methoden ↑ würde allein schon der Einsatz mehrerer Instanzen einer Methode bewirken, denn jeder Instanz könnte eine Variable, über die Prognosen erstellt werden können sollen, als abhängige Variable zugeteilt werden und die Instanz könnte sich auf die unabhängigen Variablen beschränken, die überhaupt in Zusammenhang mit der jeweiligen abhängigen Variablen stehen. Durch die folgenden, teilweise darauf aufbauenden Möglichkeiten sind weitere Verbesserungen möglich.

3.4.1. Chain- und Coprocessing

Hilario unterscheidet im Bereich der künstlichen Intelligenz verschiedene Arten von Hybridität [Hil95, S. 18-28], die Lenič et al. auch für das Data Mining übernehmen [LKZ+05, S. 307]. Sie unterscheidet zum einen nach dem Grad der Kopplung zwischen den Methoden [Hil95, S. 19]. Zum anderen verwendet sie die folgende Unterteilung nach der Art der Integration der Methoden [Hil95, S. 19-20].

- Chainprocessing
- Coprocessing
- Subprocessing
- Metaprocessing

Der Ansatz des *Chainprocessing* besteht darin, mehrere Methoden hintereinander zu schalten (vgl. Abbildung 3.7). Die Ausgabe der einen Methode dient der nachfolgenden als Eingabe [LKZ+05, S. 307]. Sigg weist darauf hin, dass jeder Verarbeitungsschritt eine mögliche Quelle von Fehlern und Ungenauigkeiten ist [Sig08, S. 20-21, 53, 88, 179]. Insbesondere kann bei manchen Methoden der Fall eintreten, dass sie einen Wert prognostizieren, der in der Realität nie eintritt. Einer nur wenig generalisierenden Methode, die den prognostizierten Wert für eine weitere Prognose verwenden soll, fehlt dann das nötige Vorwissen, weil der prognostizierte Wert nie in den gelernten Trainingsdaten vertreten war. Dennoch kann eine geschickte Aufteilung einer Prognose

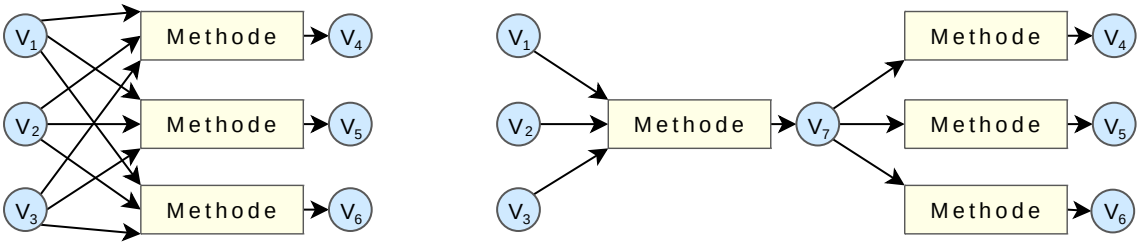

Abbildung 3.8.: Beispiel für direkte Prognose (links) und Chainprocessing (rechts), mögliche Bedeutung der Variablen: V_7: Position, V_1: Position 15 Minuten früher, V_2: Tageszeit, V_3: Wochentag, V_4: WLAN-Verfügbarkeit, V_5: Steckdosen in der Nähe, V_6: Gastronomie in der Nähe

auf hintereinander geschaltete, für die Teilprognosen gut geeignete Methoden im Vergleich zum Einsatz einer einzelnen Methode zu erheblichen Verbesserungen von Genauigkeit und Effizienz führen, wie Hilario anhand realer Beispiele zeigt [Hil95, S. 21-22]. Hilario erwähnt auch eine Vorverarbeitung als Zweck von Chainprocessing [Hil95, S. 20]. Für eine generische Prognose ↑ kommt noch ein weiterer Vorteil hinzu, da viele Anwendungen alternativ über unterschiedliche Variablen Prognosen benötigen. Bei einer direkten Prognose für jede dieser Variablen werden gemeinsame unabhängige Variablen überall einzeln einbezogen, so dass unnötig viel Vorwissen und somit Speicherbedarf entsteht (vgl. Abbildung 3.8). Durch Chainprocessing und Einführen einer Variablen mit nicht zu großem Wertebereich als Zwischenergebnis können positive Synergieeffekte herbeigeführt werden. Bei der Prognose von z.B. V_5 in Abbildung 3.8 kommt dadurch zwar bei der Prognose zu dem aufwändigen Prognoseschritt mit drei unabhängigen Variablen ein zusätzlicher bei geeigneter Wahl von V_7 deutlich einfacherer mit nur einer unabhängigen Variablen hinzu. Das Vorwissen und die entsprechenden Lernvorgänge werden jedoch von drei aufwändigen auf einen aufwändigen und drei eher einfache reduziert. Wenn eine Anwendung zum gleichen Zeitpunkt eine Prognose zu V_4, V_5 und V_6 benötigt, ergibt sich auch bei der Prognose ein deutlicher Vorteil. Insgesamt wird das Lernen von Inseln von Zusammenhängen durch Chainprocessing besser unterstützt.

Ein etwas komplexeres Thema stellt der Hybriditäts-Art des *Coprocessing* dar, die auch Petzold nutzt ↑. Beim Coprocessing arbeiten mehrere Methoden parallel als gleichwertige Partner [Hil95, S. 20] (vgl. Abbildung 3.7). Die Methoden können konkurrieren oder kooperieren und dabei auch Daten austauschen [Hil95, S. 20]. In der Regel wird davon ausgegangen, dass jede Methode ein eigenes Ergebnis hervorbringt [LKZ+05, S. 307] [Bis06, S. 653-654], so dass die potentiell unterschiedlichen Ergebnisse integriert werden müssen. Die bei der Zeitreihenanalyse übliche separate Behandlung mehrerer Komponenten wie z.B. einer Trendkomponente und anderer Komponenten, deren Summe dann den Wert der Zeitreihe ergibt, kann beispielsweise als Coprocessing aufgefasst werden ↑. Theoretische Ergebnisse sichern zu, dass mehrere Methoden zusammen im Coprocessing erheblich genauere Ergebnisse erzielen können als allein, wenn die Voraussetzung erfüllt ist, dass sie unterschiedliche Fehler machen [Die00, S. 1-2] [Bis06, S. 656-657]. Die Ergebnisse mehreren Methoden im Coprocessing sind tatsächlich oft deutlich besser als die

einer einzelnen Methode [Die00, S. 1]. Fehler durch Schätz- und Methodenunsicherheit können sich systematisch aufheben [Bis06, S. 656] [Die00, S. 2-4].

Es sind eine Reihe an Möglichkeiten vorstellbar, um die erwünschte möglichst große *Unabhängigkeit der Fehler* zu erreichen. Man kann unterschiedliche Methoden oder mehrere Instanzen der gleichen Methode mit unterschiedlichen unabhängigen Variablen einsetzen. Zunutze machen kann man sich auch die Unterschiede in den Trainingsdaten [Bis06, S. 655-659]. Ein einfacher Ansatz dafür ist das Zuteilen unterschiedlicher Teile der Trainingsdaten zu den Methoden [Bis06, S. 656]. Ein weiterer Ansatz nennt sich *Boosting*. Beim Algorithmus *AdaBoost* beispielsweise lernen zwar alle Methoden alle Trainingsdaten, aber es entstehen beim Lernen für die einzelnen Datensätze der Trainingsdaten Spezialisten [Bis06, S. 657-659]. Dazu müssen sich die Methoden beim Lernen untereinander abstimmen [Bis06, S. 657-659]. Eine Spezialisierung kann man sich z.B. anhand von Regression so vorstellen, dass bei der Minimierung des quadratischen Fehlers bestimmte Datensätze mit höherem Gewicht berücksichtigt werden.

Eine weitere wichtige Aufgabe für das Coprocessing neben der Erzeugung von Unterschieden im Fehlerverhalten der Methoden liegt in der *Integration der Ergebnisse* der Methoden. Die folgenden Möglichkeiten der Integration sind in der Literatur vorzufinden.

- (Gewichteter) Mittelwert [Bis06, S. 655-656]
- (Gewichteter) Mehrheitsentscheid [CS95, S. 90] [Pet05, S. 89]
- Datenabhängige Auswahl [Bis06, S. 654]
- Probabilistische Mischung [Bis06, S. 654]

Bei der *Mittelwert*-Möglichkeit wird der Mittelwert der Ergebnisse der einzelnen Methoden gebildet [Bis06, S. 656]. Ein *Mehrheitsentscheid* besteht darin, dass eine Mehrheit der Methoden ein Ergebnis ermittelt [Pet05, S. 89]. Mit einer *datenabhängigen Auswahl* ist gemeint, dass der Wertebereich der Kombination der unabhängigen Variablen in Bereiche aufgeteilt wird, die jeweils einer Methode zugeordnet sind [Bis06, S. 654]. Die Integration ist dann also ein Klassifikationsproblem, das z.B. durch Entscheidungsbäume gelöst werden kann [Bis06, S. 663]. Zur Erklärung einer *probabilistischen Mischung* sei angenommen, dass V_1, ..., V_{i-1} unabhängige Variablen sind, V_i die abhängige Variable ist und dass die k-te von K Methoden eine Wahrscheinlichkeitsverteilung $p_k(v_i \mid v_1, ..., v_{i-1})$ als Ergebnis bestimmt (entspricht $P_k(V_i = v_i | V_1 = v_1, ..., V_{i-1} = v_{i-1})$ bei diskreten Variablen). Es wird dann als Gesamtergebnis $p(v_i|v_1, ..., v_{i-1}) = \sum_{k=1}^{K} \pi_k(v_1, ..., v_{i-1}) p_k(v_i|v_1, ..., v_{i-1}))$ bestimmt mit $0 \leq \pi_k(v_1, ..., v_{i-1}) \leq 1$ und $\sum_{k=1}^{K} \pi_k(v_1, ..., v_{i-1}) = 1$ [Bis06, S. 672]. Die sogenannten *Gating-Funktionen* π_k legen fest, in welchen Bereichen der Wertemenge der unabhängigen Variablen welche der sogenannten *Experten* p_k am meisten zuständig sind. Tabelle 3.6 gibt einen Überblick über die Merkmale der Möglichkeiten zur Ergebnisintegration.

Es stellt sich die Frage, wie *Coprocessing beim eigenen Ansatz* eingesetzt werden kann. Durch den ohnehin sinnvollen Einsatz unterschiedlicher Methoden oder Instanzen einer Methode, z.B. zur Unterstützung unterschiedlicher Zusammenhangsarten, ergeben sich

	Mittelwert	Mehrheit	datenabh. Auswahl	probab. Mischung
Ergebnisbildung	**Kombination**	**Auswahl**	**Auswahl**	**Kombination**
Wahrscheinlichk. als Ergebnis	**nicht möglich**	**nicht sinnvoll**	**möglich**	**möglich**
abhängige Variable diskret	**egal**	**nötig**	**egal**	**egal**
abhängige Variable metrisch	**nötig**	**egal**	**egal**	**egal**

Tabelle 3.6.: Merkmale von Möglichkeiten zur Ergebnisintegration beim Coprocessing

bereits *Unterschiede* zwischen den Methoden und somit auch in deren Fehlerverhalten. Weiterführende, aufwendigere Techniken zum Einführen von Unterschieden wie Boosting erscheinen deshalb vorerst nicht sinnvoll.

Welche Möglichkeit der *Ergebnisintegration* sich am besten eignet, ist nicht ganz klar. Der Mittelwert und der Mehrheitsentscheid bieten sich wegen der einfachen Implementierbarkeit für den Anfang an, sind aber nicht bei beliebigen Variablen anwendbar und unterstützen keine Wahrscheinlichkeiten als Ergebnis (vgl. Tabelle 3.6). Die Unterstützung von Variablen mit niedrigerem Skalenniveau könnte durch Ausweichen vom Mittelwert auf den Median oder den Modalwert erreicht werden, wobei der Modalwert dem Mehrheitsentscheid entspricht. Die datenabhängige Auswahl und die probabilistische Mischung weisen am wenigsten Einschränkungen auf (vgl. Tabelle 3.6).

Ein weiterer wichtiger Aspekt der Ergebnisintegration ist die Flexibilität bei der Auswahl bzw. verstärkten Einbeziehung bestimmter Methoden bei den Prognosen, denn neben der Güte der insgesamt besten Methoden und der besonderen Eignung der Methoden für bestimmte Wertebereiche der unabhängigen Variablen ist bei der Ergebnisintegration bei Kontextdatenprognose auch die Variationsunsicherheit ein wichtiger Faktor. Durch Variationsunsicherheit können sich die Genauigkeit und das Fehlerverhalten von Methoden stark ändern. Wenn es gelingt, Variationsunsicherheit zu erkennen, könnte eine Anpassung an die veränderten Zusammenhänge darin bestehen, Methoden zu bevorzugen, die stabil gebliebene Zusammenhänge gelernt haben. Ein weiterer wichtiger Faktor, der sich aus dem adaptiven Online-Lernen ergibt, ist die Schätzunsicherheit, die z.B. beim sogenannten *Warm-Up-Prädiktor* von Petzold eine Rolle spielt [Pet05, S. 87]. Dessen Prinzip besteht darin, neben einer Methode mit hohem Bedarf an Trainingsdaten (langsames „Anlernen“) parallel eine einfache Methode mit schnellem „Anlernen“ einzusetzen [Pet05, S. 87]. Beim Warm-Up-Prädiktor ebenso wie im Allgemeinen sollte berücksichtigt werden, wenn bestimmte Methoden noch eine hohe Schätzunsicherheit aufweisen. Die nötige Flexibilität wird bei der datenabhängigen Auswahl und der probabilistischen Mischung durch die Abhängigkeit von den Werten der unabhängigen Variablen erreicht und beim Mittelwert und dem Mehrheitsentscheid durch eine mögliche Gewichtung. Bei der Gewichtung können die Gewichte beliebig flexibel gewählt werden. Die datenabhängige Auswahl und die probabilistischen Mischung können evtl. noch um eine Abhängigkeit von weiteren Faktoren als den Werten der unabhängigen Variablen erweitert werden.

Neben Chain- und Coprocessing wurden noch *Sub-* und *Metaprocessing* als Arten von Hybridität genannt. Diese Arten sind sich relativ ähnlich. Beim Subprocessing ist eine

Methode in eine andere eingebettet [Hil95, S. 19]. Beim Metaprocessing spielt eine Methode eine Metarolle (z.B. Monitoring oder Kontrolle) [Hil95, S. 19-20]. Sub- und Metaprocessing werden im nächsten Abschnitt wieder aufgegriffen.

3.4.2. Bayes'sche Netze

In Abschnitt 3.3.1 wurden *bayes'sche Netze* bereits kurz vorgestellt und dann zum Einsatz als einzelne Methode in ihrer Allgemeinheit eingeschränkt und mit anderen Methoden zu Wahrscheinlichkeitstabellen zusammengefasst ↑. In ihrer allgemeinen Form kommen bayes'sche Netze jedoch auch zum Einsatz auf einer höheren Ebene für Sub- bzw. Metaprocessing in Frage. Sie beschreiben die Zusammenhangsstruktur zwischen Variablen [Bis06, S. 359-360]. Ein bayes'sches Netz kann daher die Rolle einer übergeordneten, steuernden Instanz übernehmen, die die ihr untergeordneten Methoden koordiniert. Nach der Literatur können lokale „Modelle" wie z.B. Regression Teil eines bayes'schen Netzes sein [May04, S. 74] [Bis06, S. 667]. Abschnitt 3.4.1 hat Möglichkeiten aufgezeigt, wie Methoden im Chain- und im Coprocessing koordiniert werden können. Bayes'sche Netze eröffnen eine etwas weitere Perspektive, indem sie nicht nur eine sequentielle und parallele Verbindung von Methoden, sondern die Verbindung durch eine zyklenfreie Graphstruktur erlauben [Bis06, S. 362].

Bayes'sche Netze als Darstellungsmittel für Zusammenhänge

In Abschnitt 2.4.7 wurden Wahrscheinlichkeitsmaße P als universelle Sichtweise eingeführt, die in diesem Abschnitt als Repräsentant für die Realität aufgefasst werden. Wahrscheinlichkeitsverteilungen zu einem Wahrscheinlichkeitsmaß P, die in diesem Abschnitt alle mit p bezeichnet werden und anhand der Bezeichnungen der Argumente zu unterscheiden sind, ordnen Variablenwerten v_1, ..., v_n eine Wahrscheinlichkeit / Wahrscheinlichkeitsdichte zu (z.B. $p(v_1, ..., v_n)$ oder $p(v_n \mid v_1, ..., v_{n-1})$). Ein bayes'sches Netz kann als Darstellung eines Wahrscheinlichkeitsmaßes P verstanden werden. Borgelt et al. definieren ein bayes'sches Netz zu P als Kombination einer sogenannten *Independence Map* als qualitative Komponente und bedingten Wahrscheinlichkeiten als quantitative Komponente [BK02, S. 101, 114].

Definition 3.1 (Bayes'sches Netz):
Ein *bayes'sches Netz* zu einer Menge von Variablen W und einem zugehörigen Wahrscheinlichkeitsmaß P ist ein gerichteter azyklischer Graph mit den Variablen aus W als Knoten. Jeder Variablen $V_i \in W$ ist genau eine bedingte Wahrscheinlichkeitsverteilung mit Werten $p(v_i \mid \text{parents}(V_i))$ zugeordnet, wobei parents(V_i) ein Tupel mit den Werten der Eltern von V_i ist. Die Kanten des Graphen sind so gewählt, dass der Graph eine *Independence Map* bezüglich W und P ist. (nach [BK02, S. 101, 112-114])

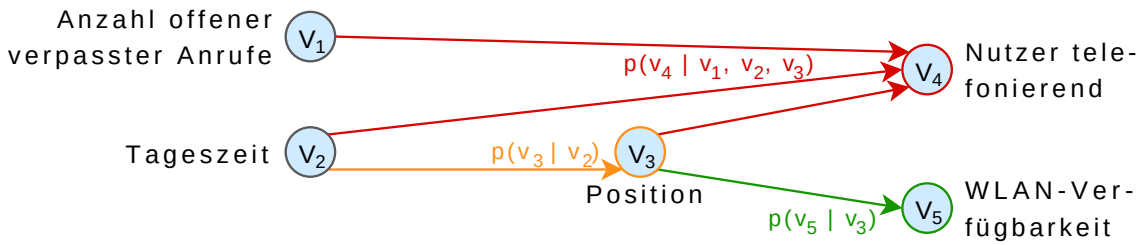

Abbildung 3.9.: Beispiel für ein bayes'sches Netz, angelehnt an Beispiel 1.2 und 2.8

Beispiel 3.3:
Abbildung 3.9 zeigt ein Beispiel für ein bayes'sches Netz. Es stellt eine fiktive Zusammenhangsstruktur zu den Variablen in Beispiel 2.8 über die Nutzung eines Mobiltelefons dar.

Der Begriff der Independence Map und damit verbundene Begriffe können auf Pearl zurückgeführt werden [Pea88]. Eine *Independence Map* zu P spiegelt die *Zusammenhangsstruktur*, also die Existenz von Zusammenhängen im konkreteren Sinne des Wortes zwischen Variablen in der Realität, die durch P als *bedingte Unabhängigkeits*-Beziehungen gegeben sind, als *d-Separation*-Beziehungen wider. Diese beiden Relationen werden im Folgenden nur kurz vorgestellt, da deren genaues Verständnis nicht erforderlich ist.

Die der stochastischen Unabhängigkeit ähnliche, über P definierte Relation der *bedingten Unabhängigkeit* von $W_1 = \{V_1, ..., V_{i-1}\}$ und $W_2 = \{V_i, ..., V_{j-1}\}$ bei gegebenem $W_3 = \{V_j, ..., V_n\}$ (symbolisch: $W_1 \perp\!\!\!\perp W_2 \mid W_3$) ist so definiert, dass $p(v_1, ..., v_{i-1}, v_i, ..., v_{j-1} \mid v_j, ..., v_n) = p(v_1, ..., v_{i-1} \mid v_j, ..., v_n) \cdot p(v_i, ..., v_{j-1} \mid v_j, ..., v_n)$ für alle Wertekombinationen mit $p(v_j, ..., v_n) \neq 0$ [SGS00, S. 10][3]. Das bedeutet z.B. für eine Prognose über W_1, dass die Werte der Variablen in W_2 als gegebene Informationen keinen Zusatznutzen schaffen, wenn die Werte der Variablen in W_3 schon als gegebene Informationen bekannt sind [Bis06, S. 372].

Die Relation der *d-Separation* ist im Vergleich zur bedingten Unabhängigkeit nicht über P, sondern allgemein über gerichtete, azyklische Graphen definiert und kann allein anhand der Struktur des jeweiligen Graphen erkannt werden [Bis06, S. 373, 378] [BK02, S. 98] [SGS00, S. 14]. Seien W_1, W_2 und W_3 disjunkte Variablenmengen. Dann sind W_1 und W_2 bei gegebenem W_3 genau dann d-separated, wenn es keinen nicht blockierten Pfad zwischen einer Variablen aus W_1 und einer Variablen aus W_2 gibt [Bis06, S. 378]. Ein Pfad ist genau dann blockiert, wenn von zwei aufeinander folgenden Kanten 1. beide in gleicher Richtung ($\rightarrow\rightarrow$) oder die erste in Rückwärts- und die zweite in Vorwärtsrichtung ($\leftarrow\rightarrow$) passiert werden und der dazwischen liegende Knoten in W_3 ist oder 2. die erste vorwärts und die zweite rückwärts ($\rightarrow\leftarrow$) passiert wird und weder der dazwischen

[3] Die in [SGS00, S. 10] nicht durchgeführte Generalisierung der Definition bedingter Unabhängigkeit von Zufallsvariablen W_1, W_2 und W_3 zu Mengen W_1, W_2 und W_3 „in offensichtlicher Weise" wird hier so vorgenommen, dass die Definition für Mengen der üblichen Definition für Zufallsvariablen mit $W_1 = (V_1, ..., V_{i-1})$, $W_2 = (V_i, ..., V_{j-1})$ und $W_3 = (V_j, ..., V_n)$ als zu den Mengen W_1, W_2 und W_3 äquivalenten Zufallsvariablen entspricht.

Abbildung 3.10.: Beziehung zwischen P (oben) und einem zugehörigen bayes'schen Netz (unten) auf qualitativer Ebene (links) und quantitativer (rechts)

liegende Knoten noch einer seiner Nachkommen in W_3 ist [Bis06, S. 378].

Ein gerichteter azyklischer Graph ist genau dann eine *Independence Map*, wenn für beliebige disjunkte Variablenmengen W_1, W_2 und W_3 gilt, dass die d-Separation von W_1 und W_2 bei gegebenem W_3 die bedingte Unabhängigkeit von W_1 und W_2 bei gegebenem W_3 impliziert [BK02, S. 101]. Eine Independence Map enthält also nur Unabhängigkeiten, die auch in der Realität existieren, aber möglicherweise auch Abhängigkeiten die es in der Realität nicht gibt. In dem bayes'schen Netz in Abbildung 3.9 ist z.B. die WLAN-Verfügbarkeit d-separated von der Tageszeit bei gegebener Position, was der Realität entspricht, denn bei fester Position kann die Tageszeit die WLAN-Verfügbarkeit nicht indirekt über die Position beeinflussen. Falls die Independence Map noch nicht minimal ist, kann durch Entfernen geeigneter Kanten die Menge der überflüssigen Abhängigkeiten reduziert werden [BK02, S. 98]. Durch Entfernen ungeeigneter Kanten entstehen jedoch Unabhängigkeiten, die in der Realität nicht existieren [Bis06, S. 392-393].

Die den Variablen zugeordneten *bedingten Wahrscheinlichkeiten* mit Werten $p(v_i \mid \text{parents}(V_i))$ stellen den quantitativen Bezug zu P und zur Wahrscheinlichkeitsverteilung p über allen Variablen $V_i \in W$ her: $p(v_1, ..., v_n) = \prod_{i=1}^{n} p(v_i|parents(V_i))$. Abbildung 3.10 veranschaulicht zusammenfassend die Beziehung zwischen P und einem zugehörigen bayes'schen Netz.

Eine bedingte Wahrscheinlichkeit $p(v_i \mid \text{parents}(V_i))$ kann als Prognoseergebnis einer Methode mit V_i als abhängiger und den Eltern von V_i als unabhängigen Variablen angesehen werden (vgl. Abschnitt 2.4.8), wobei Methoden, die nur einen Variablenwert statt einer Wahrscheinlichkeitsverteilung prognostizieren, diesem Wert die Wahrscheinlichkeit 1 zuordnen. Die bedingten Wahrscheinlichkeitsverteilungen des bayes'schen Netzes werden dann also jeweils durch eine Methode repräsentiert. Die in Abbildung 3.7 und 3.8 zu Chain- und Coprocessing explizit dargestellten Methoden und die Integration sind in den Variablen der Graphstruktur bayes'scher Netze inbegriffen.

Prognose mit bayes'schen Netzen

Für die Prognose müssen die den Variablen zugeteilten Methoden koordiniert werden. Prognose mit bayes'schen Netzen (auch *Inferenz* genannt) besteht im Bestimmen von Wahrscheinlichkeitsverteilungen $p(v_i, ..., v_n \mid v_1, ..., v_{i-1})$ bei gegebenen Werten v_1, ..., v_{i-1} [Bis06, S. 393] (vgl. auch Abschnitt 2.4.8). Die Variablen V_1, ..., V_{i-1} bilden also die gegebenen Informationen und die Wahl der Variablen V_i, ..., V_n bestimmt die gesuchten Informationen der Prognose. Trotz Fortschritten in der Forschung garantiert nach Jensen (bei der standardmäßig angenommenen freien Wahl von v_1, ..., v_{i-1} und V_i, ..., V_n) kein

Algorithmus einen akzeptablen Aufwand [Jen01, S. 160] [Bis06, S. 393] (vgl. auch [FN00, S. 364]). Der Einsatz üblicher Algorithmen, die für eine freie Wahl von v_1, ..., v_{i-1} und V_i, ..., V_n konzipiert sind, ist auch deshalb problematisch, weil die Methoden für eine effiziente Prognose bei freier Variablenwahl rückwärts prognostizieren können müssten. Es wäre also erforderlich, dass sie zu einem gegebenen Wert der abhängigen Variablen Informationen über die unabhängigen Variablen ermitteln können, z.B. bei der Ermittlung von $p(v_1 \mid v_4)$ mit v_4=true in Abbildung 3.9. Ansonsten müssten durch ineffizientes testweises Prognostizieren Werte für V_1 gefunden werden, bei denen die Methoden durch Vorwärtsprognose zum Ergebnis V_4=true kommen.

Es existiert jedoch ein Prognosealgorithmus namens *Stochastic Simulation*, der in seiner einfachen Form wegen seiner Nachteile bei freier Variablenwahl kaum in der Praxis eingesetzt wird [Bis06, S. 525] [Jen01, S. 189-191], aber bei ausschließlicher von den gegebenen Variablenwerten ausgehender Vorwärts-Prognose günstige Eigenschaften aufweist, wie im Folgenden ausgeführt wird. Der Algorithmus ähnelt dem intuitiven, beim Chain- und Coprocessing angewendeten Ansatz, dass eine Methode als Werte ihrer unabhängigen Variablen die prognostizierten Werte der Methoden übernimmt, die diesen Variablen zugeordnet sind [Jen01, S. 189-191] [Bis06, S. 525]. Der Algorithmus ermöglicht durch mehrere Durchläufe dieses Vorgehens die Ermittlung von Wahrscheinlichkeiten [Jen01, S. 189-191] [Bis06, S. 525]. Wenn das bayes'sche Netz so erstellt ist, dass die Kanten Kausalitäten ausdrücken, kann Stochastic Simulation als Simulation des Flusses der Wirkungen verstanden werden [Jen01, S. 189].

Das Grundprinzip liegt darin, dass man in einem Durchlauf mit Hilfe eines Zufallszahlengenerators in der Weise zufällig eine Wertekombination aller Variablen wählt, dass die Wahrscheinlichkeit der Wahl einer Wertekombination v_1, ..., v_n der realen, durch P gegebenen Wahrscheinlichkeit entspricht [Jen01, S. 189-190] [Bis06, S. 524-525]. Aus den gewählten Wertekombinationen kann zum Schluss durch Schätzen relativ leicht eine Wahrscheinlichkeitsverteilung als Prognoseergebnis berechnet werden [Jen01, S. 190]. Beim Wählen jeder Wertekombination wird für jede Variable V_i entsprechend ihrer bedingten Wahrscheinlichkeiten $p(v_i \mid parents(V_i))$ zufällig ein Wert v_i gewählt, wobei die Werte $parents(V_i)$ zuvor schon durch die Methoden der unabhängigen Variablen bestimmt wurden [Jen01, S. 190] [Bis06, S. 525]. In Abbildung 3.9 mit der Kante $V_2 \rightarrow V_3$ (V_2: Tageszeit, V_3: Position) könnte z.B. V_2 = 17:30 Uhr gewählt worden sein sowie $p(\text{Arbeit} \mid \text{17:30 Uhr}) = 0,3$ und $p(\text{Auto} \mid \text{17:30 Uhr}) = 0,7$ gelten. Wenn der Zufallsgenerator eine Zahl a zwischen 0 und 1 erzeugt, wählt man V_3 = Arbeit wenn $a < 0,3$ und V_3 = Auto wenn $a \geq 0,3$ [Jen01, S. 190]. Bei Methoden, die direkt einen Variablenwert und keine Wahrscheinlichkeitsverteilung als Ergebnis bestimmen, kann diese zufällige Wahl eines Variablenwertes einfach entfallen. Gewählte Wertekombinationen, die von den gegebenen Informationen abweichende Variablenwerte aufweisen, werden verworfen [Jen01, S. 191]. Dabei ist die Wahl der gegebenen Variablen prinzipiell nicht beschränkt und somit im Endeffekt sogar eine Rückwärtsprognose möglich, obwohl die Methoden nicht rückwärts prognostizieren können.

Eine beliebige Wahl der Variablen der gegebenen und gesuchten Informationen kann jedoch dazu führen, dass sehr viele Wertekombinationen verworfen werden müssen und

so ein großer Rechenaufwand entsteht [Jen01, S. 191]. Inwieweit man dies durch weiterführende, von Stochastic Simulation ausgehende Algorithmen, bei denen auch Schwierigkeiten zu erwarten sind, [Jen01, S. 191-192] [Bis06, S. 523-555] in den Griff bekommen kann und ob diese online eingesetzt werden können, könnte in möglichen weiterführenden Arbeiten geklärt werden. In dieser Arbeit wird der sicherer erscheinende Weg verfolgt, eine Beschränkung auf bestimmte Variablen vorzunehmen. Von Interesse sind Variablen, die für die Verbindung der Variablen der gesuchten Informationen mit den gegebenen wichtig sind. Konkret bietet sich eine Beschränkung auf Pfade an, die von gegebenen Variablenwerten zu Variablen der gesuchten Informationen führen und Kanten ausschließlich vorwärts passieren, da man auf einem solchen Pfad erst bei der letzten der möglichen gegebenen Variablen mit dem Algorithmus beginnen muss, denn der Pfad ist durch diese Variable blockiert (vgl. d-Separation). Auf dem sich ergebenden, mit der einzigen gegebenen Variablen beginnenden Teilpfad kann man dann den Wert dieser ersten Variablen direkt festsetzen und die Werte der nachfolgenden Variablen wie oben beschrieben per Zufall wählen, ohne Wertekombinationen verwerfen zu müssen.

Die Beschränkung auf solche Pfade beeinträchtigt die Genauigkeit auf zwei Arten. Erstens können durch die Beschränkung auf Vorwärts-Pfade nicht alle Variablen mit gegebenem Wert berücksichtigt werden. Eine Beschränkung der möglichen gegebenen Informationen auf bestimmte Variablen führt ebenso wie die Unkenntnis solcher Variablenwerte zu einer Art von Variationsunsicherheit, die erst bei der Prognose entsteht. Wenn man annimmt, dass V_i, ..., V_n mit V_1, ..., V_{i-1} zusammenhängen, aber bei der Prognose z.B. nur die Werte v_1, ..., v_{i-3} angegeben werden, wird als Prognoseergebnis $p(v_i, ..., v_n \mid v_1, ..., v_{i-3})$ bestimmt. Diese Wahrscheinlichkeitsverteilung ist das gleiche Ergebnis, zu dem auch der in Abschnitt 2.4.9 beschriebene Fall führen würde, dass V_{i-2} und V_{i-1} als nicht messbare, äußere Umstände auch beim Lernen, also überhaupt nicht berücksichtigt werden, solange die Kanten und bedingten Wahrscheinlichkeiten des bayes'schen Netzes die Verteilung p der noch berücksichtigten Variablen mit Werten $p(v_1, ..., v_{i-3}, v_i, ..., v_n)$ korrekt abbilden.

Die zweite Beeinträchtigung der Genauigkeit entsteht dadurch, dass berücksichtigte gegebene Variablen nicht immer über alle relevanten Pfade berücksichtigt werden. In Abbildung 3.9 kann z.B. bei gegebener Position (V_3) über den Pfad $V_3 \rightarrow V_4$ bestimmt werden, ob der Nutzer telefoniert. $V_3 \rightarrow V_2 \rightarrow V_4$ stellt jedoch einen weiteren nicht blockierten und trotzdem nicht berücksichtigten Pfad dar, der über die (im angenommenen Fall nicht gegebene) Tageszeit (V_2) führt. Die Prognose verhält sich so, als wenn manche Kanten nicht existieren und zusätzliche d-Separation-Beziehungen bestehen würden, zu denen es in der Realität keine entsprechenden bedingten Unabhängigkeiten gibt. Da die gegebenen Informationen nicht über alle relevanten Pfade in die Prognose der gesuchten Informationen eingebracht werden, kann diese Genauigkeitseinbuße auch als Variationsunsicherheit angesehen werden. Verringert werden kann diese Unsicherheit durch Hinzufügen von Kanten, die potentielle Variablen gesuchter Informationen direkt mit bei Prognosen verwendeten gegebenen Variablen verbinden. Dies bedeutet jedoch eine Einschränkung des Chainprocessing.

Der Vorteil des Einsatzes von Stochastic Simulation in der hier dargestellten Weise

liegt in der Einfachheit und der relativ hoch einzuschätzenden Effizienz. Es muss zwar in jedem Durchlauf des Algorithmus für jede Variable durch die entsprechende Methode eine Prognose durchgeführt werden, aber dies ist unvermeidlich für Methoden, die nur die Werte der unabhängigen Variablen als Eingabe entgegennehmen können, da mehrere mögliche Werte der unabhängigen Variablen berücksichtigt werden müssen, um eine Wahrscheinlichkeitsverteilung als Endergebnis zu erhalten. Besonders positiv hervorzuheben ist auch, dass Stochastic Simulation als Anytime-Algorithmus eingesetzt werden kann, der schon nach einer frei wählbaren Anzahl an Durchläufen ein Ergebnis liefern kann. Zudem besteht die Möglichkeit eines beschränkten Modus mit nur einem Durchlauf, bei dem jede Methode den wahrscheinlichsten Wert prognostiziert. Der Algorithmus bietet also eine ausgezeichnete Skalierbarkeit bezüglich Rechenzeit.

Erstellung bayes'scher Netze

Als weitere wichtige Frage bezüglich des Einsatzes bayes'scher Netze ist zu bedenken, wie man überhaupt in der Praxis bayes'sche Netze erstellt, da das Konzept der d-Separation als mögliche Basis eher wenig intuitiv ist [Bis06, S. 383]. Eine etablierte Variante ist die Nutzung von Expertenwissen [Wit02, S. 52, 88]. Dabei eignet sich die Interpretation einer Kante als kausaler Zusammenhang gut als Entscheidungsgrundlage für Experten beim Ziehen von Kanten, so dass die bedingten Unabhängigkeiten der realen Zusammenhangsstruktur relativ gut auf d-Separation - Beziehungen abgebildet werden [Wit02, S. 17, 88] [Jen01, S. 43-44] (vgl. auch [Jen01, S. 6-11] und [SGS00]). Eine Alternative zu Expertenwissen stellt das automatische Lernen der Struktur aus Trainingsdaten, also das automatische Ziehen von Kanten dar [Wit02, S. 100] [Bis06, S. 418]. Das wird in dieser Arbeit jedoch eher als weiterführend angesehen und nicht genauer betrachtet, da es Algorithmen erfordert, die „im Allgemeinen nicht zum Einsatz zur Laufzeit eines Systems geeignet sind" [Wit02, S. 87]. Zudem kann die Struktur in vielen Fällen überwiegend als Wissen über die Anwendungsdomäne angesehen werden, für das entschieden wurde, dass es zur Entwicklungszeit über das Prognosemodell eingebracht wird (vgl. Abschnitt 3.2.4).

Ein Problem besteht nun z.B. bei kausaler Interpretation der Kanten darin, dass keine Prognosen entgegen der Richtung der Kausalitäten durchgeführt werden können ↑. Wenn ein Gerät z.B. nicht die Position messen kann, die in Abbildung 3.9 als V_3 verwendet wird, würde es sich anbieten, zur Prognose, ob der Nutzer telefoniert (V_4), den indirekten Zusammenhang von V_4 mit der WLAN-Verfügbarkeit (V_5) zu nutzen. Wenn man annimmt, dass der Nutzer zu Hause über WLAN verfügt und überwiegend unterwegs mit seinem Mobiltelefon telefoniert, ist eine erhöhte Wahrscheinlichkeit für ein Telefongespräch zu erwarten, wenn kein WLAN verfügbar ist. Um eine solche Prognose zu ermöglichen, könnte man einfach eine entsprechende Kante $V_5 \rightarrow V_4$ hinzufügen. Allgemein formuliert muss das bayes'sche Netz so umgeformt werden, dass die Methoden in geeigneter Weise miteinander verbunden sind, um zusammen gewünschte Prognosen durchführen zu können. Es ergibt sich der an die Sichtweise von Abschnitt 3.4.1 über Chain- und Coprocessing angelehnte Begriff der Prognosestruktur.

Definition 3.2 (Prognosestruktur):
Die *Prognosestruktur* einer hybriden Prognose ist die Menge der Verbindungen zwischen den Methoden.

Die Graphstruktur bayes'scher Netze wird in dieser Arbeit nicht nur als ein Abbild der Zusammenhangsstruktur, sondern auch als Beschreibung der Prognosestruktur angesehen. Eine Anpassung der Prognosestruktur in Abbildung 3.9 ist z.B. auch für die Prognose der Position (V_3) aus der WLAN-Verfügbarkeit (V_5) nötig. Der naheliegende Ansatz, die Kante $V_3 \rightarrow V_5$ einfach umzudrehen, hat jedoch zur Konsequenz, dass V_2 und V_5 dadurch d-separated werden, wenn V_3 nicht gegeben ist, bzw. dass die WLAN-Verfügbarkeit (V_5) dann nicht mehr durch die Tageszeit als gegebene Informationen prognostiziert werden kann.

Um solche Probleme in komplexeren Netzen zu vermeiden, wird ein Ansatz zur Ermittlung einer geeigneten Prognosestruktur aus der Zusammenhangsstruktur benötigt. Es sind Möglichkeiten zur Erstellung von Netzen mit einer Prognosestruktur denkbar, die auch bestimmte Prognosen entgegen der Richtung der Kausalitäten erlauben. Das Ziel liegt darin, dass die Variablen, über die jemals Prognosen möglich sein sollen, möglichst gut vorwärts über Pfade erreichbar sind von den Variablen, deren Werte grundsätzlich vorgegeben können werden sollen. Trotzdem soll die Zusammenhangsstruktur möglichst genau abgebildet sein. Dazu kann ausgenutzt werden, dass es bei der Konstruktion eines bayes'schen Netzes immer möglich ist, eine zu wahrende totale Ordnung der Variablen vorzugeben, so dass von keinem Knoten V_i eine Kante zu einem Knoten V_j mit $V_i > V_j$ führt [Bis06, S. 360-362] (gilt z.B. für die Ordnung $V_1 < V_2 < V_3 < V_4 < V_5$ in Abbildung 3.9). Durch Zuteilen der höchsten Werte zu den potentiell gesuchten Variablen und der niedrigsten zu den potentiell gegebenen Variablen ist eine verhältnismäßig günstige Prognosestruktur zu erwarten.

Wenn es bei der Erstellung eines Netzes nicht gelingt, dass alle gewünschten Prognosen ausschließlich über Vorwärtsschritte möglich sind, können Kanten hinzugefügt werden. Das Hinzufügen von Kanten beeinträchtigt die Prognosegenauigkeit nicht, da entsprechend der d-Separation nur zusätzliche Zusammenhänge bei Lernen und Prognose betrachtet werden, die möglicherweise in der Realität nicht existieren und damit unnötig sind, oder bestehende Zusammenhänge zusätzlich in direkter Form dargestellt werden (vgl. auch [Bis06, S. 360-361]). Auf der anderen Seite können aber auch Kanten entfernt werden, die weniger von Bedeutung sind, für keine Prognose benötigt werden oder zu Effizienzproblemen führen. Wenn in der Realität existierende Abhängigkeiten vernachlässigt werden oder relevante Variablen gar nicht einbezogen werden (können), entsteht Variationsunsicherheit. Die nicht berücksichtigten Zusammenhänge führen als äußere Umstände zur Variation der beobachteten Zusammenhänge. Insgesamt führt das Entfernen von Kanten eher zu einer höheren Effizienz und einer geringeren Genauigkeit. Schwierig kann die Entscheidung über das Entfernen von Kanten werden, wenn die Zusammenhangsstruktur nicht genau bekannt ist, z.B. wegen inter-individueller Unterschiede zwischen den Nutzern.

Lernen in bayes'schen Netzen

Wenn man mit dem vorgeschlagenen Vorgehen ein Netz erstellt hat, fehlen diesem noch die *bedingten Wahrscheinlichkeiten*, die in dieser Arbeit durch die Methoden bzw. deren Vorwissen repräsentiert werden. Auch hier stehen wieder die Optionen der Nutzung von Expertenwissen und des automatischen Lernens zur Wahl. Dafür kommt nur automatisches Lernen und nicht die Nutzung von Expertenwissen in Frage, um die Anforderung des adaptiven Online-Lernens zu erfüllen. Bei entsprechenden Lernalgorithmen für bayes'sche Netze ist zu unterscheiden zwischen solchen für vollständige Trainingsdaten und solchen, die auch unvollständige Trainingsdaten unterstützen.

Der Umgang mit fehlenden Werten wird schon im Anforderungskatalog der Arbeit gefordert ↑. Anders als beim sporadischen Fehlen von Werten (z.B. fehlende GPS-Koordinaten in Gebäuden) fehlen die Werte der schon bei Hidden Markov Models erwähnten *verborgenen Variablen* sogar permanent (z.B. weil kein GPS-Empfänger vorhanden oder der mentale Zustand des Nutzers nicht messbar ist) [Wit02, S. 94]. Algorithmen für unvollständige Trainingsdaten berücksichtigen solche Variablen in gewisser Weise [Wit02, S. 95]. Wittig führt mehrere Vorteile der Verwendung verborgener Variablen für die Erstellung von Netzen an [Wit02, S. 94-95]. Zudem können durch verborgene Variablen äußere Umstände dargestellt werden, um mit Variationsunsicherheit umzugehen [Jen01, S. 87]. Von Nachteil ist allerdings, dass bei der Platzierung der verborgenen Variablen Annahmen über die Variationsunsicherheit und deren Quellen in der Anwendungsdomäne gemacht werden müssen. Ein weiterer, entscheidender Faktor ist die Komplexität der Algorithmen für unvollständige Trainingsdaten. Es ist zweifelhaft, ob ein Algorithmus existiert, der sich für den Einsatz zur Laufzeit auf mobilen Geräten eignet [Wit02, S. 87, 95] [Wit02, S. 105, 141]. Der für adaptives Lernen übliche AHUGIN-Algorithmus beschränkt sich nach den Ausführungen von Wittig zufolge auf die Bestimmung von Wahrscheinlichkeitsverteilungen und wäre deshalb wahrscheinlich nur für Wahrscheinlichkeitstabellen als Methode geeignet [Wit02, S. 96-98, 105]. Andere Lernalgorithmen für unvollständige Trainingsdaten sind wiederum nicht adaptiv [Wit02, S. 98-100, 104].

Wegen der Schwierigkeiten und eher mäßigen Erfolgsaussichten für den Einsatz von Algorithmen für unvollständige Trainingsdaten beschränkt sich diese Arbeit auf das Lernen vollständiger Trainingsdaten. Im Fall vollständiger Trainingsdaten kann jede Methode lokal und unabhängig von den anderen Methoden aus ihren eigenen Trainingsdaten lernen [Wit02, S. 96]. Die Trainingsdaten für eine Methode umfassen dann Datensätze, die Werte zu der abhängigen und den unabhängigen Variablen der jeweiligen Methode enthalten, wie es auch bisher angenommen wurde. Das lokale Lernen hat auch den Vorteil, dass durch die geringere Anzahl an Variablen in den Datensätzen die Wahrscheinlichkeit erhöht wird, dass kein Variablenwert in einem Datensatz fehlt.

Dynamische bayes'sche Netze

Bisher wurde in diesem Abschnitt nur Folgerungsprognose mit bayes'schen Netzen betrachtet. Der zeitliche Aspekt wird durch *dynamische bayes'sche Netze* (*DBN*) als Spezialfall

von bayes'schen Netzen explizit dargestellt [Wit02, S. 38]. Es existieren mehrere Varianten dieser Form bayes'scher Netze [Wit02, S. 38-40]. DBN unterteilen die Variablen in drei Typen [Wit02, S. 39]: *Statische Variablen* ändern ihren Wert nicht im Verlauf der Zeit [Wit02, S. 39]. Die Werte *temporärer* und *dynamischer Variablen* sind dagegen zeitabhängig [Wit02, S. 39]. Es existieren mehrere Exemplare einer temporären bzw. dynamischen Variablen ($V_{i,1}$, $V_{i,2}$, $V_{i,3}$, ...), so dass in der Regel (abhängig von der DBN-Variante) jedes Exemplar genau einer Zeitscheibe zugeordnet ist [Wit02, S. 39]. Die Zeitscheiben können dann auch als mehrere Exemplare eines bayes'sches Netzes mit temporären und dynamischen Variablen angesehen werden. Temporäre Variablen (bzw. deren Exemplare) sind nur für die jeweilige Zeitscheibe relevant und beeinflussen nur indirekt die Variablen anderer Zeitscheiben [Wit02, S. 39]. Mit dynamischen Variablen lassen sich dagegen z.B. Markovketten darstellen, indem von jedem Exemplar der Variablen eine Kante zum Exemplar in der nächsten Zeitscheibe gezogen wird [Wit02, S. 38-39]. Auch Kanten zwischen Knoten in unterschiedlichen Zeitscheiben sind mit bedingten Wahrscheinlichkeiten versehen [Wit02, S. 39]. Eine Besonderheit von DBN für Algorithmen zur Prognose besteht im Vergleich zu nicht dynamischen bayes'schen Netzen darin, dass mit dem Fortschreiten der Zeit möglicherweise neue Zeitscheiben erzeugt und alte zerstört werden müssen [Wit02, S. 38].

Fazit

Es hat sich in diesem Abschnitt gezeigt, dass in Erwägung gezogene Algorithmen zum Lernen und zur Prognose für bayes'sche Netze und die damit verbundenen Möglichkeiten wegen der Anforderungen der Effizienz, des adaptiven Online-Lernens und des Vorhabens des Einsatzes mehrerer unterschiedlicher Methoden überwiegend nicht oder nur ansatzweise genutzt werden können. Im Bereich der Prognose hat sich jedoch eine Abwandlung der Stochastic Simulation als weiterverfolgte Lösung gefunden. Bayes'sche Netze in der hier dargestellten Form beinhalten außerdem das Konzept des Chainprocessing und ermöglichen die Nutzung des Konzepts des Coprocessing mit den entsprechenden Vorteilen für die Prognose. Bayes'sche Netze eignen sich zudem gut zur Ermittlung und adäquaten Darstellung der Zusammenhangsstruktur sowie gleichermaßen der Prognosestruktur und bieten sich somit als Grundlage für ein Prognosemetamodell zur Erstellung von Prognosemodellen mit mehreren Methoden und als Formalismus zur Ausarbeitung von Prognosemodellen an. Durch die graphische Darstellung der Struktur ist eine Erleichterung bei der Modellierung zu erwarten [Wit02, S. 9].

Für das zu entwickelnde Verfahren bleibt zu klären, wie die prinzipiell notwendigen Aufgaben der Prognose und des Lernens ↑ konkret durch mehrere Methoden und einen koordinierenden Formalismus ausgeführt werden können und wie Wissen über die Anwendungsdomäne dafür als Prognosemodell eingebracht werden kann.

4

Verfahren der strukturierten Kontextdatenprognose

In Abschnitt 3.2.4 wurden bereits ausgehend von den untersuchten Verfahren Entscheidungen bezüglich der Gestaltung eines Verfahrens getroffen, das den Zielen dieser Arbeit möglichst gut gerecht wird. Dies war für die weitere Suche, Auswahl und Bewertung geeigneter Lösungsmöglichkeiten von Bedeutung. Ausgehend von diesen Entscheidungen und den gesammelten Erkenntnissen aus Kapitel 3 wird in diesem Kapitel ein eigenes Verfahren zusammen mit einer entsprechenden Architektur entwickelt. Zur Beherrschung der Komplexität und systematischen Herleitung beschreitet das Kapitel einen Weg der schrittweisen Verfeinerung vom Grundansatz zum Basissystem und letztendlich zu verschiedenen Erweiterungen.

4.1. Grundansatz

Dieser Abschnitt leitet grob den Aufbau und die Funktionsweise eines Prognosesystems her, das nach dem entwickelten Verfahren arbeitet.

4.1.1. Ansatz der strukturierten Prognose

Wie in Abschnitt 3.2.4 gezeigt wurde, ist es vorteilhaft, wenn mehrere, austauschbare Methoden hybrid zum Einsatz kommen ↑ und Wissen über die Anwendungsdomäne von den Anwendungsentwicklern zur Entwicklungszeit über das Prognosemodell eingebracht wird ↑. Dadurch entstehen insbesondere in den Bereichen Prognosegenauigkeit, Effizienz und Flexibilität Vorteile ↑.

Entsprechend sieht das Verfahren vor, dass die Anwendungsentwickler, die das Prognosesystem für ihre Anwendung benutzen möchten, einmalig zur Entwicklungszeit ein anwendungsspezifisches Prognosemodell erstellen. Dieses Vorgehen macht Gebrauch von der Voraussetzung in der Zielsetzung der Arbeit, dass das Prognosesystem in Verbindung mit einer Anwendung eingesetzt wird. Ein Prognosemodell bildet entsprechend der Definition von Prognosemodellen (Definition 2.13) einen Teil des Vorwissens. Es bildet also einen Teil der Zusammenhänge im Prognosesystem ab, die als Gesetzmäßigkeiten das Verhalten des Kontextes und alle Vorgänge darin bestimmen. Das Prognosemodell ist der Teil des Vorwissens, der schon zur Entwicklungszeit feststeht. Neben solchem Vorwissen berücksichtigt das Verfahren auch die bedeutende Anforderung des adaptiven Online-Lernens, die das Erweitern von Vorwissen durch Lernen zur Laufzeit verlangt. Man kann sich dies anhand von Beispiel 1.2 über die Prognose des Nutzungsverhaltens eines Mobiltelefonbesitzers zum Energiemanagement klar machen (vgl. auch Beispiel 2.8 und 3.3). Anwendungsentwickler, die eine solche Prognose realisieren wollen, müssen ein anwendungsspezifisches Prognosemodell erstellen, das z.B. beinhalten kann, dass

ein Zusammenhang im konkreteren Sinne des Wortes zwischen der Anzahl verpasster Anrufe und der Neigung zum Telefonieren besteht und dass darüber hinaus die Neigung zum Telefonieren bei einer kleinen Anzahl verpasster Anrufe in etwa linear mit der Anzahl verpasster Anrufe wächst. Das Prognosesystem ermittelt dann zur Laufzeit durch adaptives Online-Lernen individuell für den jeweiligen Nutzer als weiteres Vorwissen, wie stark die Neigung zum Telefonieren mit der Anzahl verpasster Anrufe steigt.

Zusammenhänge zwischen einzelnen Variablen festzulegen wie den zwischen der Anzahl verpasster Anrufe und der Neigung zum Telefonieren folgt auch dem Prinzip des hybriden Einsatzes mehrerer, austauschbarer Methoden, das in Abschnitt 3.4 vertieft wurde. Prognosen werden unterteilt in kleinere Prognosen, die dann jeweils durch eine Methode durchgeführt werden. Die Methoden haben dann jeweils nur eine kleine Prognoseaufgabe zu bewältigen und können so Prognosen zugeordnet werden, dass sie ihre Stärken voll ausspielen ↑. Eine Methode prognostiziert den Wert ihrer einen abhängigen Variablen aus den Werten ihrer unabhängigen Variablen, deren Werte jeweils wieder durch eine Methode prognostiziert werden. Die Methoden arbeiten jeweils lokal und unabhängig voneinander. Die Methoden können wie beim Chainprocessing hintereinander geschaltet werden oder entsprechend der allgemeineren Sicht bayes'scher Netze als Netz organisiert sein. So kann in Beispiel 1.2 die Telefonieraktivität aus der Position des Nutzers mit seinem mobilen Gerät und der Zeit und die Position wiederum aus der Position zu vorhergehenden Zeitpunkten und der Zeit prognostiziert werden. Zudem bietet sich die Möglichkeit des Coprocessing, also der parallelen Durchführung der gleichen Prognose durch unterschiedliche Methoden zur Verbesserung der Prognosegenauigkeit.

Die folgende Auflistung fasst die Vor- und Nachteile der Prinzipien des hybriden Einsatzes mehrerer, austauschbarer Methoden und des Einbringens von Wissen über die Anwendungsdomäne von den Anwendungsentwicklern in das Prognosemodell aus Kapitel 3 zusammen.

- Einbeziehen von Wissen über Anwendungsdomäne (vgl. Abschnitt 3.2.4):
  - \+ Grundsätzliche Vereinfachung durch genauere Festlegung benötigter Funktionalität und zu untersuchender Zusammenhänge
  - \+ Daraus resultierende Möglichkeit höherer Genauigkeit und Effizienz
- Manuelles Einbringen von Wissen über Anwendungsdomäne zur Entwicklungszeit (vgl. Abschnitt 3.2.4):
  - \+ Komplexität einer automatischen Ermittlung ohne menschliche Erfahrungen vermieden
  - \+ Flexible Berücksichtigung anwendungsspezifischer Anforderungen
  - \- Relativ hohe Anforderungen an Anwendungsentwickler
- Hybridität und Austauschbarkeit von Methoden (vgl. Abschnitt 3.2.4):
  - \+ Bessere Anpassbarkeit an Anwendungsdomäne
  - \+ Geringere Anforderungen an Methoden
  - \+ Kombination der unterschiedlichen Vorzüge von Methoden

- Sequentielle Hybridität (Chainprocessing) (vgl. Abschnitt 3.4.1):
  - \- Ungenauigkeit von zwei Verarbeitungsschritten statt einem
  - \+ Bei gut geeigneten Methoden für einzelne Verarbeitungsschritte bessere Genauigkeit und Effizienz möglich
  - \+ Wenn mehrere Prognosen alle einen gemeinsamen Zwischenschritt nutzen, Möglichkeit geringeren Ressourcenaufwands
- Parallele Hybridität (Coprocessing) (vgl. Abschnitt 3.4.1 ↑ ↑):
  - \+ Höhere Genauigkeit bei unterschiedlichen Fehlern der Methoden
  - \+ Möglichkeit der situationsabhängigen Bevorzugung von Methoden

Eine Verbindung der Prinzipien des Einbeziehens von Wissen über die Anwendungsdomäne zur Entwicklungszeit durch das Prognosemodell und dem hybriden Einsatz mehrerer, austauschbarer Methoden ergibt sich, indem die Anwendungsentwickler im Prognosemodell die Methoden so kombinieren und anpassen können, dass jedem Zusammenhang zwischen Variablen lokal die jeweils optimale Methode zugeteilt ist. Es wird damit Vorwissen, das schwierig zu ermitteln ist, zur Entwicklungszeit als Wissen über die Anwendungsdomäne eingebracht. Dies Schwierigkeit wird durch Abschnitt 3.3.4 und Abschnitt 3.4.2 verdeutlicht. Das Prognosemodell beinhaltet dann die Variablen, ordnet den Variablen Methoden für deren Prognose zu und gibt die Prognosestruktur an, die die Verbindungen zwischen den Methoden beschreibt. Diese Verbindungen spiegeln wieder, welche Zusammenhänge zwischen Variablen in die Prognose einbezogen werden sollen. Von Nachteil ist, dass unter Umständen manche Aspekte des über das Prognosemodell eingebrachten Vorwissens über die Zusammenhänge und die Eignung von Methoden inter-individuellen Unterschieden, also Unterschieden zwischen verschiedenen Nutzern, unterworfen sind. Die Anpassung an den Nutzer ist dann für diese Aspekte nicht optimal.

Da das erarbeitete Verfahren auf einer Struktur von Variablen und Methoden basiert, führt es eine *strukturierte Kontextdatenprognose* durch. Die Struktur wird durch eine auf bayes'schen Netzen basierende Sichtweise dargestellt, da diese eine flexible Graphstruktur und Möglichkeiten der Netzerstellung bietet sowie die Realisierung von Chain- und Coprocessing ermöglicht (vgl. Abschnitt 3.4.2). Durch das Einbringen von Wissen über die erforderlichen Prognosen als Wissen über die Anwendungsdomäne kann die Prognosestruktur in vielen Fällen so gestaltet werden, dass der Verzicht auf Rückwärtsprognose und das damit verbundene Schonen von Ressourcen nicht zu gravierenden Nachteilen führen (vgl. Abschnitt 3.2.4, 3.4.2, 3.4.2).

Das Prognosemodell bildet durch die Kombination von Methoden eine Architektur im Kleinen, die Methoden wie Softwarekomponenten verbindet. Es kann durch die Möglichkeit der Austauschbarkeit von Methoden im Rahmen obiger Prinzipien und durch standardisierte Schnittstellen leicht von den Anwendungsentwicklern für ihre Zwecke angepasst werden. Die Austauschbarkeit von Methoden schließt auch ein, dass von den Anwendungsentwicklern neue Methoden implementiert werden können. Es wird somit ein *Prognoseframework* zur Verfügung gestellt.

Im Rahmen dieses Frameworks können nicht nur Methoden der Zukunftsprognose eingesetzt werden, die Prognosen über die Zukunft durchführen, sondern auch Methoden der Folgerungsprognose (z.B. Clustering zur Vergröberung von Variablenwerten). Methoden der Folgerungsprognose können unter Anderem durch Nutzung kausaler Zusammenhänge einzelne Verarbeitungsschritte beim Chain-Processing übernehmen, um insgesamt eine Zukunftsprognose zu realisieren, z.B. durch die Prognose der Verfügbarkeit von WLAN an einer prognostizierten zukünftigen Position des Nutzers für eine insgesamt gewünschte Prognose der zukünftigen WLAN-Verfügbarkeit. Zudem ergibt sich durch Methoden der Folgerungsprognose ein Zusatznutzen, der über Zukunftsprognose hinaus geht.

Ein weiteres Beispiel für den Einsatz von strukturierter Kontextdatenprognose ist entsprechend der Idee von Petzold die Zukunftsprognose der Position des Nutzers mit (existierenden) spezialisierten Methoden für die Prognose der Position und das Gewinnen weiterer Informationen aus der Position über Folgerungsprognose.

Bei der Prognose z.B. der Position können Methoden, die jeweils auf sequentielle Muster, periodische Muster und Trends als Zusammenhangsarten spezialisiert sind, durch Coprocessing verbunden werden. Solange der Nutzer einem täglichen Schema folgt (z.B. dem Weg zur Arbeit morgens und dem Weg nach Hause abends), kann das Ergebnis der Prognose, die auf periodischen Mustern basiert, verwendet werden. Wenn der Nutzer von diesem Schema abweicht und die Prognose über periodische Muster unsicher ist, kann z.B. die Prognose nach dem Trend, also der Bewegungsrichtung genutzt werden.

In Beispiel 2.2 über den Versand von Daten am Gate eines Flughafens über ein stark ausgelastetes lokales Netz könnte man im Rahmen strukturierter Kontextdatenprognose zunächst die Auslastung des Netzes über den Trend prognostizieren und daraus die verbleibende Zeit für den Versand der Daten. Durch die Unterteilung der Prognose bietet sich auch die Möglichkeit einer verteilten Prognose, bei der die Prognose über die Auslastung des Netzes vom Flughafen bezogen wird, der permanent Zugang zu entsprechenden Trainingsdaten besitzt und vermutlich über leistungsfähigere Systeme als mobile Geräte verfügt.

Basierend auf dem Prognosemodell, das die Methoden konfiguriert und ihre Zusammenstellung festlegt, müssen Lernen und Prognose durchgeführt werden. Das Lernen erfolgt lokal für die einzelnen Variablen bzw. deren Methoden (vgl. Abschnitt 3.4.2). Bei der Prognose werden jedoch mehrere Methoden einbezogen, so dass eine *Koordination* erforderlich ist, die einen Rahmen für die Methoden bildet. Bei der Prognose muss die Güte der lokalen Prognosen der Methoden in globale Güte umgesetzt werden. Dazu muss die Heterogenität der Methoden überbrückt werden. Es werden universelle Schnittstellen benötigt, die die Einbindung möglichst beliebiger Methoden ermöglichen.

4.1.2. Architektur

In Abschnitt 2.3.7 wurde eine grundlegende Architektur genannt, in die das Prognosesystem zusammen mit der Anwendung und einem optionalen Kontextsystem eingebettet ist (vgl. Abbildung 2.3). Dieser Abschnitt entwickelt nun die Architektur des Prognosesystems als Teil dieser Gesamtarchitektur.

Grundaufbau

Die *Architektur* des Prognosesystems selbst ergibt sich aus Lernen und Prognose als prinzipiell notwendige Aufgaben für Kontextdatenprognose (vgl. Abbildung 2.9). Um die Anforderung der Abstraktion von den Trainingsdaten zu erfüllen, muss ein expliziter Lernschritt durchgeführt werden, der die Trainingsdaten in Vorwissen überführt. Es erfolgt also ein Lernschritt, der Kontextdaten als Trainingsdaten verarbeitet und aus den darin enthaltenen Zusammenhängen Vorwissen erzeugt. Das Vorwissen als Abbild der Zusammenhänge dient wiederum als Grundlage zur Prognose. Da Lernen und Prognose zeitlich unabhängig voneinander ablaufen und nur über das Vorwissen miteinander gekoppelt sind, bieten sie sich als Elemente der Architektur an. Für die Organisation dieser Elemente wird das etablierte Prinzip einer Schichtenarchitektur gewählt. Vertikale Schichten wie bei Mayrhofer ↑ und Sigg ↑, durch die die Daten von links nach rechts fließen, sind jedoch weniger geeignet, denn durch den Einsatz mehrerer Methoden ↑ können im Zuge der Beschränkung auf stabile Inseln von Zusammenhängen selektive Prognosen über bestimmte Variablen durchgeführt werden. Es erscheint daher eine Steuerung der Prognose von oben durch die Anwendung oder ein Kontextsystem als sinnvoll. Dies führt zu einer horizontalen Schichtung, die in Abbildung 4.1 dargestellt ist. Das Vorwissen, auf das bei Lernen und Prognose Zugriffe erfolgen, wird als Daten in der untersten Schicht angesiedelt. Die Methoden nehmen sowohl in der Vorwissenschicht, als auch in der Lern- und der Prognoseschicht Raum ein, da konkrete Methoden ihre individuelle Form von Vorwissen besitzen und dieses auf ihre Weise zur Prognose nutzen bzw. beim Lernen aktualisieren. Die Methoden sind innerhalb der Schichten gekapselt und besitzen wohldefinierte Schnittstellen, damit sie austauschbar sind.

Vorwissen

Der erste Schritt für den Einsatz des Prognosesystems, um Prognosen für eine Anwendung durchzuführen, ist das Erstellen eines Prognosemodells durch die Anwendungsentwickler zur Entwicklungszeit der Anwendung (links in Abbildung 4.1). Das erstellte Prognosemodell muss dem Prognosesystem zugeführt werden. Dazu kann es vom Prognosesystem von einem lokalen Speicher oder auch von einem entfernten Gerät geladen werden. Das Prognosemodell beinhaltet einerseits die Prognosestruktur und die Zuordnung von Methoden zu Variablen und andererseits auch Vorwissen der einzelnen Methoden wie z.B. die Größe einer Wahrscheinlichkeitstabelle, die in Abbildung 4.1 zur Illustration verwendet wird.

Neben dem Prognosemodell existiert das Vorwissen auf der Instanzebene, das im Gegensatz zum Prognosemodell erst zur Laufzeit entsteht und deshalb auch nicht nur gelesen, sondern auch lokal oder auf einem entfernten Gerät gespeichert werden kann. Das Vorwissen auf der Instanzebene enthält das beim adaptiven Online-Lernen ermittelte Vorwissen der einzelnen Methoden wie z.B. die Werte in einer Wahrscheinlichkeitstabelle oder die Steigung einer Regressionsgeraden (vgl. Abbildung 4.1).

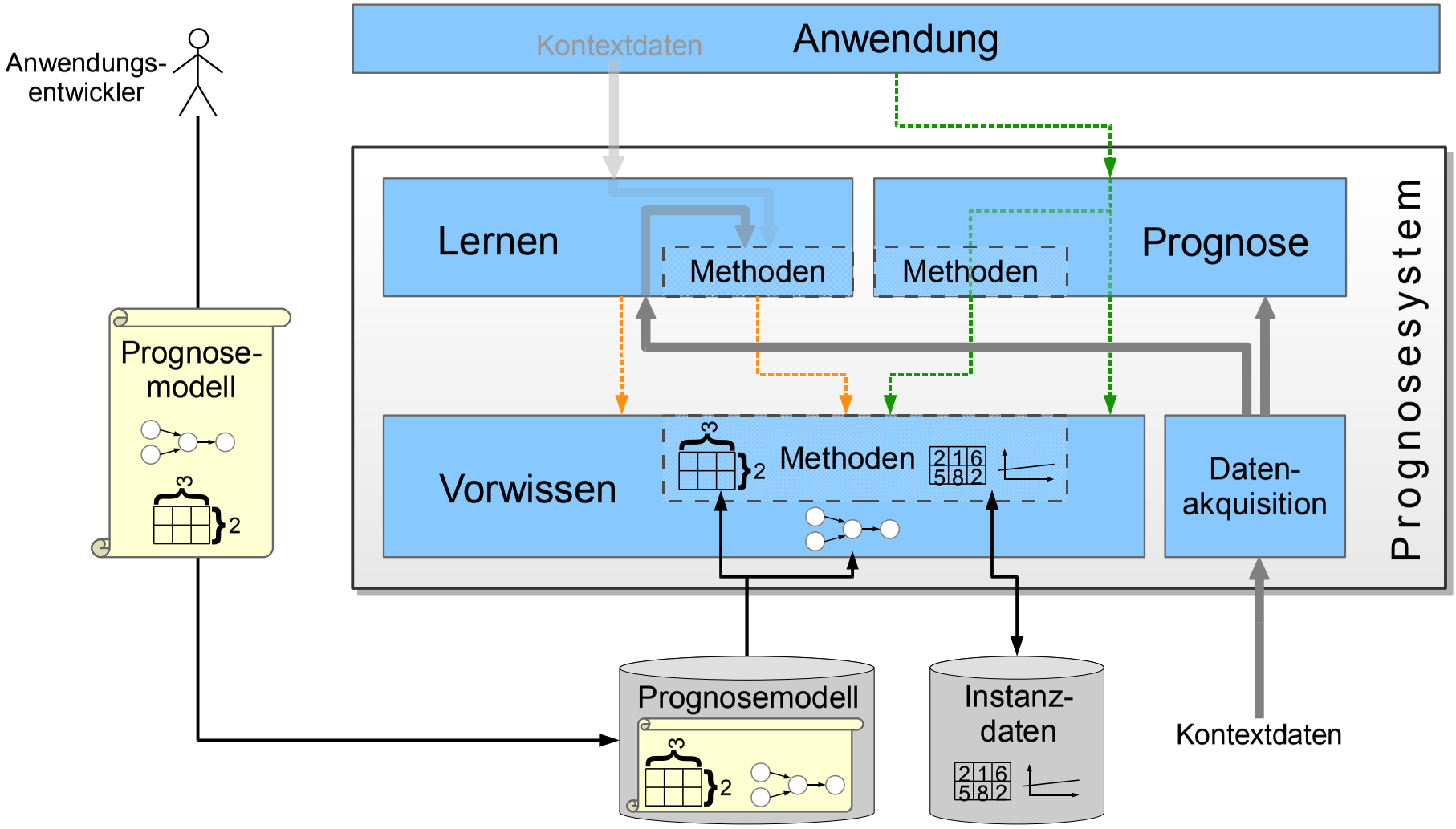

Abbildung 4.1.: Architektur zum entwickelten Verfahren (gestrichelte Pfeile: Benutzt-Beziehungen; durchgängige, dünne Pfeile: Lesen, Speichern oder Übertragen von Daten; dicke, graue Pfeile: Datenflüsse mit Kontextdaten)

Lernen

Das Vorwissen auf der Instanzebene wird beim Lernen aktualisiert bzw. erweitert. Die Lernschicht bezieht dazu Kontextdaten, bildet daraus entsprechend dem Prinzip des induktiven Lernens als bedeutendstem Grundkonzept des Lernens Datensätze und übergibt sie den Methoden als Trainingsdaten, damit sie ihr jeweiliges Vorwissen auf der Instanzebene durch Lernen aktualisieren bzw. erweitern können (vgl. Abbildung 4.1). Ein Datensatz enthält eine Wertekombination der unabhängigen Variablen und der einen abhängigen Variablen der Methode. Ein Datensatz zu Beispiel 1.2 für eine Methode zur Prognose, ob der Nutzer telefoniert, kann z.B. als Variablenwerte enthalten, dass zwei verpasste Anrufe registriert sind und der Nutzer gerade telefoniert. Um auf die einzelnen Methoden zugreifen und ihnen ihre individuell benötigten Datensätze übergeben zu können, muss die Lernschicht mit Hilfe der Prognosestruktur initialisiert werden (linker oranger Pfeil in Abbildung 4.1).

Das Lernen erfolgt normalerweise kontinuierlich und eigenständig durch die Lernschicht, ohne dass die Anwendung darin involviert ist, so dass diese entlastet wird. Die Lernschicht bezieht in diesem Standardfall die zu lernenden Kontextdaten von der Datenakquisitionsschicht (vgl. Abbildung 4.1). Ergänzend dazu ist auch die flexiblere Möglichkeit vorgesehen, dass die Anwendung der Lernschicht zu lernende Kontextdaten übergibt. Da es sich dabei nicht um die primäre Variante des Lernens handelt, ist sie in Abbildung 4.1 blass dargestellt. Die Anwendung kann mit dieser Variante z.B. wie beim Batch-Lernen eine größere Menge an Datensätzen lernen lassen, die aus einer Historie mit Kontextdaten gebildet werden. Das kann von den Anwendungsentwicklern genutzt werden, um zur Entwicklungszeit a priori bekanntes Wissen über Zusammenhänge einzubringen, die sich bei allen Nutzern gleichen. Konkret müssen die Anwendungsentwickler dazu Daten von einem potentiellen Nutzer messen, speichern und diese dann lernen lassen. Die entstandenen Instanzdaten können zusammen mit der entwickelten Anwendung und dem Kontextsystem ausgeliefert werden.

Prognose

Bei der Prognoseschicht stellt im Gegensatz zur Lernschicht das Anstoßen einer Prognose bei Bedarf durch die Anwendung den Standardfall dar. Zusätzlich ist auch eine Publish-Subscribe-Schnittstelle denkbar. Zur Durchführung einer Prognose müssen im Wesentlichen die gegebenen Informationen, also bekannte Werte von Variablen zu bestimmten Zeitpunkten, und Angaben zu den gesuchten Informationen, also den gewünschten Prognoseergebnissen, bekannt sein. Die gesuchten Informationen in Beispiel 1.2 könnten z.B. darin bestehen, dass prognostiziert werden soll, ob der Nutzer in einer Stunde telefonieren wird. Als gegebene Informationen könnten die aktuelle Zeit, die Position in der Gegenwart und jüngeren Vergangenheit sowie die aktuelle Anzahl verpasster Anrufe herangezogen werden. Die gegebenen Informationen werden normalerweise automatisch von der Datenakquisitionsschicht bezogen, so dass die Anwendung nicht mit dem Sammeln und der Angabe dieser belastet wird (vgl. Abbildung 4.1). Die Anwendung muss dann also für die Durchführung eines Prognose nur angeben, worüber sie eine Prognose

benötigt. Alternativ kann sie jedoch auch selbst die gegebenen Informationen angeben. Das kann z.B. nützlich sein, wenn die Anwendung definitives Wissen über die Zukunft besitzt, das dann als gegebene Informationen genutzt werden kann. Wenn es sich bei der Anwendung beispielsweise um ein Navigationssystem handelt und der Nutzer einen Zielort eingibt, steht dem Navigationssystem die Möglichkeit offen, dieses relativ sichere Wissen über die Zukunft in Prognosen einzubringen.

Zur Durchführung einer Prognose greift die Prognoseschicht zum einen auf die Prognosestruktur und zum anderen auf die Methoden zu (vgl. grüne Pfeile in Abbildung 4.1). Die Methoden führen lokal für die einzelnen Variablen Prognosen durch. Die Prognosestruktur wird benötigt, um zu ermitteln, für welche Variablen Prognosen durchgeführt werden müssen und an die Methoden welcher Variablen schon prognostizierte Variablenwerte weitergereicht werden müssen. Es findet eine Koordination der Methoden der einzelnen Variablen statt.

Datenakquisition

Es werden sowohl für das Lernen, als auch die Prognose Kontextdaten benötigt. Für das Beziehen von Kontextdaten ist primär die Datenakquisitionsschicht vorgesehen. Diese enthält Adapter, die von den Anwendungsentwicklern implementiert werden können und z.B. auf Sensoren oder andere Geräte zugreifen. Höhere Schichten können das Beziehen von Daten direkt anfordern. Auch das Abonnieren von Daten über eine Publish-Subscribe-Schnittstelle ist an dieser Stelle denkbar. Ein Spezialfall liegt vor, wenn neben der Anwendung bereits ein Kontextsystem existiert, das von der Anwendung zum Umgang mit Kontextdaten benutzt wird und auf das die Adapter zur Ermittlung von Kontextdaten zugreifen sollen (vgl. Abbildung 2.3). In diesem Fall befindet sich das Prognosesystem abweichend von Abbildung 2.3 in der Gesamtarchitektur neben dem Kontextsystem beziehungsweise zwischen der Datenakquisitionsschicht des Kontextsystems und höheren, auf das Prognosesystem zugreifenden Schichten des Kontextsystems.

Es können entsprechend der Entscheidung in Abschnitt 3.2.4 wie bei Sigg ↑ sowohl Low-Level-Kontextdaten, als auch interpretierte und aggregierte High-Level-Kontextdaten von der Datenakquisitionsschicht bezogen werden. Eine Vorverarbeitung von Kontextdaten wird nicht wie bei Mayrhofer ↑ explizit vorgesehen, da die Zielsetzung der Arbeit voraussetzt, dass die bezogenen Daten schon vorverarbeitet sind.

4.1.3. Einsatz

Das Prognosesystem dient der Unterstützung einer Anwendung, indem es für sie Prognosen über Kontextdaten durchführt (vgl. Abschnitt 2.3.7). Dazu erfolgt eine Anpassung an die Anwendung und die Anwendungsdomäne zur Entwicklungszeit (vgl. Abschnitt 3.2.4). Zur Laufzeit wird das Prognosesystem von der Anwendung für Prognoseaufgaben benutzt ↑. Der genaue Ablauf des Einsatzes des Prognosesystems hängt vom Anwendungsszenario ab. Insbesondere in Szenarien mit starker Verteilung kann der Ablauf komplexere Formen annehmen (vgl. auch Abschnitt 4.4.1). In diesem Abschnitt wird jedoch zunächst ein einfacher Standardablauf beschrieben, der sich überwiegend aus den

bisherigen Ausführungen ergibt.

Zur Veranschaulichung wird Beispiel 1.2 über die Prognose der Telefonieraktivität eines Mobiltelefonbesitzers für das Energiemanagement herangezogen. In diesem Beispiel soll letztendlich prognostiziert werden, wie voll der Akku des Mobiltelefons in der Zukunft sein wird. Diese Prognose soll durch Prognose der Telefonieraktivität erfolgen. Als relevante Variablen bieten sich daher der Ladezustand des Akkus und die Telefonieraktivität an. Die Telefonieraktivität kann z.B. dargestellt werden als Anteil der Zeit, in der telefoniert wurde, an aufeinander folgenden Zeiträumen fester Länge. Zur Prognose der Telefonieraktivität sollten weitere Variablen gewählt werden. Dafür können die Position, die Zeit und die Anzahl offener verpasster Anrufe verwendet werden (vgl. Abbildung 3.9). Als nächster Schritt nach der Wahl von Variablen ist es in der Regel sinnvoll herauszufinden, zwischen welchen Variablen Zusammenhänge bestehen. Dann kann das Prognosemodell erstellt werden, das die gewählten Variablen beinhaltet. Um die Prognosestruktur festzulegen, müssen Verbindungen zwischen den Variablen gezogen werden. Diese können mehr oder weniger stark den realen Zusammenhängen zwischen den Variablen entsprechen. Zur Verringerung des Ressourcenaufwands wird man beispielsweise häufig Vereinfachungen vornehmen (vgl. auch Abschnitt 3.4.2). Abschnitt 4.3 geht genauer auf Möglichkeiten zur Erstellung von Prognosemodellen und der Festlegung der Prognosestruktur ein.

Wenn die Variablen und die Prognosestruktur durch Einsatz von Wissen über die Anwendungsdomäne gewählt sind, folgt die stärker technisch geprägte Ausgestaltung. Es müssen die Skalenniveaus der Variablen (nominal, ordinal oder metrisch) festgelegt werden. Den Variablen müssen Adapter zum Beziehen von deren Werten in der Datenakquisitionsschicht und geeignete Methoden für deren Prognose und das Lernen zugeordnet werden. Für manche Variablen wird man bestehende Adapter und Methoden verwenden können, für andere kann das Implementieren neuer Adapter oder Methoden notwendig werden.

Wenn die Anpassung an die Anwendung abgeschlossen ist, kann das Prognosesystem durch die Anwendung benutzt werden. Die Anwendung kann das Prognosemodell und gegebenenfalls schon gespeichertes Vorwissen auf der Instanzebene laden lassen. Das automatische Lernen kann jederzeit gestartet oder gestoppt werden. Für welche Variablen über welche Adapter in der Datenakquisitionsschicht gelernt werden soll, ist im Prognosemodell festgelegt. Unabhängig vom Lernen können jederzeit Prognosen durchgeführt werden. Die Anwendung kann z.B. prognostizieren lassen, wie voll der Akku in 15 Stunden sein wird, und aufbauend darauf die Aufgabe der Nutzung von Prognosen zum Energiemanagement erfüllen.

4.2. Basissystem

Das Basissystem verfolgt als wesentliche Ziele adaptives Online-Lernen, die Beschreibung prinzipieller Unsicherheit durch Wahrscheinlichkeitsverteilungen, den Einsatz der beiden gewählten Methoden als Referenzkonfiguration und die Erfüllung weiterer, unabdingbarer Anforderungen wie Effizienz.

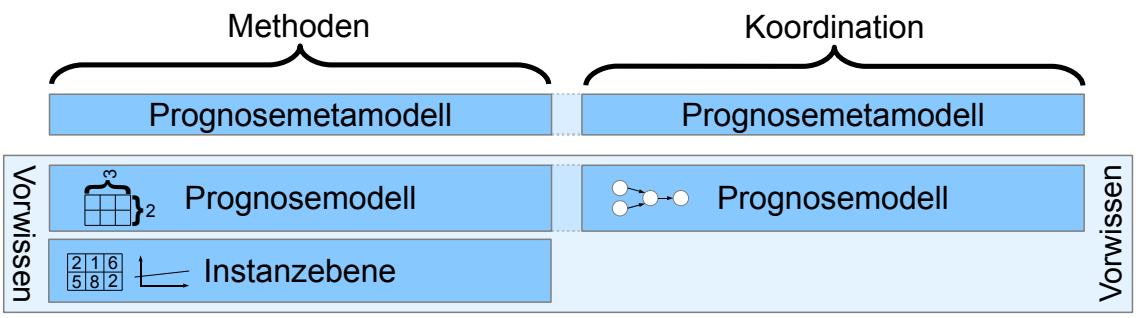

Abbildung 4.2.: Prognosemetamodell und konzeptionelle Unterteilung von Vorwissen nach Zugehörigkeit zu den Methoden oder zum Bereich der Koordination der Methoden sowie nach Modell- und Instanzebene

4.2.1. Vorwissen

Dieser Abschnitt nimmt zunächst eine konzeptionelle Unterscheidung verschiedener Arten von Vorwissen vor und überträgt die Unterscheidung auf die Architektur des Prognosesystems. Dann konzentriert er sich auf eine genauere Spezifizierung des Teils von Vorwissen, das unabhängig von Methoden ist und für die Prognoseschicht eine große Rolle spielt.

Arten von Vorwissen

Der Bereich des Vorwissens kann konzeptionell in verschiedene Teilbereiche unterteilt werden, die in Abschnitt 4.1.2 schon angesprochen wurden. Auf diese Unterteilung wird in den nachfolgenden Abschnitten häufig zurückgegriffen.

Es kann zum einen unterschieden werden zwischen methodenspezifischem Vorwissen (*Methodenvorwissen*) und die Koordination der Methoden betreffendem Vorwissen (*Koordinationsvorwissen*) und zum anderen zwischen dem Prognosemodell und Vorwissen auf der Instanzebene (vgl. Abbildung 4.2). Vorwissen auf der Instanzebene ist komplett methodenspezifisch, da das Lernen entsprechend der getroffenen Entscheidungen bezüglich der Verwendung von Konzepten bayes'scher Netze lokal für die einzelnen Methoden der Variablen erfolgt (vgl. Abschnitt 3.4.2). Das heißt, dass jede Methode separate Datensätze mit den Werten ihrer unabhängigen Variablen und ihrer einen abhängigen Variablen lernt. Das Prognosemodell drückt im Bereich der Koordination die Prognosestruktur und im Bereich der Methoden beispielsweise Parameter wie die Wertemengen der unabhängigen und abhängigen Variablen oder die Größe einer Wahrscheinlichkeitstabelle aus.

Das Prognosemetamodell ist nicht konkret im Prognosesystem enthalten, sondern eher von theoretischer Bedeutung. Es wird im Bereich der Koordination durch die an bayes'schen Netzen orientierte Sichtweise und im Bereich der Methoden durch die jeweiligen Methoden gebildet. Es gibt Syntax und Semantik von Prognosemodellen bzw. einer textuellen oder grafischen Repräsentation dieser vor. Es wird davon ausgegangen, dass das von den Anwendungsentwicklern erstellte Prognosemodell dem Prognosesystem in Form einer solchen Repräsentation zur Verfügung gestellt wird, die dann dieser Syntax entsprechen muss. Im Bereich der Koordination sind z.B. syntaktisch Knoten und Kanten

erlaubt und die Semantik besteht darin, dass entlang der Kanten zwischen den Knoten als Variablen Prognosen durchgeführt werden. Die Syntax sollte für eine Implementierung genau spezifiziert werden. Um die angestrebte Austauschbarkeit von Methoden und die Erweiterbarkeit um neue Methoden sicherzustellen, muss im Bereich der Methoden auch die durch Prognosemetamodelle neuer Methoden gegebene Syntax integriert werden können. Dadurch wird ermöglicht, dass im Prognosemodell methodenspezifische Einstellungen festgelegt werden können wie z.B. im Fall von Wahrscheinlichkeitstabellen ihre Größe, also die Anzahl der Zeilen und Spalten. Das Prognosemetamodell lässt im Bereich der Methoden in der Regel wenig Spielraum für Festlegungen des Prognosemodells bezüglich der Instanzebene. Bei Wahrscheinlichkeitstabellen z.B. werden zwar im Prognosemodell die Anzahl der Zeilen und Spalten festgelegt, in denen die Daten auf der Instanzebene stehen können, aber die Tabellenstruktur an sich ist schon durch das Prognosemetamodell vorgegeben. Daher ist die Syntax des kompletten Vorwissens überwiegend durch das Prognosemetamodell festgelegt. Die Bedeutung des Prognosemodells liegt insbesondere im Bereich der Koordination mehr in Vorgaben für die Vorgänge des Lernens und der Prognose. Diese Vorgänge liegen zwar auch auf der Instanzebene, sind aber nicht in Abbildung 4.2 dargestellt, da es sich nicht um Vorwissen handelt.

Die konzeptionelle Unterteilung zwischen Methodenvorwissen und Koordinationsvorwissen geht unmittelbar in die Architektur des Prognosesystems ein, wie man in Abbildung 4.1 sieht. Dort wird in der Vorwissenschicht der Bereich der Methoden vom restlichen Bereich getrennt. Auch in einer textuellen Repräsentation des Prognosemodells erfolgt diese Trennung, indem die Variablen und die Verbindungen dazwischen auf der einen Seite und das Vorwissen der Methoden auf der anderen Seite separat angegeben werden. Die Methoden müssen dann den Variablen zugeordnet werden. Die Unterscheidung zwischen Methodenvorwissen und Koordinationsvorwissen ist wichtig für die Austauschbarkeit von Methoden.

Die Unterscheidung zwischen Modell- und Instanzebene ist ebenfalls in der Architektur berücksichtigt und wird auch deutlich daran, dass ein persistentes Speichern oder Verschicken des Prognosemodells und des Vorwissens auf der Instanzebene separat erfolgen (vgl. Abbildung 4.1). Eine Trennung von Vorwissen auf der Instanzebene vom Prognosemodell kommt den zusätzlichen Funktionalitäten der Erzeugung von Netzfragmentinstanzen und der Anpassung an Änderungen von Zusammenhängen entgegen, die in Abschnitt 4.4.2 und Abschnitt 4.4.3 thematisiert werden.

Prognosenetze

Eine noch offene Aufgabe für diesen Abschnitt besteht darin, genauer die Syntax und Semantik des Vorwissens zu spezifizieren. Für das Methodenvorwissen braucht dies hier nicht zu erfolgen, da Syntax und Semantik von Methodenvorwissen vom Metamodell der jeweiligen Methoden abhängt und die Methoden selbst für den Zugriff auf ihr Vorwissen zuständig sind. Syntax und Semantik des Koordinationsvorwissens sind durch das Metamodell festgelegt, das sich an bayes'schen Netzen orientiert, jedoch noch nicht genauer festgelegt wurde. Eine genaue Festlegung erfolgt im Folgenden durch die formale Definition eines *Prognosenetzes* und eines *aufgefalteten Prognosenetzes*. Diese Defi-

nitionen spielen eine entscheidende Rolle für die Formulierung von Algorithmen, die eine Koordination der Methoden bei Prognosen in der Prognoseschicht vornehmen. Eine solche Koordination ist notwendig, weil die Methoden jeweils nur ihre unabhängigen Variablen und ihre abhängige Variable kennen und kein Wissen über das Netz besitzen. Durch die Trennung von Methodenvorwissen und Koordinationsvorwissen wird vermieden, dass die Anwendungsentwickler sich beim Implementieren von Methoden mit der Netzstruktur beschäftigen müssen.

Die Definitionen beschreiben die Prognosestruktur, indem sie Variablen als Knoten über Kanten miteinander verbinden. Dabei wird wie bei dynamischen bayes'schen Netzen auch die Zeit berücksichtigt und wie bei der Definition historischer Daten von diskreter Zeit ausgegangen, was keine Einschränkung darstellt [Hüb03, S. 123] [Kri99, S. 5]. Die zugrunde gelegte diskrete Zeit ist dadurch gekennzeichnet, dass Zeitpunkte mit gleichem Abstand betrachtet werden. Zwischen zwei aufeinander folgenden Zeitpunkten liegt ein Zeitschritt. Die Zeitpunkte sind jeweils durch eine ganze Zahlen benannt, wobei situationsabhängig festgelegt werden kann, welchen Zeitpunkt 0 bezeichnet. Unabhängig von der Festlegung des Nullpunktes können jedoch Zeitdifferenzen verwendet werden, wovon die folgende Definition eines Prognosenetzes Gebrauch macht.

Definition 4.1 (Prognosenetz):
Ein *Prognosenetz* ist ein endlicher gerichteter Graph $\mathcal{N} = (W, E)$. Die Knotenmenge W ist eine Menge von Variablen $\{V_1, ..., V_n\}$. Eine Kante $V_i \overset{\Delta}{\rightarrow} V_{i'} := (V_i, \Delta, V_{i'})$ aus der Kantenmenge $E \subseteq W \times \mathbb{N}_0 \times W$ drückt aus, dass bei der Prognose von $V_{i'}$ für einen Zeitpunkt j der Wert von V_i zum Zeitpunkt $j - \Delta$ einbezogen wird. Δ bezeichnet einen Zeitversatz. Abkürzend seien die Schreibweisen $V_i \rightarrow V_{i'} := V_i \overset{0}{\rightarrow} V_{i'}$ und $V_i \overset{1,...,l}{\longrightarrow} V_{i'} := \bigcup_{k=1}^{l} V_i \overset{k}{\rightarrow} V_{i'}$ erlaubt. Ein Prognosenetz enthält keine Zyklen der Art $V_i \overset{\Delta_1}{\rightarrow} ... \overset{\Delta_l}{\rightarrow} V_i$ mit $\sum_{k=1}^{l} \Delta_k = 0$.

In Abbildung 4.3 ist ein mögliches Prognosenetz zu Beispiel 1.2 dargestellt. Eine Kante von einer Variablen Y zu einer Variablen Z drückt aus, dass Y eine unabhängige Variable für die Methode von Z ist. Der (prognostizierte) Wert von Y wird also bei der Prognose von Z einbezogen. Im Gegensatz zum bayes'schen Netz zu Beispiel 1.2 in Abbildung 3.9 können die Kanten im Prognosenetz eine Beschriftung besitzen, die ausdrückt, dass der Wert von Z zu einem bestimmten Zeitpunkt vom Wert von Y abhängt, den Y zu einem früheren Zeitpunkt besaß. Die Kantenbeschriftung gibt die Differenz zwischen diesen beiden Zeitpunkte an. Statt einem Prognosenetz kann auch die gleichwertige Darstellung eines aufgefalteten Prognosenetzes verwendet werden, das keine Zeitdifferenzen verwendet, sondern jede Variable V_i explizit für alle Zeitpunkte separat als $V_{i,j}$ darstellt und dann alle Kanten für alle Zeitpunkte explizit zieht. Ein Prognosenetz wird wegen seiner Kompaktheit zur Formulierung der Prognosestruktur durch die Anwendungsentwickler verwendet. Für Prognosen wird jedoch intern das im Folgenden definierte zugehörige *aufgefaltete Prognosenetz* verwendet, da dieses dem Prognosealgorithmus das Berücksichtigen von Zeitdifferenzen an den Kanten erspart.

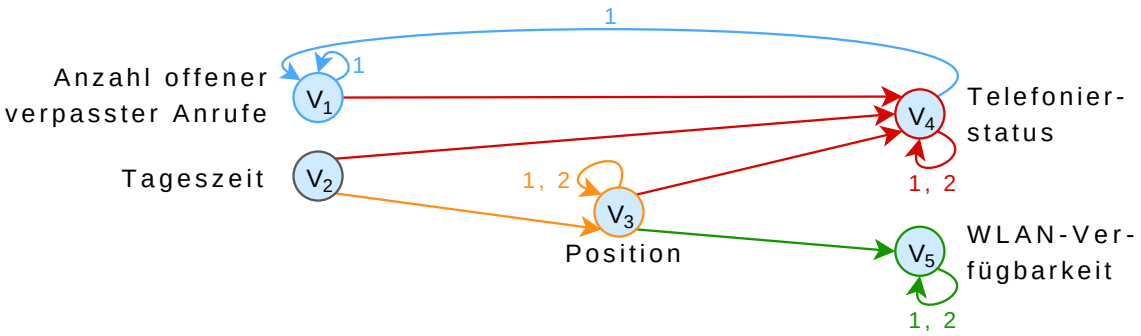

Abbildung 4.3.: Prognosenetz zu Beispiel 1.2, 2.8 und 3.3

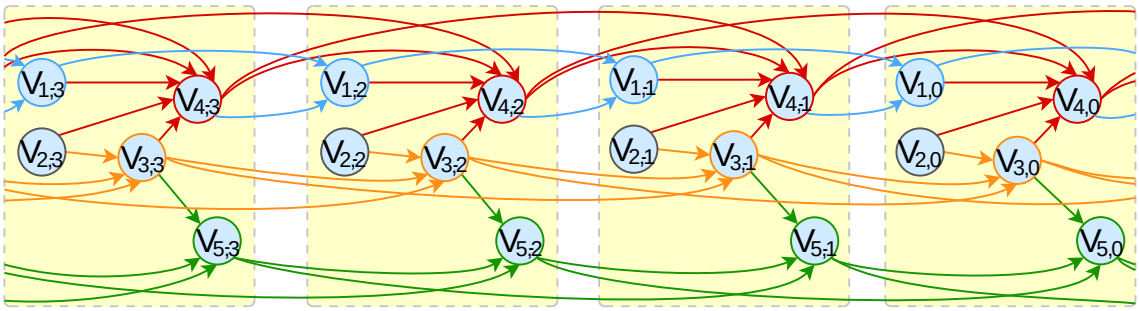

Abbildung 4.4.: Aufgefaltetes Prognosenetz zum Prognosenetz in Abbildung 4.3

Definition 4.2 (Aufgefaltetes Prognosenetz):
Das *aufgefaltete Prognosenetz* zu einem Prognosenetz $\mathcal{N} = (W, E)$ ist ein Graph $\mathcal{N}_\mathcal{A} = (W_A, E_A)$ mit $W_A = \bigcup_{j\in\mathbb{Z}}\{V_{1,j}, ..., V_{n,j}\}$ und $E_A = \{(V_{i,j}, V_{i',j'}) \in W_A \times W_A : \exists\, (V_i, \Delta, V_{i'}) \in E \text{ mit } j' - j = \Delta\}$.

Abbildung 4.4 zeigt ein Beispiel für ein aufgefaltetes Prognosenetz. Ein aufgefaltetes Prognosenetz greift die explizite Unterscheidung von Variablen für unterschiedliche Zeitpunkte j auf, wobei j hier negativ sein kann, damit für eine Prognose auf beliebig ferne Vergangenheit zurückgegriffen werden kann. Das Konzept von Variablen als Sichtweise zur Abbildung der Realität korrespondiert zur universellen Sichtweise auf Prognose in Abschnitt 2.4. Diese passt wegen ihrer Universalität gut zum entwickelten Verfahren, das auf eine hohe Generik abzielt.

Das aufgefaltete Prognosenetz zu einem Prognosenetz kann im Gegensatz zu diesem keine Zyklen enthalten, da sich das Verbot von Zyklen, die sich auf einen Zeitpunkt beschränken, von Prognosenetzen zu aufgefalteten Prognosenetzen überträgt und außerdem wegen $\Delta \geq 0$ keine Kanten von einem Zeitpunkt zu einem früheren erlaubt sind. Ein Beweis für die Zyklenfreiheit von aufgefalteten Prognosenetzen ist in Anhang A zu finden. Entgegen der Zeit gerichtete Kanten sind nicht zugelassen, da bei Zukunftsprognose in Richtung der Zeit prognostiziert wird. Wegen der Zyklenfreiheit kann ein aufgefaltetes Prognosenetz nach Definition 3.1 als bayes'sches Netz zur Variablenmenge W aufgefasst werden, dessen bedingte Wahrscheinlichkeiten durch die den Variablen

zugeordneten Methoden repräsentiert werden.

Man kann sich wie bei einem dynamischen bayes'schen Netz unterschiedliche Zeitscheiben vorstellen, die jeweils ein Exemplar aller V_i mit den entsprechenden Ein- und Ausgangskanten enthalten (vgl. Abbildung 4.4). Zusätzlich wird für die einzelnen V_i in jeder Zeitscheibe das gleiche Vorwissen verwendet. Diese bei dynamischen bayes'schen Netzen übliche Beschränkung [Wit02, S. 39] wird als akzeptabel angesehen, da die Zeitpunkte durch Einführen einer Variablen unterschieden werden können, die die aktuelle Zeit als Wert besitzt und von Methoden als unabhängige Variable benutzt werden kann. Auf eine explizite Darstellung statischer Variablen wie bei dynamischen bayes'schen Netzen wird im Rahmen dieser Arbeit verzichtet, da eine statische Variable in einem Prognosenetz als V_i mit $V_i \xrightarrow{1} V_i$ dargestellt werden kann, wobei die zugehörige Methode den Wert ihrer einzigen unabhängigen Variablen als prognostizierten Wert der abhängigen Variablen übernehmen muss [Wit02, S. 39]. Temporäre und dynamische Variablen werden nicht explizit in obigen Definitionen unterschieden, da der Unterschied irrelevant erscheint für Prognose und Lernen. An den in Prognosenetzen möglichen Zyklen wird deutlich, dass wie in der Systemdynamik Rückkopplungen dargestellt werden können, über die Variablen des Prognosenetzes selbst ihre zukünftigen Werte beeinflussen.

4.2.2. Prognose

Die Prognoseschicht wird nur bei der Durchführung einer Prognose aktiv. Von ihr durchgeführte Prognosen sind unabhängig von früheren Prognosen. Wegen dieser nahe liegenden Forderung benötigt die Prognoseschicht grundsätzlich keinen Zustand, der beim Durchführen von Prognosen verändert wird und über die Prognosen hinaus bestehen bleibt. Eine weitere Konsequenz besteht darin, dass sich dieser Abschnitt mit dem Vorgang einer einzelnen Prognose beschäftigt. Dazu geht er davon aus, dass die Methoden für die Prognose der Werte der einzelnen Variablen vorhanden sind, und arbeitet einen Weg heraus, mit dem der Wert einer Variablen prognostiziert werden kann, ohne dass die Werte der unabhängigen Variablen ihrer Methode gegeben sein müssen. Stattdessen werden die Werte der unabhängigen Variablen wiederum durch deren Methoden prognostiziert, sofern es nötig ist. Auf diese Weise wird bis zu den gegebenen Informationen, also den Variablen, für die Werte gegeben sind, vorgestoßen. Darauf aufbauend werden wiederum höherwertige Prognosemodi entwickelt, die unterschiedliche Arten gesuchter Informationen unterstützen (z.B. Wert einer Variablen gesucht bei gegebener Zeit oder Zeit gesucht, zu der ein Variablenwert eintritt).

Für die Prognose ist eine Koordination der Methoden nötig, da sie nur ihre unabhängigen Variablen und ihre abhängige Variable kennen. Dazu wird das Koordinationsvorwissen verwendet, das das Prognosenetz enthält. Die Koordination der Methoden bei der Prognose erfolgt durch die oben dargestellte Abwandlung von Stochastic Simulation (vgl. Abschnitt 3.4.2). Um diese anwenden zu können, muss zunächst aus dem Prognosenetz das zugehörige aufgefaltete Prognosenetz als bayes'sches Netz generiert werden, das als Grundlage dafür vorgesehen ist ↑, weil Ansätze wie Stochastic Simulation nicht direkt mit der Zeit umgehen können. Das aufgefaltete Prognosenetz ist jedoch unendlich, so dass es nicht vollständig generiert werden kann. Es bietet sich daher schon bei diesem

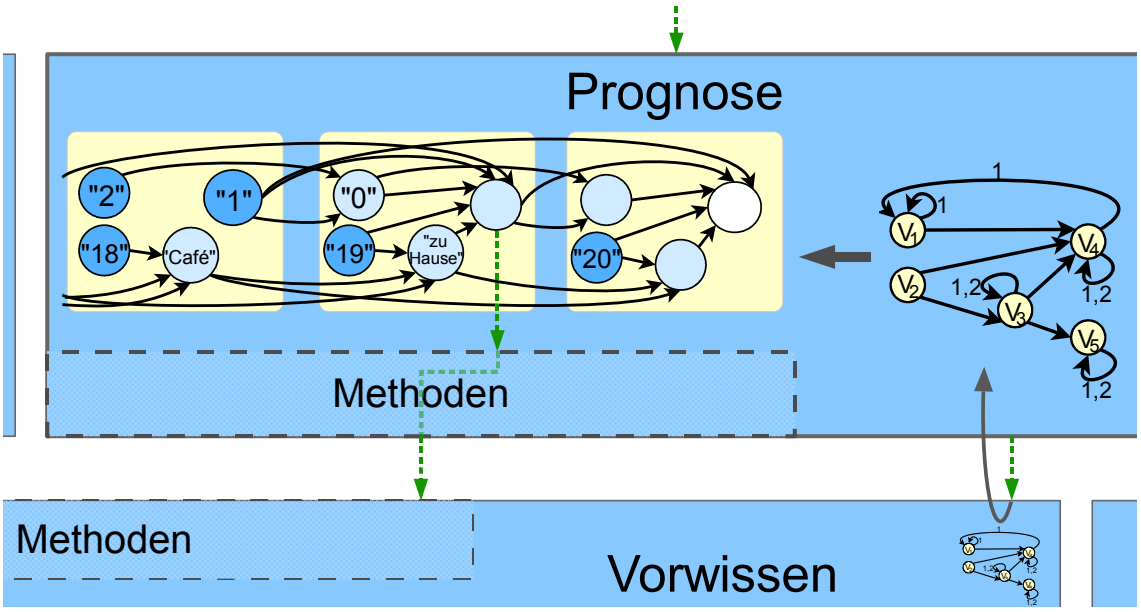

Abbildung 4.5.: Vorgang der Durchführung einer Prognose in entsprechendem Ausschnitt der Architektur des Prognosesystems (vgl. Abbildung 4.1): Zugriff auf Prognosenetz in Vorwissenschicht, Generierung des relevanten Ausschnittes des aufgefalteten Prognosenetzes und Prognose damit; Werte dunkelblauer Variablen gegeben, schon prognostizierte Variablenwerte und linker grüner Pfeil als Momentaufnahme; Netze entsprechen denen in Abbildung 4.3 und 4.6

Generierungsschritt die Anwendung der oben für Stochastic Simulation vorgesehenen Beschränkung auf bestimmte Variablen an ↑, um nur einen relevanten, endlichen Ausschnitt des aufgefalteten Prognosenetzes zu generieren. Derartige Beschränkungen zur Vermeidung von Komplexitätsproblemen werden auch bei Wittig im Rahmen dynamischer bayes'scher Netze erwähnt [Wit02, S. 38].

Die Prognose teilt sich entsprechend den bisherigen Überlegungen auf in einen Generierungsschritt und einen Prognoseschritt im engeren Sinne, der auf dem generierten relevanten Ausschnitt des aufgefalteten Prognosenetzes aufbaut. Der gesamte Vorgang wird durch Abbildung 4.5 veranschaulicht. Beim Generierungsschritt wird auf das Koordinationsvorwissen zugegriffen, in dem das Prognosenetz enthalten ist. Der Prognoseschritt benutzt dann die Methoden der Variablen. Der generierte Ausschnitt des aufgefalteten Prognosenetzes kann nur für die jeweilige Prognose verwendet werden, da es von den gegebenen und gesuchten Informationen der Prognose abhängt, welcher Ausschnitt relevant ist.

Generierung des relevanten Ausschnittes des aufgefalteten Prognosenetzes

Für die Generierung ist eine Auswahl der Variablen im aufgefalteten Prognosenetz erforderlich, welche wie oben vorgesehen für Pfade notwendig sind, die von gegebenen Variablenwerten zu Variablen der gesuchten Informationen führen und Kanten ausschließlich vorwärts passieren (vgl. Abschnitt 3.4.2). Für diese Auswahl wird der Ansatz ge-

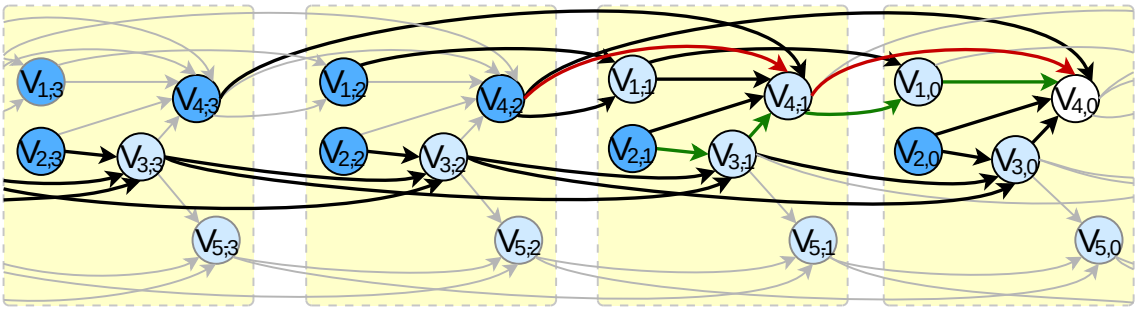

Abbildung 4.6.: Aufgefaltetes Prognosenetz aus Abbildung 4.4 ($V_{1,j}$: Anzahl offener verpasster Anrufe, $V_{2,j}$: Tageszeit, $V_{3,j}$: Position, $V_{4,j}$: Telefonierstatus, $V_{5,j}$: WLAN-Netzverfügbarkeit), $V_{4,0}$ gesucht, Werte der dunkelblauen Variablen gegeben, $V_{3,j}$ und $V_{5,j}$ nicht gegeben da kein GPS-Empfang und WLAN-Modul deaktiviert, Netzausschnitt für Prognose von $V_{4,0}$ bestehend aus dicken Kanten und schwarz berandeten Variablen, darin zwei farblich hervorgehobene Pfade

wählt, ausgehend von den Variablen der gesuchten Informationen nach gegebenen Variablen zu suchen, da dann im häufigen Fall der Prognose des Wertes einer einzelnen Variablen nur von dieser Variablen ausgehend gesucht werden muss. Dabei werden Pfade beschritten, die Kanten ausschließlich rückwärts passieren. Es wird vorausgesetzt, das das aufgefaltete Prognosenetz dabei erst bei Bedarf Stück für Stück aufgebaut wird. Dazu muss für eine erreichte Variable $V_{i',j}$ in einem beschrittenen Pfad im aufgefalteten Prognosenetz zum Fortsetzen des Pfades eine Eingangskante $V_i \overset{\Delta}{\rightarrow} V_{i'}$ der Variablen $V_{i'}$ im (nicht aufgefalteten) Prognosenetz gewählt und müssen die Variable $V_{i,j-\Delta}$ und die Kante $V_{i,j-\Delta} \rightarrow V_{i',j}$ zum aufgefalteten Prognosenetz hinzugefügt werden (vgl. Definition 4.2).

Wenn auf einem Pfad im aufgefalteten Prognosenetz beim Generieren des relevanten Ausschnittes auf eine gegebene Variable gestoßen wird, ist kein Fortsetzen dieses Pfades über die Eingangskanten der gegebenen Variablen nötig ↑. Wenn von einer Variablen $V_{i,j}$ aus über mindestens eine ihrer Eingangskanten kein Pfad zu einer gegebenen Variablen existiert, brauchen keine Pfade über diese Kanten weiterverfolgt zu werden. Damit die Methode von $V_{i,j}$ bei der Prognose die erforderlichen Werte der unabhängigen Variablen erhält, sollten jedoch die betroffenen unabhängigen Variablen nicht entfernt werden. Stattdessen kann man alle Eingangskanten der betroffenen unabhängigen Variablen entfernen und die Werte dieser Variablen jeweils mit einer Methode prognostizieren lassen, die keine unabhängigen Variablen erfordert. Eine solche Methode repräsentiert eine nicht bedingte Wahrscheinlichkeitsverteilung $p(v_{i,j})$ und ist auch für Variablen nötig, die schon im (nicht aufgefalteten) Prognosenetz keine Eingangskanten besitzen, solange nicht garantiert werden kann, dass der Variablenwert bei jeder Prognose gegeben ist. Es muss nun jeder Variablen neben ihrer normalen Methode auch eine Methode zur Prognose ohne unabhängige Variablen im Prognosemodell zugeordnet sein. Abbildung 4.6 zeigt das Ergebnis der Auswahl eines Netzausschnittes.

Um nicht alle Pfade separat absuchen zu müssen, kann das Problem rekursiv, ähnlich wie beim Backtracking gelöst werden. Für eine Variable müssen alle Eingangskan-

ten verfolgt werden. Die Variable wird als Teil des zu bestimmenden Ausschnittes des aufgefalteten Prognosenetzes ausgewählt, wenn über alle Eingangskanten jeweils über mindestens einen Pfad eine gegebene Variable gefunden wurde. Wenn über keine Eingangskante eine gegebene Variable gefunden wurde, wird die Variable zunächst nicht ausgewählt. Wenn nur über einen Teil der Eingangskanten keine gegebenen Variablen gefunden wurden, wird die Variable ausgewählt und es werden auch alle unabhängigen Variablen ausgewählt, über die keine gegebenen Variablen gefunden wurden, jedoch alle ihrer Eingangskanten entfernt. Ob die Variable ausgewählt wird, meldet sie zurück an ihren Vorgänger im Pfad, der von der gesuchten Variablen ausgeht.

Das Problem ist nun, dass das aufgefaltete Prognosenetz im Regelfall unendlich ist, so dass es zu keiner Termination der Suche kommt. Es wurden jedoch die folgenden Kriterien identifiziert, die hinreichende, aber keine notwendigen Bedingungen dafür sind, dass von einer Variablen $V_{i,j}$ aus keine gegebene Variable mehr erreicht werden kann.

(1) $V_{i,j}$ besitzt keine Eingangskanten.

(2) Alle gegebenen Variablen $V_{i',j'}$ liegen in späteren Zeitscheiben als $V_{i,j}$ ($j' > j$).

(3) Für alle gegeben Variablen $V_{i',j'}$ mit $j' \leq j$ gilt, dass V_i im (nicht aufgefalteten) Prognosenetz von keiner Variablen $V_{i'}$ erreichbar ist.

(1) erscheint ebenso wie (2) unmittelbar einleuchtend. (2) ist ein gutes Kriterium, wenn nur gegebene Informationen über die jüngere Vergangenheit zur Verfügung stehen. (3) basiert darauf, dass es zu jedem Pfad $V_{i',j'} \rightarrow \ ... \ \rightarrow V_{i,j}$ im aufgefalteten Prognosenetz einen Pfad $V_{i'} \stackrel{\Delta_1}{\rightarrow} \ ... \ \stackrel{\Delta_l}{\rightarrow} V_i$ im Prognosenetz gibt, was sich per vollständiger Induktion beweisen lässt. (3) wird von (2) impliziert und kann zusätzlich in vielen der weiteren Fälle gelten, in denen noch gegebene Informationen in früheren Zeitscheiben vorhanden aber nicht erreichbar, also irrelevant sind. (3) ist jedoch aufwändiger zu berechnen, z.B. indem man für eine Zeitscheibe j ausgehend von der Menge an Variablen $V_{i'}$, die zu einem Zeitpunkt $j' \leq j$ gegeben sind, solange wie möglich Variablen zur Menge hinzufügt, die mit der Ausgangskante einer bereits in der Menge vorhandenen Variablen verbunden sind. Da unklar ist, inwieweit sich der Aufwand von (3) lohnt, werden im Rahmen dieser Arbeit nur (1) und (2) verwendet.

Die Zeitkomplexität der Generierung eines Ausschnittes des aufgefalteten Prognosenetzes ergibt sich aus der Anzahl an bearbeiteten Variablen, die linear von der Größe des Prognosenetzes und der Anzahl durchsuchter Zeitscheiben abhängt. Ebenso verhält es sich bei der Platzkomplexität im ungünstigsten Fall, dass alle bearbeiteten Variablen als Teil des Netzausschnittes gewählt werden.

Prognose mit dem relevanten Ausschnitt des aufgefalteten Prognosenetzes

Wenn der relevante Ausschnitt des aufgefalteten Prognosenetzes generiert ist, kann damit die Prognose durchgeführt werden. Der Algorithmus für Prognosen mit dem generierten relevanten Ausschnitt des aufgefalteten Prognosenetzes ergibt sich aus der oben beschriebenen Abwandlung von Stochastic Simulation. Für deren Anwendung wird wieder ein rekursiven Vorgehen gewählt, da sich dadurch auf einfache Weise eine geeignete

Prognosereihenfolge ergibt, bei der vor der Prognose eines Variablenwertes die Werte der unabhängigen Variablen prognostiziert werden. Es ergibt sich für einen einzelnen Durchlauf für jede Variable der folgende abstrakte Algorithmus, der auch durch Abbildung 4.5 veranschaulicht wird.

```
1  WENN Variablenwert in aktuellem Durchlauf schon prognostiziert DANN
2      Gib bereits prognostizierten Wert zurück;
3  ANSONSTEN
4      WENN Variablenwert gegeben DANN
5          Gib gegebenen Wert zurück;
6      ANSONSTEN
7          Lasse Werte der unabhängigen Variablen prognostizieren;
8          Prognostiziere eigenen Wert mit Methode der Variablen;
9          Gib prognostizierten eigenen Wert zurück;
10     ENDE WENN
11 ENDE WENN
```

Listing 4.1: abstrakter Algorithmus zur Prognose mit relevantem Ausschnitt des aufgefalteten Prognosenetzes

Bei Methoden mit einer Wahrscheinlichkeitsverteilung als Ergebnis wird der prognostizierte eigene Wert standardmäßig, wie oben für die Abwandlung von Stochastic Simulation beschrieben, per Zufall entsprechend der Verteilung gewählt ↑. Methoden mit einem Variablenwert als Ergebnis können trotz der Orientierung an bayes'schen Netzen, die auf Wahrscheinlichkeitsverteilungen basieren, ganz direkt ohne solche Maßnahmen eingesetzt werden. Auch allein mit Methoden, die nur einen Zeitschritt in die Zukunft prognostizieren können, sind Prognosen über mehrere Zeitschritte hinweg möglich. Trotz der Behandlung von Unsicherheit durch Wahrscheinlichkeiten ist auch der Einsatz von Methoden, die Fuzzy Sets benutzen, grundsätzlich denkbar, wenn die Fuzzy Sets als Variablenwerte dargestellt werden. Wenn beim Einsatz solcher Methoden auf Wahrscheinlichkeiten verzichtet werden soll, kann die Anzahl der Durchläufe der Prognose auf 1 reduziert werden.

Die Prognose kann entsprechend dem Namen „Stochastic Simulation" als Simulation mit mehreren Durchläufen verstanden werden, bei der jeder Durchlauf einem möglichen Wertverlauf der Variablen entspricht, der als historische Daten ausgedrückt werden kann. Durch die Simulation selbst ist ein System definiert, das in allen Simulationsdurchläufen einer Wahrscheinlichkeitsverteilung gehorcht, die die Verteilung des betrachteten Systems in der realen Welt approximiert. Die Wahrscheinlichkeitsverteilung der Simulation ist Teil des Wahrscheinlichkeitsmodells zu den historischen Daten. Die Ermittlung von Prognoseergebnissen geschieht durch Beobachtung mehrerer Durchläufe der Simulation und Schätzen von Parametern oder Verteilungen aus den Durchläufen entsprechend dem frequentistischen Ansatz der Statistik. Dieses Schätzen könnte man auch „Schätzen zweiter Ordnung" nennen. Schätzen zweiter Ordnung unterliegt wie Schätzen erster Ordnung einer gewissen Unsicherheit. Variationsunsicherheit ist ausgeschlossen, da die Methoden in allen Durchläufen auf das gleiche Vorwissen zurückgreifen und daher eine identische, unabhängige Verteilung vorliegt, aber Schätzunsicherheit tritt auf. Je mehr

Durchläufe betrachtet werden, desto geringer ist die Schätzunsicherheit ↑. Gleichzeitig steigt jedoch mit der Anzahl der Durchläufe auch der Zeitaufwand.

Wie viele Durchläufe für gute Ergebnisse notwendig sind, hängt wie bei Schätzungen von Verteilungen, die durch reale Systeme statt Prognosemodelle und Vorwissen auf der Instanzebene erzeugt sind, von den gesuchten Informationen ab und nicht von der Größe des Prognosemodells, aus dem sich die gesuchten Informationen ergeben. Wenn die Verteilung einer nominal skalierten Variablen geschätzt werden soll, kommt es auf die Anzahl ihrer möglichen Werte an. Bei einer metrisch skalierten Variablen ist von Bedeutung, ob ihr Verlauf eher grob oder eher detailreich und zerklüftet ist. Grundsätzlich ist eine Anzahl in der Größenordnung von mehreren hundert Durchläufen in vielen Fällen als ausreichend einzuschätzen, da dieser Wert in die Nähe z.B. der Anzahl befragter Menschen bei Umfragen kommt, bei denen ebenfalls eine Schätzung stattfindet. Ein deutlich höherer Wert erscheint für die meisten Fälle nicht geeignet, da bei Kontextdatenprognose ohnehin mit größeren Ungenauigkeiten durch Variationsunsicherheit zu rechnen ist. Die Möglichkeit der Wahl der Anzahl an Durchläufen führt zu einer guten Skalierbarkeit bezüglich Rechenaufwand ↑. Um die Anzahl an Durchläufen nicht zu niedrig wählen zu müssen, kommt zur Effizienzsteigerung das Cachen der Prognoseergebnisse einer Methode für Werte der unabhängigen Variablen, die in mehreren Durchläufen wiederholt auftreten, in Frage. Das macht besonders Sinn beim Einsatz von Methoden mit höherem Rechenaufwand bei der Prognose, bei wenigen unabhängigen Variablen mit kleinen Wertemengen und bei unabhängigen Variablen, deren Werte häufig in den gegebenen Informationen von Prognosen enthalten sind.

Die Zeitkomplexität des abstrakten Algorithmus in Listing 4.1 ist durch die Anzahl der von Methoden ausgeführten Prognosen gegeben. Zur Herleitung der Komplexität sei $|W|$ die Anzahl der Variablen im Prognosenetz, s die Anzahl an Zeitscheiben des Ausschnittes des aufgefalteten Prognosenetzes und k die Anzahl an Durchläufen. In jedem Durchlauf wird nach Listing 4.1 höchstens eine Prognose pro Variable des Ausschnittes des aufgefalteten Prognosenetzes durchgeführt. Die Anzahl dieser Variablen ist durch $|W| \cdot s$ beschränkt, da der Ausschnitt des aufgefalteten Prognosenetzes nach Definition 4.2 zu jeder Variablen V_i des Prognosenetzes höchstens eine Variable $V_{i,j}$ pro Zeitscheibe enthalten kann. Es werden also in jedem Durchlauf höchstens $|W| \cdot s$ Prognosen durchgeführt. Da k Durchläufe stattfinden, ergibt sich letztendlich $O(|W| \cdot s \cdot k)$ als Zeitkomplexität.

So ergibt sich z.B. auch eine gute Zeitkomplexität bei der Realisierung von Markovketten, für die Sigg $O(n \cdot |W|^2)$ ansetzt, wobei n in etwa s entspricht [Sig08, S. 213-214].

Realisierung höherwertiger Prognosemodi

Aufbauend auf dem ausgearbeiteten Algorithmus werden verschiedene *Prognosemodi* realisiert, die sich in der Art gesuchter Informationen und der Vorgehensweise bei der Prognose unterscheiden.

- *Zustandsprognose*: Es wird eine Wahrscheinlichkeitsverteilung für das Eintreten der Werte aus der Wertemenge einer gesuchten Variablen zu einem bestimmten Zeitpunkt prognostiziert. Dazu finden mehrere Durchläufe statt. Wenn ein Teil der ein-

gesetzten Methoden Wahrscheinlichkeitsverteilungen unterstützt, werden für die gesuchte Variable potentiell unterschiedliche Werte prognostiziert. Aus mehreren dieser Werte wird eine Wahrscheinlichkeitsverteilung geschätzt. Aus der prognostizierten Wahrscheinlichkeitsverteilung kann auf Wunsch der Anwendung auch der Modalwert bestimmt werden.

- *Zeitprognose*: Das Prognoseergebnis besteht in einer Wahrscheinlichkeitsverteilung dafür, dass eine bestimmte Variable zu Zeitpunkten in einem bestimmten Beobachtungszeitraum das erste Mal einen bestimmten Wert annimmt. Es ist die Wahrscheinlichkeitsverteilung der Zeitpunkte gesucht. Aufbauend darauf kann auch für komplexe Aussagen prognostiziert werden, ab wann diese erfüllt sind, wenn die Anwendungsentwickler für eine solche Aussage eine binäre Variable einführen, die angibt, ob die Aussage erfüllt ist, und eine entsprechende Methode zur Bestimmung des Variablenwertes implementieren. Es wird ähnlich wie bei der Zustandsprognose vorgegangen, jedoch für jeden Durchlauf statt des Wertes einer Variablen der Zeitpunkt gespeichert, zu dem der festgelegte Variablenwert das erste Mal eintritt. Für Zeitprognose ist keine spezielle Unterstützung durch die Methoden notwendig.
- *Reduzierter Modus*: Es wird nur ein Durchlauf des Algorithmus ausgeführt, so dass sich der Modus für Geräte mit wenig Ressourcen und Methoden mit höherem Rechenaufwand bei der Prognose eignet. Abweichend vom Standard prognostizieren Methoden, die Wahrscheinlichkeitsverteilungen unterstützen, keinen zufällig entsprechend der Verteilung gewählten Wert, sondern den Modalwert. Zustands- und Zeitprognose können beide jeweils sowohl im reduzierten, als auch im nicht reduzierten Modus eingesetzt werden. Ein im reduzierten Modus ermittelter Wert kann jedoch z.B. bei Zustandsprognose abweichen vom Modalwert der im nicht reduzierten Modus prognostizierten Wahrscheinlichkeitsverteilung als Gesamtergebnis der Prognose, z.B. in Abbildung 4.3 bei der Prognose der WLAN-Verfügbarkeit (V_5) aus der Tageszeit (V_2) über die Position (V_3) bei gegebenem V_2 = 18 Uhr sowie P(V_3 = Café | V_2 = 18 Uhr) = 0,55, P(V_3 = Wohnung | V_2 = 18 Uhr) = 0,45, P(V_5 = kein Netz verfügbar | V_3 = Café) = 0,6 und P(V_5 = kein Netz verfügbar | V_3 = Wohnung) = 0.

Die angebotenen Prognosemodi stehen der Anwendung zur Auswahl, wenn sie eine Prognose durchführen lässt. Sie wählt so gleichzeitig, welche Art gesuchter Informationen sie als Prognoseergebnis wünscht.

4.2.3. Lernen

Neben der Prognose muss das Lernen durchgeführt werden. Das Verfahren der strukturierten Kontextdatenprognose konzentriert sich auf den wichtigen Bereich des adaptiven Online-Lernens, da dieser eine Anpassung an den individuellen Nutzer ermöglicht, ohne die Unsichtbarkeit zu stören (vgl. Abschnitt 2.5.4). Zudem kann a priori vorhandenes Wissen grundsätzlich auch durch das im Vergleich zum Batch-Lernen komplexere adaptive Lernen ins Prognosemodell eingebracht werden, wobei durch die Fokussierung auf

adaptives Online-Lernen allerdings nur bestimmte, dafür geeignete Methoden betrachtet werden.

Da das Lernen für die Variablen als abhängige Variablen lokal durch die entsprechenden Methoden erfolgt (vgl. Abschnitt 3.4.2), müssen diese jeweils die Funktionalität bereitstellen zum Lernen eines Datensatzes, der eine Wertekombination von abhängiger Variable und unabhängigen Variablen enthält. Dieser Abschnitt beschreibt im Folgenden die Teile des Lernens, die nicht von den Methoden übernommen werden.

Der Teil der Lernschicht, der keine Teile von Methoden enthält, hat dafür Sorge zu tragen, dass den Methoden regelmäßig zu lernende Datensätze als Trainingsdaten zugeführt werden. Zur Bildung eines solchen Datensatzes fordert er im Standardfall des eigenständigen Lernens ohne Zutun der Anwendung Werte der abhängigen und der unabhängigen Variablen bei der Datenakquisitionsschicht an (vgl. Abbildung 4.1). Die Zeitpunkte der angeforderten Werte entsprechen den Zeitversätzen der Kanten im Prognosenetz. Für eine Kante $V_i \xrightarrow{3} V_{i'}$ wird z.B. der Wert von V_i zu einem Zeitpunkt j und der Wert von $V_{i'}$ zum Zeitpunkt $j + 3$ benötigt. Wenn ein Wert in einem Datensatz fehlt, kann der Datensatz nicht gelernt werden (vgl. Abschnitt 3.4.2).

Es ist allerdings die zusätzliche Schwierigkeit zu berücksichtigen, dass die Werte von Variablen, die über bestimmte Methoden bestimmt werden, nicht gemessen und damit nicht von der Datenakquisitionsschicht angefordert werden können, jedoch als andere Repräsentation der Werte ihrer unabhängigen Variablen unmittelbar und deterministisch von diesen abhängen. Zur Lösung des Problems werden solche Methoden als *Repräsentationsabbilder* bezeichnet, die die Eigenschaften aufweisen, dass die Prognose deterministisch ist, das Ergebnis bei korrekten Werten der unabhängigen Variablen auch korrekt ist, kein Lernen durchgeführt wird und dass die Methode nur anwendbar ist, wenn alle Eingangskanten der abhängigen Variablen den Zeitversatz 0 besitzen. Wenn beim Lernen für eine Variable der Wert einer ihrer unabhängigen Variablen nicht von der Datenakquisitionsschicht bezogen werden kann und die Methode der unabhängigen Variablen ein Repräsentationsabbilder ist, wird ihr Wert durch Prognose aus den Werten ihrer unabhängigen Variablen prognostiziert. Mit diesen wird rekursiv ebenso verfahren, so dass auch hintereinander geschalteten Repräsentationsabbildern Rechnung getragen wird. Die Eigenschaften von Repräsentationsabbildern erleichtern die Vorbereitung solcher Prognosen.

Von Nachteil bei dieser Lösung ist, dass eine Abhängigkeit zwischen Lern- und Prognoseschicht entsteht, die jedoch nicht zyklisch ist. Zur Vermeidung dieser Abhängigkeit könnte man in Erwägung ziehen, Umwandlungen von Repräsentationen in die Vorwissenschicht zu verlagern und nicht als Aufgabe von Methoden zu begreifen. Umwandlungen von Repräsentationen als eigenständiges, vom Konzept der Methode losgelöstes Konzept im System würden jedoch zu einer Erhöhung der Komplexität des Systems führen, z.B. durch die Vorkehrungen zur Erweiterbarkeit und Austauschbarkeit anwendungsspezifischer Umwandlungen durch die Anwendungsentwickler. Noch gravierender wäre das Problem, dass Umwandlungen von Repräsentationen über mehrere hintereinander geschaltete Variablen hinweg erfolgen können, so dass die rekursive Funktionalität der Prognoseschicht benötigt wird. Eine Benutzung dieser aus der Vorwissenschicht

heraus würde jedoch zu einer zyklischen Abhängigkeit mit einem großen Zyklus führen und kommt damit nicht in Frage.

Dennoch ist weiterführend die Suche nach einer möglichst grundlegenden, konzeptionellen Lösung für das Problem anzustreben, dass mehrere Variablen im Prognosenetz im Grunde die gleiche Variable in der Realität repräsentieren, denn das Problem führt zu weiteren Schwierigkeiten bei Prognosenetzen der folgenden Art. Es sei angenommen, dass eine Variable Z nur eine andere Repräsentation einer Variablen X darstellt. Wenn X gegeben, also gemessen ist, soll Z über einen Repräsentationsabbilder aus X bestimmt werden. Wenn X nicht gegeben ist, soll Z jedoch aus einer Variablen Y prognostiziert werden. Das ist insbesondere auch ein Problem für die Prognose, denn der Methode von Z ist nicht bekannt, ob X gegeben ist oder nicht. Dieses Problem wird auch anhand des Einsatzes des Prognosesystems für die DEMAC-Middleware in Abschnitt 5.3.1 als umfassendes, konkretes Beispiel noch klarer werden.

4.2.4. Datenakquisition

Sowohl bei der Prognose, als auch beim Lernen werden Kontextdaten benötigt. Für die Prognose werden die Werte aller Variablen zur Nutzung als gegebene Informationen in einer kurzen Historie in der Prognoseschicht gespeichert. Beim Lernen werden aus Variablenwerten Datensätze für die einzelnen Variablen mit ihren Methoden gebildet. Die Datenakquisitionsschicht stellt die automatische Variante des Beziehens von Kontextdaten dar ↑, die als primäre Variante angesehen wird, weil sie die Anwendung entlastet und durch die Möglichkeit des Implementierens neuer Adapter zum Akquirieren von Daten durch die Anwendungsentwickler auch schon eine relativ hohe Flexibilität bietet. Bestehende Kontextsysteme besitzen häufig ebenfalls eine untere Schicht, die für das Beziehen von Daten zuständig ist [Tur06, S. 49-55]. Wie und wie oft die Daten für die einzelnen Variablen akquiriert werden sollen, ist im Prognosemodell festzulegen.

Eine einfache Ausgestaltung der Datenakquisitionsschicht sieht so aus, dass sie auf Anforderung die Messung von Daten zu einem gewünschten Zeitpunkt über Messadapter vornimmt (vgl. Abbildung 4.7). Ein Adapter ermittelt einen Wert, der als Wert einer Variablen zu einem bestimmten Zeitpunkt verwendet werden kann. Durch mehrere Anforderungen an einen Messadapter und langsame Messungen können Abweichungen vom gewünschten Messzeitpunkt entstehen, die im Rahmen einer adapterspezifischen Toleranz geduldet werden. Da die Adapter anwendungsspezifisch sein können, wird deren Austausch und das Implementieren neuer Adapter durch die Anwendungsentwickler ermöglicht.

Im Ubiquitous Computing bestehen durch Techniken wie Sensornetze und smarte Gegenstände relativ umfassende Möglichkeiten zum Messen. Eine besondere Herausforderung stellt es dar, wenn High-Level-Kontext wie z.B. die Stimmung des Nutzers aus gemessenem Low-Level-Kontext ermittelt werden soll, um damit Prognosen durchzuführen. Dafür ist vertikale Verarbeitung wie eine Interpretation oder Aggregation nötig (vgl. Abschnitt 2.3.6). Diese kann entweder direkt von den Adaptern durchgeführt werden oder im Netz durch entsprechende Methoden erfolgen. Die Anwendungsentwickler müssen dazu geeignete Adapter bzw. Methoden implementieren. Weiterhin ist die

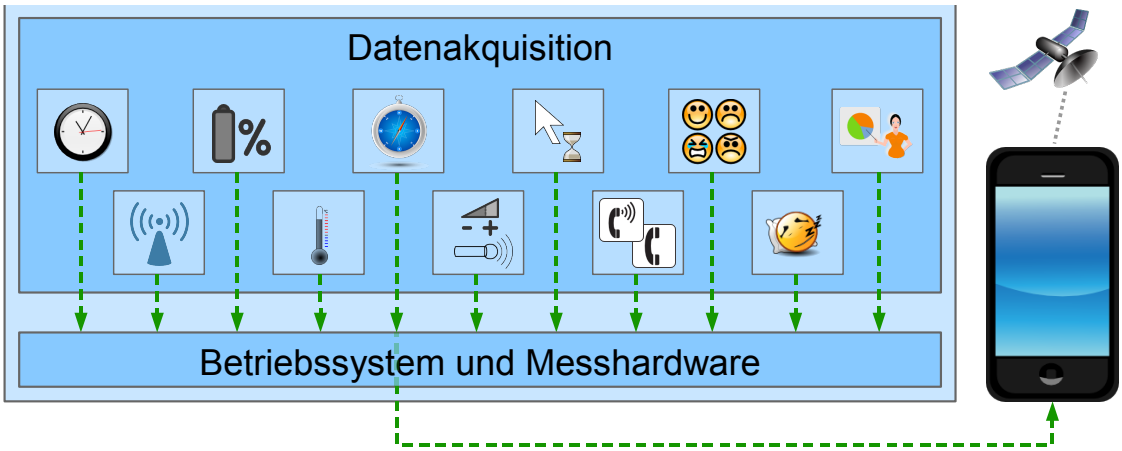

Abbildung 4.7.: einfache Ausgestaltung der Datenakquisitionsschicht mit Adaptern, die bei Bedarf Kontextdaten messen oder von entfernten Geräten anfordern, in Abbildung dargestellte Adapter als Beispiele für vorstellbare Adapter bestimmen Zeit, drahtlose Netzwerke, Akkuladezustand, Temperatur, Position, Lautstärke, Wartezeiten bei der Gerätenutzung, Telefonieraktivität, Stimmung des Nutzers, ob Nutzer schläft und ob eine Präsentation stattfindet

Anwendung von Data-Mining-Techniken zur Vorverarbeitung von Daten wie z.B. der Behandlung von fehlenden Werten und Ausreißern in Erwägung zu ziehen. Fehlende Werte stellen allerdings wegen der Möglichkeit der Prognose von Variablenwerten ohne Werte der unabhängigen Variablen für die Prognoseschicht ohnehin kein Problem dar.

Weitergehende, mögliche Funktionalitäten sind das Einbeziehen ausgetauschter und evtl. auch archivierter Daten (vgl. Abbildung 4.7). Berücksichtigt werden kann auch, dass für die Prognose und für das Lernen Werte für die gleichen Variablen zu unterschiedlichen, jedoch möglicherweise ähnlichen Zeitpunkten angefordert werden. Eine dadurch ermöglichte mehrfache Verwendung von Daten verringert die zu beschaffenden Datenmengen, schont so Ressourcen und vermeidet zeitliche Verzögerungen bei mehreren, konkurrierenden Zugriffen auf die Datenakquisitionsschicht. Um das Potential einer mehrfachen Verwendung von Daten zu nutzen, muss eine zeitliche Abstimmung zwischen Konsumenten von Daten erfolgen, was wegen einer einheitlichen Länge eines Zeitschrittes innerhalb eines einzelnen Prognosesystems unproblematisch ist. Zusätzlich müssen Daten gepuffert werden, solange zu erwarten ist, dass es noch Interessenten gibt, die sie abholen möchten. Insgesamt bietet sich zur Bereitstellung der gleichen Daten für mehrere Abnehmer das Publish-Subscribe-Paradigma an.

4.2.5. Referenzkonfiguration verwendeter Methoden

Methoden spielen sowohl für das Lernen, als auch für die Prognose eine Rolle ↑. Eine Methode wird lokal für eine Variable als abhängige Variable der Methode eingesetzt ↑ und operiert ausschließlich auf den Werten der abhängigen und der unabhängigen Variablen. Zur Kapselung und Entlastung der Methoden wird eine Kenntnis der Netzstruktur durch Methoden vermieden. Methoden können anwendungsspezifisch von den An-

wendungsentwicklern gewählt und zusammengestellt werden ↑, wobei die Möglichkeit des Implementierens von Methoden eingeschlossen wird. Da jedoch viele Methoden für verschiedenste Anwendungen geeignet sind, wird in dieser Arbeit eine Basis für eine Referenzkonfiguration vorgeschlagen.

Die Referenzkonfiguration beinhaltet Wahrscheinlichkeitstabellen und einfache lineare Regression als in Abschnitt 3.3.1 gewählte Methoden. Die Wahl dieser Methoden basiert darauf, dass sie sich gut ergänzen und so eine sinnvolle Grundlage für eine Referenzkonfiguration bilden ↑. Wahrscheinlichkeitstabellen bilden Zusammenhänge relativ genau ab und neigen zu einem hohen Speicherbedarf ↑. Sie unterstützen Variablen mit nominalem Skalenniveau und fallen somit in die Klasse der Methoden, die Klassifizierung durchführen ↑. Lineare Regression generalisiert dagegen stark und benötigt kaum Speicherplatz ↑. Sie unterstützt Variablen mit metrischem Skalenniveau und fällt daher in die Klasse der Methoden, die Regression durchführen, also eine Prognose, bei der die abhängige Variable metrisches Skalenniveau aufweist ↑. Neben den sich ergänzenden Eigenschaften der gewählten Methoden ist außerdem vorteilhaft, dass Wahrscheinlichkeitstabellen bereits erprobt sind im Bereich der Kontextdatenprognose und sowohl Zukunfts- als auch Folgerungsprognose unterstützen ↑.

Zusätzlich zu Wahrscheinlichkeitstabellen und einfacher linearer Regression werden eine Reihe weniger aufwendiger, überwiegend mehr spezialisierter Methoden als Teil der Referenzkonfiguration angeboten, für die dennoch in vielen Anwendungsfällen Bedarf besteht.

- Relativ universelle Methoden:
  - Wahrscheinlichkeitstabellen
  - Einfache lineare Regression
  - Empirische Verteilungsfunktion
- Stark spezialisierte Methoden:
  - Zeitinkrementierer
  - Zeitperiodisierer
  - Binning
  - 2D-Binning
  - Polynomextrapolation
  - Mittelwert
  - Mehrheitsentscheid

Universelle Methoden

Wahrscheinlichkeitstabellen entsprechen der Darstellung in Abschnitt 3.3.2. Sie speichern, wie häufig die einzelnen Wertekombinationen der unabhängigen und der abhängigen Variablen beim Lernen aufgetreten sind, und ermöglichen, daraus Wahrscheinlichkeitsverteilungen zu bestimmen. Sie können unmittelbar auch als Markovketten benutzt

werden. Verzichtet wird im Rahmen dieser Arbeit auf die aufwändigere Realisierung eines Naive Bayes Klassifikators, der auf mehrere Wahrscheinlichkeitstabellen zurückgreifen würde. Ein Naive Bayes Klassifikator würde bei Erfüllung der entsprechenden Unabhängigkeitsannahme gegenüber einer einzelnen, großen Wahrscheinlichkeitstabelle Speicherplatz sparen. Zur als weiterführende Aufgabe ansehbaren Verringerung des hohen Speicherbedarfs von Wahrscheinlichkeitstabellen könnte auch ausgenutzt werden, dass von der bei hohem Speicherbedarf sehr großen Menge an Wertekombinationen der abhängigen und unabhängigen Variablen in den meisten Anwendungsbereichen vermutlich nur relativ wenige überhaupt auftreten. Dies könnte z.B. für eine Komprimierung genutzt werden, die jedoch einen schnellen Datenzugriff und eine Vergröberung nur weniger wichtiger Bereiche der Daten gewährleisten müsste. Zudem wäre zu verhindern, dass das Inkrementieren eines Wertes durch eine vergröberte Speicherung der Daten zu keiner Veränderung führt, so dass der Wert unveränderbar wird. Eine weitere Möglichkeit wäre adaptives Clustering wie z.B. bei Mayrhofer ↑, jedoch nur lokal angewendet auf die Wertekombinationen der unabhängigen Variablen. Dazu müsste jedoch insbesondere von der dahinter geschalteten Methode unterstützt werden, dass sich die Wertemenge des Clusteringergebnisses mit der Zeit ändern kann ↑. Letztendlich ist eine Verringerung des Speicherbedarfs von Wahrscheinlichkeitstabellen wünschenswert, aber nicht von fundamentaler Bedeutung, da das Verfahren der strukturierten Kontextdatenprognose zum einen durch das Konzept des Coprocessing eine Aufteilung der unabhängigen Variablen auf mehrere parallele Prognosen ermöglicht und zum anderen durch die Austauschbarkeit von Methoden in kritischen Fällen bezüglich des Speicherbedarfs die Option des Ausweichens auf andere Methoden bietet.

Für metrisches Skalenniveau wird mit Regression entsprechend Abschnitt 3.3.3 bereits eine entsprechende Methode angeboten. Sie prognostiziert den Wert ihrer abhängigen Variablen $Z_{j'}$ aus dem Wert ihrer unabhängigen Variablen X_j durch Ausrechnen von $Z_{j'}(X_j)$ in der linearen Gleichung $Z_{j'}(X_j) = a + bX_j$ und schätzt beim Lernen die Parameter a und b. Dabei wird als zufällige Komponente auch ein Störterm berücksichtigt, für dessen Bestimmung ausgehend von einer Gleichung für Batch-Lernen in [Ste07, S. 183] auf die gleiche Weise wie in Abschnitt 3.3.3 eine Gleichung für adaptives Online-Lernen bestimmt wurde. Weiterführend wäre eine Erweiterung auf mehrere unabhängige Variablen sinnvoll, sofern dabei die Effizienz erhalten und adaptives Online-Lernen möglich bleibt. Regression eignet sich im Gegensatz zu Wahrscheinlichkeitstabellen kaum zur in Abschnitt 4.2.2 als notwendig erkannten Prognose ohne Rückgriff auf unabhängige Variablen, da sie im Kern nur den Wert einer Variablen prognostiziert, der ohne Werte unabhängiger Variablen bei gleichem Vorwissen immer gleich wäre.

Daher wird für die Prognose von Variablen mit metrischem Skalenniveau ohne unabhängige Variablen die *empirische Verteilungsfunktion* verwendet, die die unbedingte Verteilung der abhängigen Variablen nicht nur wie Regression parametrisch als Normalverteilung darstellt, sondern stattdessen aus gespeicherten gemessenen Werten der Variablen Wahrscheinlichkeiten bestimmt, indem sie die Anzahl von bestimmten Teilen der Werte durch die Gesamtanzahl der Werte teilt [Ste07, S. 20]. Die empirische Verteilungsfunktion weist also eine geringe Methodenunsicherheit auf, generalisiert aber leider

beim Lernen gar nicht, da sie nur Werte speichert. Eine geeignete Methode, die mehr als eine Normalverteilung und weniger als die empirische Verteilungsfunktion generalisiert und adaptives Online-Lernen unterstützt, hat sich leider nicht gefunden. Der mit der mangelnden Generalisierungsfähigkeit der empirischen Verteilungsfunktion beim Lernen verbundene Speicherbedarf hält sich jedoch in Grenzen, da sie nur zur Darstellung eindimensionaler Verteilungen genutzt wird, so dass eine moderate Grenze speicherbarer Variablenwerte festgelegt werden kann, ab der Werte gelöscht werden.

Spezialisierte Methoden

Die stark spezialisierten Methoden in obiger Auflistung sind eher als simpel einzustufen, führen überwiegend deterministische Folgerungsprognosen durch und erfordern kein Lernen. Der *Zeitinkrementierer* prognostiziert den Wert einer Variablen, die die Zeit als kontinuierlich wachsende Größe ausdrückt, in der Zeitscheibe $j+1$ aus ihrem Wert in der Zeitscheibe j, indem er zu diesem die Länge des Zeitraumes zwischen zwei Zeitscheiben addiert. Der *Zeitperiodisierer* prognostiziert aus einer solchen Zeitvariablen die Zeit als periodische Größe mit fester Periode (z.B. die Stunde eines Tages oder den Wochentag). Eine Variable mit der Zeit als periodische Größe trägt als unabhängige Variable möglichen Unterschieden von Zusammenhängen zu unterschiedlichen Zeitpunkten Rechnung und ermöglicht das Einbeziehen zeitlicher Regelmäßigkeiten bzw. periodischer Zusammenhänge mit fester Periode.

Mit dem Begriff des *Binning* werden in der Statistik und in anderen Bereichen unterschiedliche Konzepte bezeichnet. Hier ist mit Binning die Diskretisierung der Werte einer stetigen Variablen mit metrischem Skalenniveau zu Werten einer diskreten Variablen mit nominalem Skalenniveau gemeint. Dabei wird die Zuordnung von Intervallen zu Werten mit nominalem Skalenniveau im Prognosemodell festgelegt. Durch Binning kann ausgenutzt werden, dass Methoden, die ein bestimmtes Skalenniveau erfordern, grundsätzlich auch auf Daten mit höherem Skalenniveau anwendbar sind ↑. *2D-Binning* stellt eine erweiterte Form von Binning dar, die zwei unabhängige Variablen erlaubt. Sie bildet keine Intervalle im eindimensionalen Raum auf diskrete Werte ab, sondern Rechtecke im zweidimensionalen Raum. Dadurch wird es z.B. möglich, die Position des Nutzers und des Gerätes in Form von Koordinaten auf einen diskreten Ort abzubilden.

Die *Polynomextrapolation* stellt einen ersten, einfachen Versuch dar, Trends als Zusammenhangsart bei Variablen mit metrischem Skalenniveau zu unterstützen. Es wird der Wert einer Variablen X_3 im aufgefalteten Prognosenetz prognostiziert, der den Zeitpunkt 3 beschreibt, und es werden dazu die Werte der Variablen X_0, X_1 und X_2 als unabhängige Variablen verwendet. Dabei wird für diesen ersten Versuch vorausgesetzt, dass die Werte x_0, x_1 und x_2 keinen starken kurzzeitigen Schwankungen unterliegen. Bei einer Prognose werden dann die Koeffizienten a, b und c eines von der Zeit abhängigen Polynoms zweiten Grades $x_j = a \cdot j^2 + b \cdot j + c$ so bestimmt, dass die Punkte $(0, x_0)$, $(1, x_1)$ und $(2, x_2)$ auf dem Funktionsgraphen des Polynoms liegen. Der Wert x_3 wird dann durch Einsetzen von 3 als Wert für j in $a \cdot j^2 + b \cdot j + c$ ausgerechnet. Die Nutzung von Trends bei Prognosen besitzt den besonderen Vorteil, dass so auch Situationen bewältigt werden, für die kein Vorwissen auf der Instanzebene gesammelt werden konnte, da kein Vorwissen auf der

Instanzebene benötigt wird. Die Polynomextrapolation weist allerdings den Nachteil auf, dass der prognostizierte Wert bei Prognosen in die fernere Zukunft häufig sehr schnell gegen ∞ strebt.

Die Methoden für die Berechnung des Durchschnitts und für einen Mehrheitsentscheid dienen der Ergebnisintegration beim Coprocessing. Die probabilistische Mischung als alternative Möglichkeit der Ergebnisintegration eignet sich eher nicht für das Verfahren der strukturierten Kontextdatenprognose, da sie Wahrscheinlichkeiten als Eingabe erwartet. Die Implementierung der datenabhängigen Auswahl bietet sich ebenso wie die Umsetzung einer Gewichtung beim Durchschnitt und dem Mehrheitsentscheid als weiterführende Aufgabe an.

4.3. Erstellung von Prognosemodellen

Das Erstellen eines Prognosemodells findet zur Entwicklungszeit durch die Anwendungsentwickler statt. Das Prognosemodell enthält im Gegensatz zum Vorwissen auf der Instanzebene keine Daten, die zur Laufzeit entstehen, also insbesondere kein zur Laufzeit erlerntes Vorwissen. Innerhalb des Prognosemodells ist zwischen Koordinationsvorwissen und Methodenvorwissen zu unterscheiden (vgl. Abschnitt 4.2.1). Das Koordinationsvorwissen ist gegeben durch das Prognosenetz, das die Prognosestruktur ausdrückt.

Dieser Abschnitt gliedert sich in zwei Teile. Der erste zeigt ein systematisches Vorgehen zur Erstellung von Prognosemodellen auf. Der zweite geht auf Netzmuster ein, die sich für bestimmte Zwecke in Prognosenetzen eignen.

4.3.1. Systematisches Vorgehen

In Abschnitt 4.1.3 wurde bereits ein grundlegender Ablauf zur Erstellung von Prognosemodellen anhand eines Beispiels erläutert. In Abschnitt 3.4.2 wurde die Erstellung bayes'scher Netze erörtert. Die Vorgehensweisen in diesen beiden Abschnitten sind in Abbildung 4.8 zu einem konkreten, umfassenderen Vorgehen zusammengefasst, das im Folgenden erläutert wird.

Wie allgemein bei der Erstellung von Modellen müssen die Anwendungsentwickler Wissen über die Anwendungsdomäne besitzen oder gewinnen. Im Fall von Prognosemodellen müssen zunächst Variablen identifiziert werden. Bei der Wahl von Variablen, die im Prognosemodell verwendet werden sollen, ist zu berücksichtigen, dass ihre Werte messbar oder anderweitig (z.B. über andere Geräte) ermittelbar sein müssen, um Trainingsdaten für adaptives Online-Lernen bilden zu können. Unterstützend können Ansätze des Data Mining im Bereich der Vorverarbeitung wie z.B. Ansätze der Dimensionsreduktion für die Wahl relevanter Variablen verwendet werden.

Es empfiehlt sich dann zu ermitteln, zwischen welchen Variablen Zusammenhänge bestehen, also die Zusammenhangsstruktur zu ermitteln (vgl. Abschnitt 3.4.2). Dadurch bringt man in Erfahrung, welche Zusammenhänge bestehen, die zu berücksichtigen sind, um möglichst realitätsgetreue Prognosen zu erhalten. Für die Ermittlung der Zusammenhangsstruktur bestehen wie bei bayes'schen Netzen die Optionen, Expertenwissen über die Kausalitäten in der Anwendungsdomäne zu nutzen oder die Struktur mittels

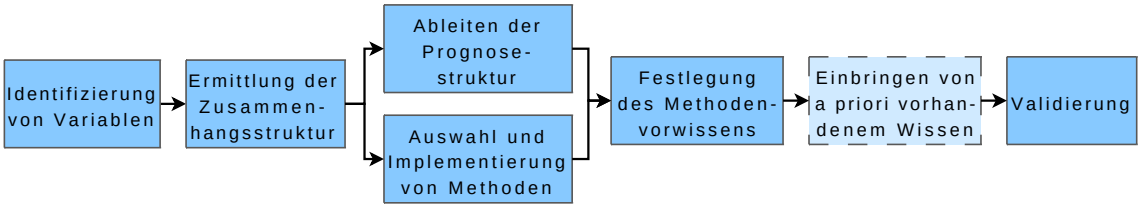

Abbildung 4.8.: grundlegendes Vorgehen zur Erstellung von Prognosemodellen

spezieller Werkzeuge automatisch lernen zu lassen. Bei einer Nutzung bestehenden Expertenwissens erübrigt sich eine explizite Ermittlung der Zusammenhangsstruktur. Zur automatischen Ermittlung kommen existierende Ansätze zum automatischen Lernen der Struktur bayes'scher Netze in Frage. Die Struktur wird dabei aus Kontextdaten gelernt, die von potentiellen Nutzern stammen können. Wenn sich die Zusammenhangsstruktur zwischen unterschiedlichen Nutzern unterscheidet, kann als Gesamtzusammenhangsstruktur die Kombination der Zusammenhänge der unterschiedlichen potentiellen Nutzer verwendet werden. Dies kann allerdings zu einem erhöhten Ressourcenaufwand führen, wenn die Zusammenhänge alle bei Prognosen berücksichtigt werden sollen.

Wenn Wissen über die Anwendungsdomäne in Form der Zusammenhangsstruktur in Erfahrung gebracht ist, muss die Prognosestruktur festgelegt werden, die durch ein Prognosenetz repräsentiert wird und damit den wichtigsten Teil des Prognosemodells darstellt. Der Übergang von der Zusammenhangsstruktur zur Prognosestruktur stellt den Schritt von der Realität zum Modell dar, das der Ausführung der Prognosen relativ nah steht. Das Prognosemodell beeinflusst z.B. die Effizienz von Prognosen. Die Zusammenhangsstruktur kann in vielen Fällen mit kleinen bis mittleren Änderungen als Prognosestruktur übernommen werden. Wenn man etwa in Beispiel 1.2 den kausalen Zusammenhang in das Prognosenetz übernimmt, dass die Telefonieraktivität kausal von der Anzahl verpasster Anrufe abhängt, führt im Prognosenetz eine Kante von der Variablen, die die Anzahl verpasster Anrufe angibt, zu der Variablen, die die Telefonieraktivität ausdrückt. Inwieweit Änderungen beim Übernehmen der Zusammenhangsstruktur nötig sind, hängt zum einen davon ab, wie großer Ressourcenaufwand durch die Prognosestruktur entsteht. Zum anderen kommt es darauf an, über welche Variablen Prognosen möglich sein sollen, denn davon hängt es ab, ob mit der Richtung der Kausalitäten oder entgegen der Richtung der Kausalitäten prognostiziert werden muss. Wenn bei der Analyse des Anwendungsbereichs von den Kausalitäten ausgegangen wurde und entgegen der Kausalitäten prognostiziert werden soll, muss das Netz grundlegend geändert werden, da nur Prognosen entlang der Kanten des Prognosenetzes unterstützt werden. Abschnitt 3.4.2 beschäftigt sich genauer mit der Überführung der Zusammenhangsstruktur zur Prognosestruktur bei bayes'schen Netzen. Die Überlegungen sind auf Prognosenetze übertragbar, da aufgefaltete Prognosenetze als bayes'sche Netze aufgefasst werden können.

Parallel zur Formulierung der Prognosestruktur sollte die Auswahl und bei Bedarf auch Implementierung von Methoden für die einzelnen Variablen erfolgen, denn erst, wenn die Prognosestruktur und die Methoden bekannt sind, kann eingeschätzt werden,

wie hoch die Effizienz ist, also wie viel Speicherplatz das Vorwissen auf der Instanzebene belegt und wie schnell die Prognosen sind. Solche Einschätzungen können wiederum für Anpassungen des Prognosemodells genutzt werden. Wenn man z.B. eine Wahrscheinlichkeitstabelle verwenden möchte und die Anzahl der unabhängigen Variablen groß ist, kann es Sinn machen, statt einer mehrere Wahrscheinlichkeitstabellen parallel im Coprocessing einzusetzen, die alle jeweils den gleichen Wert zu prognostizieren versuchen, jedoch jeweils nur einen Teil der unabhängigen Variablen benutzen. Das könnte z.B. bei dem in Abbildung 4.3 dargestellten Prognosenetz für die Prognose des Telefonierstatus Sinn machen. Bei der Wahl einer Methode für eine Variable sind außerdem Faktoren wie die Skalenniveaus der unabhängigen Variablen und der abhängigen Variablen sowie die Zusammenhangsart zu bedenken. Nicht jede Methode ist für jede Variable geeignet im Prognosenetz. Insgesamt spielen bei der Auswahl und dem Einsatz von Methoden die folgenden Dimensionen eine Rolle (vgl. auch Abschnitt 3.3.4).

- Zusammenhangsart (z.B. sequentiell, periodisch, linear, ...)
- Zukunfts- oder Folgerungsprognose
- adaptives Online-Lernen mit Vorwissen auf der Instanzebene oder a priori vorhandenes Vorwissen
- Skalenniveau der Variablen (nominal, ordinal oder metrisch)
- Coprocessing oder keine parallele Hybridität
- Unterstützung von Wahrscheinlichkeiten
- Nutzung bestehender oder Implementierung neuer Methoden

Als relevante Grundlagen für die Auswahl von Methoden wurden bereits die Charakteristika von Wahrscheinlichkeitstabellen und einfacher linearer Regression für den Fall der Nutzung von Methoden der Referenzkonfiguration und die Prinzipien des Chain- und Coprocessing erörtert. Auf der Seite der Anforderungen an die Charakteristika einer Methode spielt die Zusammenhangsart eine besondere Rolle, da sie unter Umständen erst ermittelt werden muss. Dafür können Ansätze der deskriptiven Statistik verwendet werden. So kann z.B. mit Kontextdaten potentieller Nutzer geprüft werden, ob in Beispiel 1.2 zwischen der Anzahl verpasster Anrufe und der Telefonierdauer bei den meisten Nutzern ein linearer Zusammenhang besteht, so dass lineare Regression sich als Methode anbietet. Wenn festgestellt wird, dass die Zusammenhangsarten sich von Nutzer zu Nutzer unterscheiden, kann eine Methode gewählt werden, die möglichst viele der Zusammenhangsarten unterstützt. Für Analysen wie die Ermittlung der Zusammenhangsart bietet sich auch der Einsatz entsprechender Werkzeuge an, damit der Vorbereitungsaufwand für die Anwendungsentwickler nicht zu groß wird. Für das Verfahren der strukturierten Kontextdatenprognose sind keine speziellen Werkzeuge vorgesehen, aber es sind allgemeine Werkzeuge verfügbar, die z.B. Ansätze des Data Mining oder der Statistik implementieren.

Wenn die Methoden gewählt sind, muss noch das Methodenvorwissen wie z.B. die Größe einer Wahrscheinlichkeitstabelle festgelegt werden. Wenn schließlich alles bis ins Detail konfiguriert ist, kann optional noch das Einbringen von a priori vorhandenem

Wissen erfolgen. Bei a priori vorhandenem Wissen handelt es sich nach Abschnitt 2.5.4 um Vorwissen, das zur Laufzeit des Prognosesystems auf der Instanzebene existiert, jedoch schon zur Entwicklungszeit eingebracht wird. A priori vorhandenes Wissen kann Zusammenhänge betreffen, die bei allen Nutzern gleich sind. Grundsätzlich kann a priori vorhandenes Wissen entweder von Experten stammen und direkt eingebracht werden oder durch Lernen von Kontextdaten potentieller Nutzer gewonnen werden ↑. Die zweite Möglichkeit kann genutzt werden, indem die Entwickler das Prognosesystem zur Entwicklungszeit der Anwendung das Prognosesystem starten, die Kontextdaten lernen, das gewonnene Vorwissen speichern lassen und die Anwendung und das Prognosesystem schließlich mit diesem Vorwissen ausliefern. Das gewonnene Vorwissen ist dann allerdings nicht im Prognosemodell gespeichert, sondern separat als Instanzdaten (vgl. Abbildung 4.1). Inwieweit die erste Möglichkeit des Einbringens von Expertenwissen unterstützt wird, hängt von der Implementierung des Prognosesystems ab. Möglich ist aber in jedem Fall, dass insbesondere Methoden, die kein Lernen unterstützen, das Einbringen von Wissen als Methodenvorwissen im Prognosemodell erlauben. Das ist z.B. bei Binning der Fall, das eine Diskretisierung von Werten durchführt und Angaben dazu benötigt, welche Intervalle zu diskreten Werten zusammengefasst werden sollen. Solches Vorwissen steht a priori vorhandenem Wissen nah.

Als letzter Schritt der Erstellung eines Prognosemodells erfolgt dessen Validierung. Dabei wird geprüft, ob das Prognosemodell sich für die Anwendung eignet. Als Kriterien dafür kommt unter Anderem die Auswahl methodenspezifischer Anforderungen aus dem Anforderungskatalog, die in Abschnitt 3.3.4 zu finden ist, in Frage. Wenn das Prognosemodell nicht die Erwartungen der Anwendungsentwickler erfüllt, muss zu einem früheren Schritt des Vorgehens der Modellerstellung zurückgekehrt werden. Abschnitt 5.3.1 gibt ein umfassendes, konkretes Beispiel zur Erstellung eines Prognosemodells.

4.3.2. Netzmuster

Das Prognosenetz ist der wichtigste Bestandteil eines Prognosemodells. Möglichkeiten der Erstellung von Prognosenetzen sind der Einsatz von Expertenwissen und die manuelle Konstruktion des Prognosenetzes. Selbst wenn die Zusammenhangsstruktur automatisch ermittelt wird, entscheidet sich schon viel durch die Entscheidungen von Menschen bei der Wahl der Variablen. Durch die graphische Struktur von Prognosenetzen können Anwendungsentwickler bzw. Anwendungsexperten relativ intuitiv modellieren. Dennoch besteht eine Lücke zwischen Variablen und Kanten als Bestandteile eines Prognosenetzes auf der einen Seite und anwendungsnahen Konzepten auf der anderen Seite. Daher werden in diesem Abschnitt *Netzmuster* betrachtet. Dabei handelt es sich um ganze Netzfragmente, die für bestimmte Zwecke in Prognosenetzen eingesetzt werden können. Weiterführend könnte die Definition von Prognosenetzen in der Weise um zusätzliche Netzbestandteile angereichert werden, dass damit solche Muster leichter und kompakter dargestellt werden können. Die Muster machen auch deutlich, wie stark die Ausdrucksfähigkeit von Prognosenetzen ist.

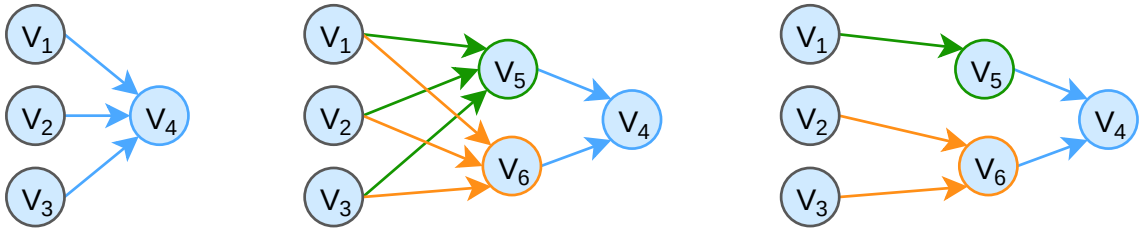

Abbildung 4.9.: Netzfragment ohne Coprocessing (links) und in zwei Varianten mit Coprocessing

Coprocessing

Abbildung 4.9 zeigt ein Beispiel für das in Abschnitt 3.4.1 eingeführte Konzept des Coprocessing, ausgedrückt mit Prognosenetzen. Die Methoden von V_5 und V_6 prognostizieren jeweils den Wert von V_4 und die Methode von V_4 führt die Ergebnisintegration durch. Bei der rechten Variante in der Abbildung als Spezialfall der mittleren berücksichtigen V_5 und V_6 jeweils nur einen Teil der ursprünglichen unabhängigen Variablen von V_4 im Netzfragment ohne Coprocessing. V_5 könnte z.B. den Ort in Abhängigkeit von der Zeit und V_6 den Ort in Abhängigkeit von den letzten beiden Aktivitäten des Nutzers prognostizieren. Durch eine solche Aufteilung der unabhängigen Variablen können Ressourcen geschont werden (z.B. Speicherplatz bei Wahrscheinlichkeitstabellen).

Ereignisse

Bisher wurden Variablen überwiegend zur Repräsentation von Zuständen verwendet. Das nächste Netzmuster in Abbildung 4.10 links dient der Darstellung von Ereignissen im Sinne von Zustandsübergängen als eine alternative Sichtweise, die in manchen Anwendungsbereichen geeigneter sein kann. V_1, V_2 und V_3 in der Abbildung drücken Zustände aus und V_4, V_5 und V_6 repräsentieren jeweils ein Ereignis. Diese Arten von Variablen werden im Folgenden *Zustandsvariablen* und *Ereignisvariablen* genannt. Ein Ereignis kann entweder eintreten oder nicht eintreten. Daher sind Ereignisvariablen binäre Variablen, die genau dann den Wert 1 annehmen, wenn das jeweilige Ereignis zwischen der vorhergehenden und der aktuellen Zeitscheibe eingetreten ist. Die Methode von V_4 in Abbildung 4.10 überprüft, wie sich die Werte von V_1 und V_2 jeweils gegenüber der vorhergehenden Zeitscheibe geändert haben. Die Eingangskanten von V_4 können so interpretiert werden, dass der Wert von V_4 durch die Zustandsübergänge von V_1 und V_2 definiert ist, oder in der Weise, dass das Ereignis sich von dem Ereignis, das durch die Zustandsübergänge definiert ist, unterscheidet, aber durch dieses ausgelöst wird. V_4 kann seinerseits die Ereignisse V_5 und V_6 auslösen. Ereignisse können Zustände beeinflussen (z.B. V_3) oder selbst die gesuchten Informationen einer Prognose bilden.

Ereignisvariablen, die sowohl über ihre Ein- als auch ihre Ausgangskanten mit Zustandsvariablen verbunden sind, können mit Transitionen in Petrinetzen verglichen werden, wobei die Eingangskanten im Prognosenetz die Funktion der Eingangskanten im Petrinetz als Vorbedingung für das Schalten der Transition wahrnehmen. Das Entfernen und Hinzufügen von Marken erfolgt im Prognosenetz über die Ausgangskanten.

Abbildung 4.10.: Netzmuster für Ereignisse (links) und Netzmuster für Flüsse (rechts)

Ereignisvariablen erfordern entsprechende Methoden. Die Methode einer Ereignisvariable kann ohne Lernen und Messwerte auskommen, wenn das Ereignis durch die Zustände anderer Variablen definiert ist und nur ein Übergang zu einer anderen Sichtweise stattfindet. Für das Messen der Werte einer Ereignisvariablen, die ausschließlich mit anderen Ereignisvariablen verbunden ist, wird man vermutlich häufig die Zustände messen müssen, über die das Ereignis definiert ist, um so indirekt den zu messenden Variablenwert zu bestimmen.

Flüsse

Ein weiteres Netzmuster orientiert sich an den in Abschnitt 3.3.1 vorgestellten Stock and Flow Diagrams. In diesen spielen Flüsse zwischen Variablen eine Rolle, die man sich wie Behälter mit Wasser vorstellen kann. Es kann Wasser hinzu fließen und Wasser abfließen. Der rechte Teil von Abbildung 4.10 stellt einen Zufluss von V_1 nach V_2 und einen Abfluss von V_2 nach V_3 als Variablen mit metrischem Skalenniveau als Prognosenetzfragment dar. Der Wert von V_4 gibt die Menge an, die sinnbildlich in der Zeit von der vorhergehenden Zeitscheibe bis zur aktuellen über den Zufluss hinzu geflossen sein soll und kann sich durch Prognose aus anderen Variablen ergeben. Die Methode von V_1 muss dann $v_{1,j} = v_{1,j-1} - v_{4,j}$ als Variablenwert prognostizieren. Der Wert von V_5 gibt an, wie viel von V_2 über den Abfluss abgeflossen sein soll. Der zu prognostizierende Wert von V_3 ergibt sich damit analog zu v_1 als $v_{3,j} = v_{3,j-1} + v_{5,j}$. Der Wert von V_2 entsteht schließlich durch Überlagerung von Zu- und Abfluss: $v_{2,j} = v_{2,j-1} + v_{4,j} - v_{5,j}$. Es ist allerdings zu befürchten, dass V_4 und V_5 in vielen Fällen nicht direkt für adaptives Online-Lernen messbar sind. Solange V_1 und V_3 keine weiteren Eingangskanten besitzen, ist jedoch eine indirekte Messung über diese möglich. Gelöst ist das Problem ebenfalls, wenn V_4 und V_5 über Repräsentationsabbilder als Methoden prognostiziert werden.

Relationen

Werte von Variablen können als Darstellung von Attributwerten von Entitäten aufgefasst werden. Offen geblieben ist aber bisher die Behandlung von Strukturen und Relationen $R \subseteq E \times F$ zwischen Entitäten aus E und Entitäten aus F, für die an dieser Stelle Netzmuster entwickelt werden.[1]

[1]Die Anregung zum Aufgreifen dieses Themas stammt in erster Linie von Jan D.S. Wischweh von der Universität Hamburg, der die Bedeutung von Relationen hervorhebt, die auch in seiner noch nicht abge-

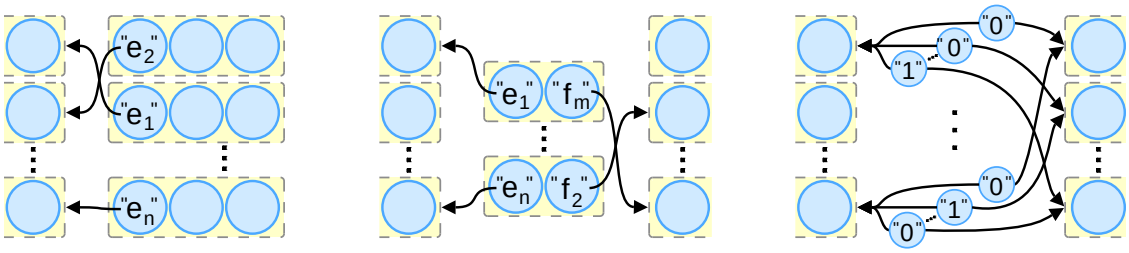

Abbildung 4.11.: Netzmuster für 1:n-Relationen (links) und zwei Netzmuster für n:m-Relationen (Mitte und rechts), gelb unterlegt: Variablen einer Entität, Pfeile: Referenzen zu Entitäten durch Variablenwerte als Fremdschlüsselwerte (links und Mitte) bzw. implizit durch Variablen (rechts)

1:n-Beziehungen sind relativ leicht ausdrückbar, wenn man bedenkt, dass man eine Menge von Variablen als Tupel, das eine Entität aus E repräsentiert, in einer Relation im relationalen Modell [KE06] interpretieren kann (vgl. Abbildung 4.11 links). Eine der Variablen besitzt dann als Wert den Identifikator der Entität aus F, mit der die Beziehung besteht.

Schwieriger erscheint die Umsetzung von n:m-Beziehungen, für die verschiedene Möglichkeiten existieren. Die erste Möglichkeit ergibt sich aus den Ansätzen von Mayrhofer und Sigg, neben den üblichen Skalenniveaus nominal, ordinal und metrisch auch nicht-atomare Datentypen wie z.B. Listen oder Mengen zu unterstützen [May04, S. 46] [Sig08, S. 54-55]. Mit einer Menge als Wert einer Variablen eines Tupels, das eine Entität aus E repräsentiert, kann eine Beziehung zu mehreren anderen Entitäten aus F ausgedrückt werden, so dass eine n:m-Beziehung beschrieben wird. Da aber die Menge aller Teilmengen der Entitäten in F mit $2^{|F|}$ sehr groß werden kann, könnte bei vielen Methoden ein zu hoher Ressourcenbedarf entstehen. Darüber hinaus sind durch die Asymmetrie der Modellierung Prognosen darüber, mit welchen Entitäten aus E eine Entität aus F in Beziehung steht, nur sehr umständlich möglich. Daher erscheint die erste Möglichkeit eher weniger gut geeignet.

Die zweite Möglichkeit verwendet ausgehend vom relationalen Modell, wie in relationalen Datenbanken üblich, eine eigene Relation für die n:m-Beziehung (vgl. Abbildung 4.11 Mitte). Da für das zur Laufzeit benötigte Hinzufügen oder Entfernen von Tupeln aus der Relation jedoch Variablen hinzugefügt oder entfernt werden müssen, macht diese Lösung nur Sinn, wenn das Prognosesystem Instanzen unterstützt und dabei außerdem Abfragen über die Existenz von Variablen erlaubt und selbstständig Instanzen erzeugen und zerstören kann, um das Bestehen von Relationen zwischen bestimmten Entitäten prognostizieren zu können.

Die dritte Möglichkeit stellt die auch bei der zweiten Möglichkeit verwendete Relation für die n:m-Beziehung so dar, dass für jedes mögliche Tupel der Relation eine binäre Variable eingeführt wird, die genau dann den Wert 1 annimmt, wenn das Tupel in der Relation ist, also die Beziehung zwischen den entsprechenden Entitäten besteht (vgl. Abbildung 4.11 rechts). Dafür werden $|E| \cdot |F|$ Variablen benötigt, von denen allerdings jede

schlossenen Diplomarbeit „Aktivitätsorienterte Kontextadaption für mobile Anwendungen“ eine Rolle spielen.

einzelne mit nur zwei möglichen Werten eher zu einem geringen Ressourcenaufwand führt.

Die dritte Möglichkeit weist gegenüber der zweiten den Vorteil auf, dass das Prognosesystem keine Instanzen unterstützen muss, solange die Mengen der Entitäten E und F zur Entwicklungszeit bekannt sind. Ansonsten müssen bei der dritten ebenso wie bei der zweiten Möglichkeit Instanzen unterstützt werden.

4.4. Erweiterungen

Das im vorhergehenden Abschnitt 4.2 vorgestellte Basissystem bildet ein funktionsfähiges Prognosesystem. Dennoch bleiben noch Wünsche offen. Die folgenden Erweiterungen bieten Ansätze zur Erfüllung der wichtigsten dieser Wünsche.

4.4.1. Verteilte Prognose

In Abschnitt 2.5.8 wurden als grundsätzliche zwei Möglichkeiten der Verteilung bereits die Nutzung der Prognoseergebnisse auf einem entfernten Gerät (*entfernte Nutzung*) sowie die Durchführung der Datenakquisition auf einem entfernten Gerät (*entfernte Datenakquisition*) genannt. Abbildung 4.12 veranschaulicht diese Möglichkeiten. Die in Abschnitt 4.1.2 gewählte Architektur bietet zusätzlich die ebenfalls nützlich erscheinende Möglichkeit des *Austausches von Vorwissen*, das dem Konzept des deduktiven Lernens entspricht (vgl. Abbildung 4.13).

Eine entfernte Datenakquisition bietet insbesondere den Vorteil, zusätzliche Sensoren wie z.B. GPS-Empfänger nutzen zu können (vgl. auch Abbildung 4.7). Ein besonderer Vorzug der entfernten Nutzung liegt darin, dass die Nutzung von Prognosen eines entfernten, möglicherweise stationären, leistungsfähigen Gerätes wie eines Smart Spaces den Zugang zu beispielsweise ortsbezogenen Prognosen ermöglicht. So könnte z.B. ein Navigationssystem in einem unbekannten Gebiet von einem stationären System Prognosen über den Verkehr anfordern. Der Austausch von Vorwissen bietet den Vorteil, dass Prognosen mit Vorwissen von einem entfernten Gerät auch in dessen Abwesenheit und ohne weitere Belastung von Netzwerken durchgeführt werden können. Damit eignet sich das Austauschen von Vorwissen z.B. auch gut bei nutzerspezifischen Prognosen wie der Prognose der Position eines anderen Nutzers in dessen Abwesenheit, um ein Zusammentreffen mit diesem planen zu können. Durch den Austausch von Vorwissen auf der Instanzebene, das sonst von den Geräten unterschiedlicher Nutzer jeweils separat gelernt würde, sich aber auf den gleichen Gegenstandsbereich (z.B. einen gemeinsamen Haushalt) bezieht, kann die hohe Schätzunsicherheit vermieden werden, wenn erst wenig Trainingsdaten gelernt sind. Insgesamt weisen alle drei Varianten verteilter Prognose unterschiedliche Charakteristika hinsichtlich Gütemaßen wie Netzlast und speziell in Ad-Hoc-Netzen hinsichtlich der Überwindung räumlicher Entfernung auf und spielen alle jeweils in bestimmten Szenarien ihre Stärken aus. Im Folgenden wird herausgearbeitet, wie die drei Varianten jeweils realisiert werden können.

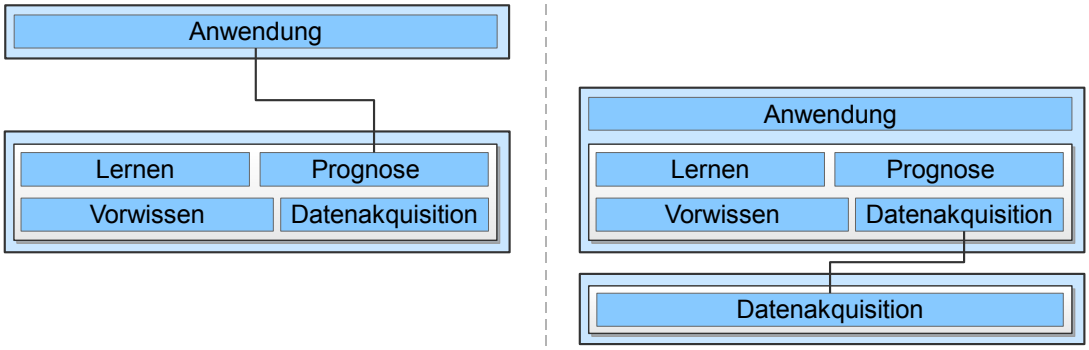

Abbildung 4.12.: entfernte Nutzung (links) und entfernte Datenakquisition (rechts) als Möglichkeiten der Verteilung, äußere Kästen stellen Geräte dar

Entfernte Datenakquisition

Eine entfernte Datenakquisition ist bei einfacher Ausgestaltung der Datenakquisitionsschicht durch Implementieren eines Adapters möglich, der den Wert einer Variablen von einem entfernten Gerät bezieht. Die Anwendungsentwickler müssen dafür die Schnittstelle der Datenakquisitionsschicht auf diesem Gerät nach außen verfügbar machen.

Zur effizienten Durchführung der Datenakquisition ergeben sich jedoch aus den folgenden Voraussetzungen zusätzliche, darüber hinaus gehende Anforderungen. Es werden in der Regel Kontextdaten benötigt, die zu bestimmten, regelmäßigen Zeitpunkten gemessen sind. Beim Lernen müssen bei der Bildung von Datensätzen die Zeitversätze der Kanten im Prognosenetz beachtet werden ↑. Beim Sammeln von gegebenen Informationen für Prognosen müssen die Zeitpunkte in ein Zeitraster passen, das dem gelernten Methodenvorwissen auf der Instanzebene entspricht. Wenn eine Methode z.B. einer Variablen X zugeordnet ist, die die Position angibt und eine Eingangskante $X \xrightarrow{1} X$ besitzt, dann ermittelt die Methode beim Lernen, wie die Position zu einem bestimmten Zeitpunkt von der Position zum Zeitpunkt davor abhängt. Wenn die Länge eines Zeitschrittes zwischen zwei Zeitpunkten als 15 Minuten festgelegt ist und diese Länge beim Lernen zugrunde gelegt wird, kann durch die Methode auch nur die Position für einen Zeitpunkt prognostiziert werden, wenn die Position 15 Minuten davor bekannt ist. Wegen dieser zeitlichen Regelmäßigkeiten macht es Sinn, dass Geräte, die Kontextdaten von einem entfernten Gerät beziehen möchten, ihren Bedarf an Daten einmalig beim entfernten Gerät inklusive Angaben zu den gewünschten Messzeitpunkten anmelden. Zu diesem Ansatz existieren mindestens die folgenden zwei Alternativen, die jedoch beide mit größeren Nachteilen behaftet sind. Eine Anforderung jedes einzelnen Messwertes durch synchrone Kommunikation verursacht eine unnötig hohe Netzlast. Ein Abholen mehrerer Messwerte nach dem Messen ohne vorherige Anmeldung erfordert, dass das entfernte Gerät permanent in geringen Zeitabständen Werte misst, die unnötig viel Speicherplatz belegen. Daher wird der Ansatz der vorherigen Anmeldung bevorzugt.

Bei diesem Ansatz ist festzulegen, was passiert, wenn ein angemeldetes Gerät ohne Ankündigung nicht mehr erreichbar ist (z.B. durch instabile Vernetzung oder Ausschalten). Potential für weitere Verbesserungen liegt vor, wenn mehrere Geräte von einem entfernten Gerät Kontextdaten beziehen und die Zeitschritte der Geräte die gleiche Län-

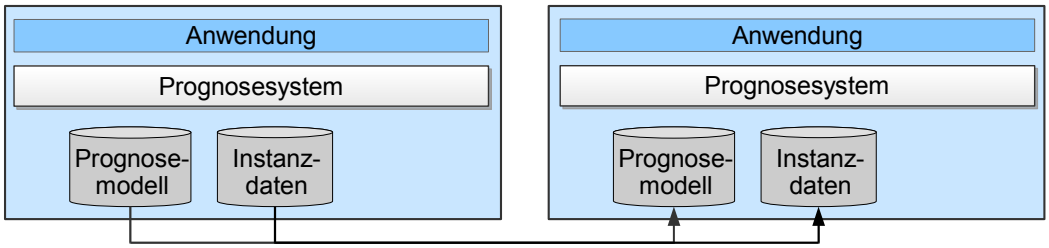

Abbildung 4.13.: Austausch von Vorwissen, äußere Kästen stellen Geräte dar

ge aufweisen. Dann kann eine zeitliche Abstimmung erfolgen, indem Geräte gebeten werden, ihre Zeitraster so zu verschieben, dass sich die Zeitraster der unterschiedlichen Geräte decken. Die Länge eines Zeitschrittes können Geräte jedoch nicht ohne Weiteres anpassen, da schon gelerntes Methodenvorwissen auf der Instanzebene sich auf die bestehende Länge bezieht (z.B. Position in Abhängigkeit von der Position 15 Minuten früher). Eine Abstimmung könnte aber schon bei der Erstellung von Prognosemodellen bei der Wahl der Zeitschrittlängen erfolgen, die möglichst Vielfache bzw. Teiler voneinander sein sollten (z.B. 15 statt 16 Minuten in Modell B bei 30 Minuten in Modell A). Dies könnte durch allgemeine Konventionen für die Erstellung von Prognosemodellen geregelt werden. Kleinere zeitliche Abweichungen zwischen tatsächlichen Messzeitpunkten und gewünschten Messzeitpunkten können durch Interpolation von Werten für die gewünschten Zeitpunkte kompensiert werden. Bei kürzerer Länge der Zeitschritte und sich schnell ändernden Messwerten ist zu berücksichtigen, dass Zeitangaben bei nicht synchronisierten Uhren unterschiedlicher Geräte nur bedingt aussagekräftig sind. Als weiterführende Möglichkeit ist vorstellbar, dass für ein angemeldetes Gerät in dessen Abwesenheit eine gewisse Menge (ortsspezifischer) Daten gepuffert wird, die es als Trainingsdaten zum Lernen verwenden möchte.

Austausch von Vorwissen

Eine Alternative zu einem solchen Puffern von Kontextdaten ist, dass das Daten beziehende Gerät, bevor es einen Ort verlässt, einem anderen Gerät sein Vorwissen auf der Instanzebene hinterlässt. Dieses Gerät kann dann nicht nur die Datenakquisition, sondern auch das Lernen stellvertretend übernehmen. Im Fall von Methoden, die beim Lernen stark von den Trainingsdaten abstrahieren, und bei längerer Lerndauer ergibt sich dadurch eine geringere Netzlast. Das Austauschen von Vorwissen kann sowohl zum gerade vorgestellten Übertragen der Lernaufgabe an andere Geräte, als auch zum Prognostizieren mit dem Vorwissen anderer Geräte genutzt werden. Der Austausch von Vorwissen wird durch die Architektur des Prognosesystems unterstützt, da das Vorwissen in der Vorwissenschicht gekapselt ist und externe Repräsentationen des Prognosemodells und des Vorwissens auf der Instanzebene existieren, die von der Anwendung als Datenstrom verschickt werden können ↑ ↑ (vgl. Abbildung 4.13).

Ein Prognosesystem, dem Vorwissen eines anderen Gerätes von der Anwendung übergeben wird, muss keine Kenntnis von der Bedeutung des Prognosemodells besitzen, um damit lernen oder prognostizieren zu können, da Syntax und Semantik des Prognosemo-

dells durch das Prognosemetamodell festgelegt sind ↑. Wenn die Anwendung Prognosen nutzen möchte, muss sie allerdings Wissen über die Bedeutung der zu prognostizierenden Variablen besitzen. Außerdem müssen auf dem Gerät die im Modell verwendeten Messadapter und Methoden verfügbar sein. Da diese stark anwendungsspezifisch sein können, könnte weiterführend geprüft werden, ob der Programmcode von Messadaptern und Methoden zusammen mit ausgetauschtem Vorwissen zu einem anderen Gerät übertragen werden kann. Bei Messadaptern für physikalische Sensoren hilft dies allerdings nur, wenn das Gerät über diese Sensoren verfügt, und die Messadapter müssen wegen der Heterogenität mobiler Geräte in manchen Fällen wie bei Mayrhofer unterschiedliche Geräteplattformen unterstützen [May04, S. 81]. Das Fehlen eines geeigneten Messadapters kann jedoch vom Prognosesystem toleriert werden, da Prognosen auch bei nicht vorhandenen Messwerten möglich sind ↑. Von Bedeutung ist außerdem, dass Messadapter, die auf den Kontext des jeweiligen Gerätes bezogen sind, nicht benutzt werden dürfen. Die per GPS gemessene Position bezieht sich z.B. auf das jeweilige Gerät. Wenn man mit dem Vorwissen eines anderen Gerätes Prognosen über dieses erstellen möchte, wird man aus diesem Grund oftmals nur die Zeit als gegebene Informationen nutzen können.

Entfernte Nutzung

Eine entfernte Nutzung von Prognosen ist prinzipiell möglich, wenn die Anwendungsentwickler die Schnittstelle zur Anforderung von Prognosen nach außen verfügbar machen und die Anwendung auf einem entfernten Gerät Prognosen in Anspruch nimmt. Als Erweiterung kann umgesetzt werden, dass sowohl Daten aus der kurzen Historie des Prognosesystems, als auch Daten vom entfernten, Prognosen nutzenden Gerät kombiniert als gegebene Informationen verwendet werden können. Eine Unterstützung des Publish-Subscribe-Paradigmas kommt ebenfalls in Frage. Ein großer weiterer Schritt ist, dass ein Prognosesystem selbstständig Prognosen von entfernten Geräten einholt und aufbauend auf den prognostizierten Werten selbst Prognosen durchführt. Zusätzlich kann es den entfernten Geräten selbst prognostizierte Werte zur Verfügung stellen, um dann letztendlich zu erreichen, dass ein Prognosenetz über mehrere Geräte verteilt werden kann. Ansatzweise ist dies auch möglich, indem ein Messadapter implementiert wird, der von entfernten Geräten Prognosen bezieht und diese wie gemessene Werte weiterreicht.

4.4.2. Netzfragmentinstanzen

Prognosenetze sind ein Abbild der Realität. Eine Variable in einem Prognosenetz repräsentiert z.B. ein Attribut einer Entität. Der Begriff der Entität wird in dieser Arbeit als abstraktes Konzept zur Bezeichnung einer Einheit in der Realität verwendet. Insbesondere die von Dey et al. übernommene Definition von Kontext (Definition 2.2) baut auf Entitäten auf. In Abschnitt 2.5.3 wird gefordert, dass ein Prognosesystem Instanzen von Entitäten unterstützt, um der Dynamik des Kontextes gerecht zu werden. Prognosen über Instanzen von Entitäten sind etwa in folgendem Beispiel sinnvoll.

Beispiel 4.1:
Eine Arbeitsgruppe von Mitarbeitern eines Unternehmens ist unterwegs und ihre mobilen Geräte sind über ein Ad-Hoc-Netz miteinander verbunden. Es kann dann vom Gerät einer der Mitglieder der Arbeitsgruppe über jedes andere Mitglied jeweils prognostiziert werden, wie sich dessen Position in nächster Zeit verändern wird. Basierend darauf kann das Gerät auch prognostizieren, ob die Arbeitsgruppe in nächster Zeit zusammen bleiben wird, und so entscheiden, ob es sich noch lohnt, eine Synchronisation gemeinsamer Daten der Arbeitsgruppe wie z.B. Textdokumenten und Tabellen zwischen den Geräten der Mitglieder zu starten.

Die Entität ist in diesem Fall der Mensch und die Instanzen der Entität sind die konkreten Mitglieder der Arbeitsgruppe. Mit dem Basissystem ist eine solche Prognose wie in diesem Beispiel im Allgemeinen nicht möglich, da die Anzahl der Instanzen unbekannt ist und somit nicht im Prognosenetz berücksichtigt werden kann, sondern stattdessen zur Laufzeit berücksichtigt werden muss.

Konzeptioneller Ausgangspunkt

Instanzen von Entitäten müssen im Prognosesystem als Teil des Vorwissens dargestellt werden, da sie den Kontext beschreiben, über den Prognosen benötigt werden. Wenn im Allgemeinen die Rede von Instanzen ist, wird gewöhnlich zwischen Typ- und Instanzebene unterschieden. Diese Unterscheidung entspricht in dieser Arbeit der Trennung von Prognosemodell und Vorwissen auf der Instanzebene (vgl. Abschnitt 4.2.1). Im Basissystem sind den einzelnen Variablen als Teil des Prognosenetzes jeweils Methoden und das Methodenvorwissen zugeordnet. Das Prognosenetz und das Methodenvorwissen bilden das Prognosemodell. Das Vorwissen auf der Instanzebene ist davon zu trennen. Beim Vorwissen auf der Instanzebene handelt es sich im Basissystem um das Vorwissen, das zur Laufzeit entsteht ↑, also das von den Methoden gelernte Vorwissen. Dies ist gut in Abbildung 4.2 zu erkennen. Die Instanzebene enthält also bisher keine Instanzen im Sinne von Exemplaren. Instanzen im Sinne von Exemplaren sind aber zur Unterstützung von Instanzen von Entitäten nötig.

Um Instanzen von Entitäten unterstützen zu können, müssen im Prognosesystem Instanzen von dem unterstützt werden, was die Entitäten repräsentiert. Attribute von Entitäten werden im Prognosenetz durch Variablen dargestellt. Instanzen von Variablen reichen jedoch nicht aus, da Entitäten des Kontextes durch mehrere Variablen beschrieben sein können. Daher müssen Instanzen von ganzen Netzfragmenten unterstützt werden. Instanzen von Netzfragmenten im Prognosesystem entsprechen dann Instanzen von Entitäten in der Realität. Innerhalb einer Netzfragmentinstanz kann dann eine Prognose über die Instanz der entsprechenden Entität durchgeführt werden. Instanzen von Netzfragmenten im Sinne von Exemplaren bilden Vorwissen auf der Instanzebene, das nicht zum Methodenvorwissen, sondern zum Koordinationsvorwissen zählt. Die Existenz von Koordinationsvorwissen auf der Instanzebene ist eine Neuerung gegenüber dem Basissystem, wie man anhand von Abbildung 4.2 erkennt. Es befindet sich dann also im

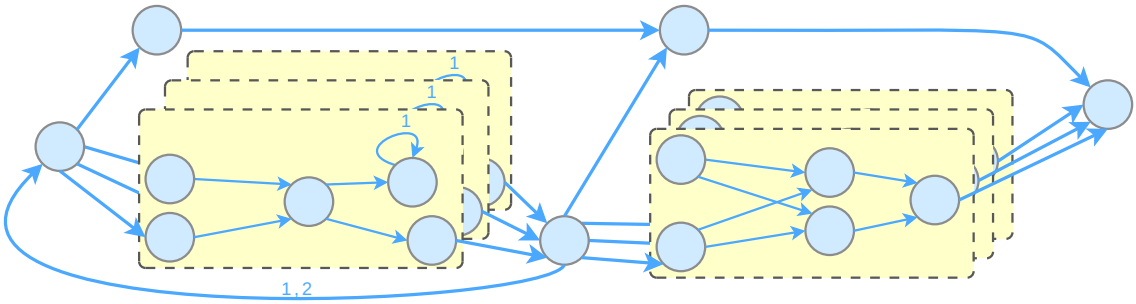

Abbildung 4.14.: abstraktes Beispiel für ein statisches Netz mit eingebetteten Instanzen

Prognosemodell das Prognosenetz mit dem Methodenvorwissen der Methoden, die den einzelnen Variablen zugeordnet sind. Ein Netzfragment des Prognosenetzes könnte entsprechend Beispiel 4.1 die Prognose der Position eines Menschen erlauben. Auf der Instanzebene befinden sich dann für jeden Menschen der Arbeitsgruppe eine Instanz des Netzfragments als Koordinationsvorwissen und das Methodenvorwissen für die einzelnen Variableninstanzen als Bestandteile der Netzfragmentinstanzen. Jeder Variablen ist in allen Netzfragmentinstanzen potentiell unterschiedliches Methodenvorwissen zugeordnet, um den unterschiedlichen Zusammenhängen, die für die einzelnen Instanzen von Entitäten gelten, Rechnung zu tragen.

Kontext, in dem sich die Entitäten befinden, wird allgemein auch synonym als Situation oder als Umgebung bezeichnet [↑]. Man kann sich daher Kontext als abstraktes Konzept vorstellen, in das alle Entitäten eingebettet sind. Aus dem Umstand, dass der Kontext alle relevanten Instanzen von Entitäten verbindet, wird die Vereinfachung abgeleitet, dass alle Instanzen von Netzfragmenten an einer festen Position in ein statisches Netz eingebettet sind, das dem Kontext entspricht und wie dieser immer existiert, so dass keine Instanzen davon erzeugt werden müssen. Es wird vermutet, dass das dadurch ausgeschlossene direkte Verbinden von Instanzen in vielen Anwendungsfällen nicht von Bedeutung ist. Abbildung 4.14 zeigt ein Beispiel für diesen Ansatz.

Von Interesse sind nun Kanten, die die Grenze eines Netzfragments überschreiten. Aufgrund der Einbettung an einer festen Position in ein statisches Netz ist schon im Prognosemodell festgelegt, von wo diese Kanten kommen bzw. wohin sie führen. Daher können alle Knoten und Kanten eines Prognosenetzes wie zuvor in der Vorwissenschicht dargestellt werden, wobei jedes Netzfragment durch genau ein Exemplar an der entsprechenden Stelle mit den entsprechenden Knoten und Kanten zu repräsentieren ist. Im Prognosemodell zu Abbildung 4.14 ist also zu jedem der zwei Netzfragmente jeweils genau eine der drei Instanzen an der jeweiligen Stelle enthalten. Zusätzlich muss eine Kennzeichnung der Netzfragmente erfolgen, damit von ihnen Instanzen erzeugt werden können. Dazu bietet es sich für die nachfolgenden Schritte an, separat im Prognosemodell Bezeichner für Netzfragmente festzulegen und jede Variable V_i mit einer Zuordnung zu maximal einem Netzfragment F zu versehen (Notation: V_i^F).

Umsetzung in der Vorwissenschicht

Für das Prognosemodell ergeben sich also durch die Unterstützung von Netzfragmentinstanzen kaum Änderungen. Es sind jedoch weitere Änderungen am Prognosesystem nötig. Diese betreffen das Vorwissen auf der Instanzebene, also die Vorwissenschicht. Die auf jeden Fall zu repräsentierenden Daten auf der Instanzebene umfassen Bezeichner der aktuellen Instanzen $f \in I(F)$ der Netzfragmente F und das Methodenvorwissen auf der Instanzebene für die einzelnen Variableninstanzen, die Teil der Netzfragmentinstanzen sind.

Es stellt sich die Frage, wie das gegenüber dem Basissystem neue Koordinationsvorwissen auf der Instanzebene, wie es in Abbildung 4.14 dargestellt ist, gehandhabt werden soll. Eine explizite Repräsentation des Koordinationsvorwissens auf der Instanzebene in der Vorwissenschicht einschließlich einer Repräsentation von Variableninstanzen macht grundsätzlich Sinn. Eine solche explizite Repräsentation ermöglicht höheren Schichten des Prognosesystems einen relativ transparenten Zugriff auf Variableninstanzen, der in etwa dem Zugriff auf Variablen im Basissystem entspricht. So wird vermieden, dass höhere Schichten selbst eine Repräsentation der Variableninstanzen erzeugen müssen und dass das Erzeugen einer solchen Repräsentation für die Lernschicht und die Prognoseschicht jeweils separat und damit insgesamt doppelt implementiert werden muss. Bei der Erzeugung einer Netzfragmentinstanz wird jedoch nicht der Weg gegangen, alle Variablen und Kanten, also die komplette Netzstruktur des Netzfragments im Prognosemodell, zur Repräsentation der Netzfragmentinstanz zu kopieren. Ein solches Vorgehen würde zu dauerhaft bestehenden, starken Redundanzen führen und einen relativ hohen Verwaltungsaufwand beim Erzeugen und Zerstören von Instanzen nach sich ziehen, da Kanten gezogen und entfernt werden müssen, die die Grenze des Netzfragments überschreiten. Aus diesen Gründen erscheint es am sinnvollsten, eine explizite Repräsentation von Koordinationsvorwissen auf der Instanzebene leichtgewichtig zu gestalten, indem sie möglichst alle Informationen, die sie bereitstellt, aus den oben aufgezählten, auf jeden Fall zu speichernden Informationen ableitet. Es wird dann höheren Schichten des Prognosesystems nur eine Sicht vermittelt, die nicht physikalisch existiert. Da das Koordinationsvorwissen auf der Instanzebene eine Sicht auf anderes Vorwissen vermittelt und damit auf dieses zugreift, bietet es sich an, das Koordinationsvorwissen auf der Instanzebene in der Architektur innerhalb der Vorwissenschicht oberhalb vom Prognosemodell anzusiedeln.

Wesentliches Element einer leichtgewichtigen Repräsentation von Koordinationsvorwissen auf der Instanzebene sind Variableninstanzen. Für diese ist zu berücksichtigen, dass eine Netzfragmentinstanz jederzeit zerstört werden kann, obwohl höhere Schichten des Prognosesystems möglicherweise noch Zugriff auf die Repräsentationen der Variableninstanzen besitzen. Die Repräsentationen der Variableninstanzen müssen daher bei der Zerstörung der Netzfragmentinstanz invalidiert werden. Da alle Repräsentationen von Variableninstanzen der Netzfragmentinstanz nur eine Sicht auf darunter liegende Daten vermitteln, kann das Invalidieren atomar durch eine einzelne Operation auf diesen darunter liegenden Daten erfolgen, so dass mögliche Synchronisationsprobleme auf elegante Weise umgangen werden. Die Repräsentationen der Variableninstanzen stellen

nach dem Invalidieren bei ihrer nächsten auszuführenden Operation fest, dass sie nicht mehr gültig sind.

Eingangskanten von Variableninstanzen

Zu den von der leichtgewichtigen Repräsentation anzubietenden Daten zählen die Eingangskanten einer Variableninstanz, die sich mit geringem Aufwand aus dem Prognosenetz ermitteln lassen. Dazu werden Regeln für verschiedene Fälle aufgestellt, die dem in Abbildung 4.14 dargestellten Konzept von Netzfragmentinstanzen in einem statischen Netz folgen. Dabei werden Variablen im Prognosenetz mit Zuordnung zu einem Netzfragment F mit V_i^F und solche ohne eine Zuordnung mit V_i bezeichnet. Für Variableninstanzen als Teil einer Netzfragmentinstanz $f \in I(F)$ eines Netzfragmentes F wird die Notation $\hat{V}_i^f$ und für solche ohne eine Zuordnung die Notation $\hat{V}_i$ verwendet.

Es sind nun für die Bestimmung der Eingangskanten einer Variableninstanz die Eingangskanten der zugehörigen Variablen im Prognosenetz zu betrachten. Aus einer Kante $V_i^F \overset{\Delta}{\rightarrow} V_{i'}$ im Prognosenetz (Fall „$V_i^F \overset{\Delta}{\rightarrow} V_{i'}$“) ergeben sich für die Variableninstanzen die Kanten $\bigcup_{f \in I(F)} \hat{V}_i^f \overset{\Delta}{\rightarrow} \hat{V}_{i'}$. Es geht also eine Kante von jeder Variableninstanz aus. Dieser Fall entspricht in Beispiel 4.1 über eine Arbeitsgruppe mit mobilen Geräten der Prognose, ob die Arbeitsgruppe zusammen bleibt, aus den Prognosen über die Position der einzelnen Personen. In Abbildung 4.14 liegt dieser Fall bei der Variablen ganz rechts und der Variablen in der Mitte unten vor. Im Fall „$V_i^F \overset{\Delta}{\rightarrow} V_{i'}^F$“, also einer Kante innerhalb eines Netzfragmentes, liegt die eine Kante $\hat{V}_i^f \overset{\Delta}{\rightarrow} \hat{V}_{i'}^f$ auf der Instanzebene vor, die in der gleichen Instanz des Netzfragmentes bleibt. Wenn der eigentlich ausgeschlossene Fall „$V_i^F \overset{\Delta}{\rightarrow} V_{i'}^G$“ mit $F \neq G$ eintritt, bei dem zwei Netzfragmente direkt miteinander verbunden sind, kann wie im Fall „$V_i^F \overset{\Delta}{\rightarrow} V_{i'}$“ vorgegangen werden, um einen Abbruch zu verhindern. Die offene Frage danach, mit welcher Instanz des Netzfragments F die Kante(n) verbunden sein soll(en), wird dann so beantwortet, dass zu jeder Instanz eine Verbindung durch eine Kante hergestellt wird. Im Fall „$V_i \overset{\Delta}{\rightarrow} V_{i'}^F$“ ergibt sich eine Kante $\hat{V}_i \overset{\Delta}{\rightarrow} \hat{V}_{i'}^f$ und im Fall „$V_i \overset{\Delta}{\rightarrow} V_{i'}$“ eine Kante $\hat{V}_i \overset{\Delta}{\rightarrow} \hat{V}_{i'}$. Eine vom statischen Teil des Prognosenetzes zu einer bestimmten Netzfragmentinstanz f führende Kante wird also direkt übernommen.

Implikationen für weitere Teile des Prognosesystems

Was die Methoden betrifft, entsteht für bestehende Methoden durch Instanzen keine Veränderung, da diese lokal und unabhängig von der Struktur des Netzes, unter Rückgriff auf ihr Methodenvorwissen arbeiten ↑. Es besteht jedoch zusätzlicher Bedarf an Methoden, die nicht an eine zur Laufzeit feste Anzahl unabhängiger Variablen gebunden sind, um so mit Eingangskanten umgehen zu können, die von einer fluktuierenden Anzahl von Variableninstanzen ausgehen (vgl. Fall „$V_i^F \overset{\Delta}{\rightarrow} V_{i'}$“).

Das Lernen muss statt für jede einzelne Variable für jede einzelne Variableninstanz durchgeführt werden. Beim Lernen für eine Variableninstanz als abhängige Variableninstanz müssen die Werte aller ihrer unabhängiger Variableninstanzen von der Datenakquisitionsschicht bezogen werden. Wenn eine neue Instanz erzeugt oder zerstört wird,

muss das Lernen für die Variableninstanzen gestartet bzw. gestoppt werden, soweit für die jeweiligen Variablen gelernt werden soll. Das Stoppen kann über das Invalidieren der Repräsentationen von Variableninstanzen erfolgen. Für das Starten kommt ein Benachrichtigungsmechanismus in Frage, durch den die Vorwissenschicht der Lernschicht Änderungen signalisiert (vgl. [GHJV95]). Für alle Variablen(instanzen) inklusive solcher, die keinem Netzfragment zugeordnet sind, muss auch berücksichtigt werden, ob neue unabhängige Variableninstanzen hinzukommen. Dies kann von der Lernschicht vor dem Bestimmen jedes Datensatzes überprüft werden.

Bei der Datenakquisition ist zu berücksichtigen, dass für unterschiedliche Instanzen auch unterschiedliche Werte gemessen werden müssen. Daher müssen mehrere Instanzen von Messadaptern erzeugt werden können, die dann entsprechend parametrisierbar sind. Im Ubiquitous Computing kann eine Messung über z.B. einen Gegenstand als eine Entität erfolgen, wenn der Gegenstand ein Smart Thing ist, das Informationen über sich mittels eines drahtlosen Ad-Hoc-Netzes mitteilt.

Erzeugung und Zerstörung von Instanzen

Das Leben einer Instanz beginnt mit ihrer Erzeugung und endet mit ihrer Zerstörung. Eine Zerstörung im Prognosesystem sollte zum Freigeben von Ressourcen erfolgen, z.B. wenn sie den Kontext verlässt, ist aber nicht gleichzusetzen mit einer Zerstörung in der realen Welt. Daher muss entschieden werden, wann Instanzen im System erzeugt und zerstört werden. Dies kann erstens dadurch geschehen, dass die Anwendung eine Erzeugung oder Zerstörung anweist, was die Anwendung bzw. ein Kontextsystem aber mit der Beobachtung des Kontextes belastet. Zweitens kann das Prognosesystem dies selbstständig entsprechend anwendungsspezifischer Regeln übernehmen. Für das Zerstören einer Instanz (z.B. eines mobilen Gerätes im Kontext) bietet sich die Realisierung einer solchen Regel als Trigger an, der durch einen bestimmten Wert einer Variablen ausgelöst wird (z.B. keine Netzverbindung zum Gerät vorhanden). Dieser Ansatz ermöglicht auch Prognosen über die Zerstörung von Instanzen. Allerdings wird die Zerstörung eines Netzfragments bei einer Prognose dadurch noch nicht insofern berücksichtigt, dass das Netzfragment aus dem benutzten Ausschnitt des aufgefalteten Prognosenetzes entfernt wird. Weiterhin ist der Bedarf der Anwendung nach Prognosen über die Erzeugung von Instanzen und Abfragen über die Menge der aktuellen Instanzen zu berücksichtigen.

Mit dem Erzeugen und Zerstören von Instanzen ist die Verwaltung des Methodenvorwissens der Variablen auf der Instanzebene verbunden. Wenn zu erwarten ist, dass zerstörte Instanzen bzw. die Entitäten später und evtl. auch regelmäßig wieder Teil des Kontextes werden, ist es sinnvoll, das Vorwissen der Instanzen über deren Zerstörung hinaus zu speichern. In welchen Fällen Vorwissen auf der Instanzebene über die Zerstörung von Instanzen hinaus erhalten bleiben soll, stellt eine weitere anwendungsspezifische Entscheidung dar, die ebenfalls entweder direkt von der Anwendung oder durch das Prognosesystem entsprechend den Vorgaben der Anwendungsentwickler gefällt werden kann. In einfacheren Fällen, bei denen für alle Instanzen die gleichen Zusammenhänge gelten, kann das gleiche Vorwissen für alle Instanzen verwendet werden.

Weiterführendes

Eine interessante weiterführende Möglichkeit besteht darin, Prognosen über Instanzen von Netzfragmenten auf entfernten Geräten durchführen zu lassen bzw. entfernte Instanzen von Netzfragmenten so in beliebiger Anzahl einbinden zu können ↑. Denkbar ist auch das Einbinden von Netzfragmentinstanzen als Teil von ausgetauschtem Vorwissen.

Weiterführend kommt zu einer weiteren Verfeinerung der Unterstützung von Instanzen die Untersuchung von Konzepten wie objektorientierten bayes'schen Netzen in Frage [Wit02, S. 44]. Mit probabilistischen relationalen Modellen existiert in diesem Bereich ein Ansatz, der möglicherweise auch zur Darstellung von Relationen genutzt werden kann [Wit02, S. 44] [KP98]. Objektorientierte bayes'sche Netze zielen neben Instanzen auch auf weitere Aspekte wie eine vereinfachte Modellierung ab und unterstützen Konzepte wie Vererbung [Wit02, S. 44].

4.4.3. Verbesserter Umgang mit Unsicherheit

In Abschnitt 2.4 wird unterschieden zwischen prinzipieller Unsicherheit auf der einen Seite und Schätz-, Variations- und Methodenunsicherheit auf der anderen Seite (vgl. auch Abschnitt 2.4.10), die an dieser Stelle anhand eines kleinen Beispiels aufgegriffen werden. Prinzipielle Unsicherheit ist ebenso wie die Gesamtheit der Zusammenhänge, auf denen Prognose basiert, Teil des betrachteten Kontextes und kann durch Wahrscheinlichkeiten angegeben werden, die dann angestrebtes Ergebnis der Prognose sind ↑. Prinzipielle Unsicherheit ist unvermeidbar ↑. Was als unvermeidbar angesehen wird, kann jedoch anwendungsspezifisch festgelegt werden ↑.

Beispiel 4.2:
Ein Gerät prognostiziert, ob der Nutzer in den nächsten Tagen mit dem Auto, der Bahn oder dem Fahrrad zur Arbeit fahren wird. Mit dieser Prognose kann es z.B. entscheiden, ob für einen Monat eine Monatskarte oder Einzelfahrkarten für die Bahn günstiger sind, und selbstständig die entsprechenden Fahrkarten erwerben. Wenn das Gerät als gegebene Informationen den Monat nutzt, der die Jahreszeit ausdrückt, kann es beispielsweise als prinzipielle Unsicherheit angesehen werden, ob der Nutzer eher in Stimmung für Auto, Bahn oder Fahrrad ist. Diese Stimmung ist kaum nachvollziehbar und kann daher als unvermeidbare Unsicherheit angesehen werden. Schätzunsicherheit besteht, wenn das Prognosesystem noch nicht genug gelernt hat, um zu wissen, ob den Nutzer z.B. die Kälte im Winter stört, ob er im Sommer die mögliche Wärme in der Bahn meidet, lieber an der frischen Luft mit dem Fahrrad fährt oder nicht auf die Klimaanlage im Auto verzichten möchte. Variationsunsicherheit liegt vor, wenn sich Zusammenhänge, die durch das Vorwissen abgebildet werden und auf denen die Prognosen basieren, durch äußere Umstände ändern. Das kann z.B. dadurch der Fall sein, dass es sich alle paar Monate ändert, in welcher Niederlassung seiner Firma der Nutzer seinen Beruf ausübt, und das Prognosesystem dies nicht berücksichtigt. Für das Prognosesystem erscheint es dann so, dass der Zusammenhang zwischen Monat und Transportmittel variiert. Methodenunsicherheit entsteht durch Vereinfachungen von Methoden, die nicht immer zur Realität

passen müssen. Es könnte z.B. eine Methode eingesetzt werden, die voraussetzt, dass die Wahrscheinlichkeit der Nutzung des Fahrrades von der Mitte des Winters bis zur Mitte des Sommers linear ansteigt und von der Mitte des Sommers bis zur Mitte des Winters linear absinkt.

Das Basissystem drückt prinzipielle Unsicherheit durch Wahrscheinlichkeitsverteilungen aus, die sich entsprechend der universellen Sichtweise auf Prognose in Abschnitt 2.4 als Schätzung aus Durchführungen von Zufallsexperimenten ergeben. Eine Durchführung eines Zufallsexperiments entspricht dem Lernen einer Kombination von Variablenwerten $v_{i,j}$ als Datensatz der Trainingsdaten. Die Wahrscheinlichkeitsverteilungen als Ausdruck prinzipieller Unsicherheit werden jedoch durch Schätz-, Variations- und Methodenunsicherheit verfälscht. Daher wird für Schätz-, Variations- und Methodenunsicherheit auch der Begriff der Unsicherheit zweiter Ordnung verwendet. Zum Teil kann diesem Problem von den Anwendungsentwicklern durch gute Prognosemodelle entgegengewirkt werden, z.B. durch Einbeziehen der aktuellen Arbeitsstätte in das Prognosemodell zu obigem Beispiel zur Verringerung der Variationsunsicherheit ↑. Die in diesem Abschnitt entwickelte Erweiterung im Bereich der Unsicherheit zielt zur weiteren Verbesserung des Umgangs mit Unsicherheit darauf ab, die Möglichkeiten des Prognosesystems zum Umgang mit Unsicherheit zur Laufzeit zu verbessern. Die Erweiterung weist einen relativ experimentellen Charakter auf. Dieser Abschnitt beschäftigt sich im Detail damit, in welcher Weise eine Verfälschung der Wahrscheinlichkeitsverteilungen im Basissystem durch Schätz-, Variations- und Methodenunsicherheit stattfindet, und ausgehend davon, wie man das Ausmaß dieser Formen von Unsicherheit erkennen, angeben und teilweise verringern kann, was eine ganze Reihe von Anforderungen des Anforderungskatalogs aus Abschnitt 2.6 fordern. Die Entstehung dieser Arten von Unsicherheit betrifft die Lernschicht und ist zum Teil aus Gründen wie der Effizienz schwer vermeidbar. Ihr Ausmaß hängt jedoch von der jeweiligen Prognose ab, so dass zur Erkennung und Angabe Veränderungen des Prognosesystems in der Prognoseschicht nötig sind. Zur Verringerung von Unsicherheit werden auch Veränderungen an der Lern- und der Vorwissenschicht angesprochen werden. Zuvor wird jedoch aufgezeigt, wie Angaben zur Unsicherheit genutzt werden können, und so eine Motivation geschaffen.

Nutzung von Unsicherheitsangaben

Die Ermittlung von Angaben zur Unsicherheit stellt eine Möglichkeit der Skalierung dar. Es bietet sich daher an, die Ermittlung von Angaben zur Unsicherheit frei aktivierbar und deaktivierbar zu gestalten. Bei aktivierter Ermittlung können die Angaben zum einen für die Ergebnisintegration beim Coprocessing genutzt werden, um bei einer Prognose unabhängige Variablen mit geringer Unsicherheit stärker zu berücksichtigen und so eine geringere Unsicherheit des Gesamtergebnisses der Prognose zu erreichen (vgl. Abschnitt 3.4.1 und Abbildung 4.9). Zum anderen kann die Anwendung Angaben zur Unsicherheit als Angaben über Genauigkeit und Verlässlichkeit der Prognose zur Vermeidung falscher Entscheidungen nutzen (vgl. Abschnitt 2.5.6). Bei der Nutzung durch die Anwendung

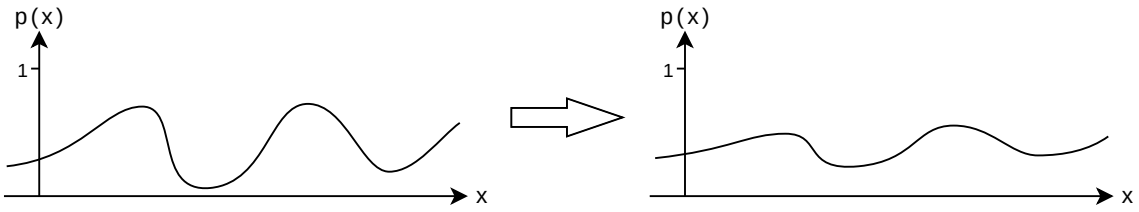

Abbildung 4.15.: Verteilung einer metrisch skalierten Variablen (links) und Ergebnis der Überlagerung mit einer Gleichverteilung (rechts)

sind die beiden Möglichkeiten zu unterscheiden, dass sich die Anwendung für die verschiedenen Fälle interessiert, die in der Zukunft eintreten können, und dass sie sich nur für den wahrscheinlichsten Fall interessiert.

Wenn die Anwendung sich für alle Fälle interessiert, also mögliche Variablenwerte bei einer Zustandsprognose und mögliche Zeitpunkte bei einer Zeitprognose, wird sie im Basissystem eine Wahrscheinlichkeitsverteilung prognostizieren lassen. Wenn man davon ausgeht, dass die Verteilung prinzipielle Unsicherheit ausdrückt, sie jedoch durch andere Arten von Unsicherheit verfälscht ist, müsste man eigentlich wiederum Qualitätsangaben über diese Verteilung machen. Viele Anwendungen interessieren sich aber nur für den Fall, der tatsächlich in der Zukunft eintritt, sind jedoch zur Vermeidung falscher Entscheidungen aufgrund der Existenz von Unsicherheit dazu gezwungen, mehrere mögliche Fälle zu berücksichtigen. Um den Aufwand beim Einsatz des Prognosesystems für Anwendungsentwickler möglichst gering zu halten, wird nach der Prognose in der Prognoseschicht eine Integration von Angaben zu unterschiedlichen Unsicherheitsarten angestrebt, indem Angaben zur Unsicherheit zweiter Ordnung in eine prognostizierte Wahrscheinlichkeitsverteilung p einfließen.

Ein einfacher Ansatz für dieses Einfließen besteht darin, $p(x_i|x_1, ..., x_{i-1})$ bei festen Werten für x_1, ..., x_{i-1} als gegebene Informationen der Prognose mit einer Gleichverteilung [Hüb03, S. 25, 37-38] $p'(x_i|x_1, ..., x_{i-1})$ zu überlagern, die allen möglichen Werten $x_i \in D(X_i)\backslash?$ den gleichen Wert zuordnet. Die Verteilung p nähert sich dann einer Gleichverteilung an (vgl. Abbildung 4.15). Der Sinn liegt darin, dass eine Gleichverteilung einen maximalen Grad an Unsicherheit aufweist, da sie keine Wertekombinationen durch hohe Wahrscheinlichkeiten bevorzugt. Die Verteilung p weist somit nach der Überlagerung einen höheren Grad an Unsicherheit und abgeschwächte Zusammenhänge auf (vgl. auch Abschnitt 2.4.8). Das ist z.B. sehr sinnvoll im Fall einer prognostizierten Verteilung, die eine hohe Schätzunsicherheit aufweist, sogar nur auf einem einzigen gelernten Datensatz basiert und daher einem einzelnen Wert eine Wahrscheinlichkeit von 100% zuordnet (z.B. mit 100% Wahrscheinlichkeit mit dem Fahrrad zur Arbeit im April in Beispiel 4.2). Aus dieser Verteilung wird eine ausgewogene Verteilung, die keine zuverlässige Aussage über den zukünftigen Wert macht, anstatt einen Wert fälschlicherweise als sicher eintretenden zukünftigen Wert zu prognostizieren.

Bei der Überlagerung ist die Gleichverteilung um so stärker zu gewichten, je größer die Unsicherheit zweiter Ordnung ist. Die Überlagerung erfolgt so, dass in einem Durchlauf des die Methoden koordinierenden Prognosealgorithmus der prognostizierte Variablen-

wert bei Zustandsprognose bzw. der prognostizierte Zeitpunkt bei Zeitprognose mit einer gewissen, sich aus der Unsicherheit zweiter Ordnung ergebenden Wahrscheinlichkeit verworfen und stattdessen ein zufälliger Wert entsprechend einer Gleichverteilung bestimmt wird. Das Verwerfen mit einer bestimmten Wahrscheinlichkeit lässt sich leicht realisieren durch Generieren einer Zufallszahl aus dem Intervall [0,1) und Festlegen einer Entscheidungsgrenze, die angibt, welchen Wert die Zufallszahl überschreiten muss zum Verwerfen des prognostizierten Wertes.

Manche Anwendungen interessieren sich nur für den wahrscheinlichsten zukünftigen Fall anstatt für alle möglichen Fälle und für Wahrscheinlichkeitsverteilungen. Eine solche Anwendung kann im Basissystem entweder den reduzierten Modus benutzen oder den Modalwert aus einer prognostizierten Verteilung bestimmen lassen. Die prognostizierte Verteilung im nicht reduzierten Modus kann zur Bestimmung von Angaben über die Unsicherheit, ob der Modalwert tatsächlich eintreten wird, genutzt werden. Im Fall einer nominal skalierten Variablen kann z.B. die Wahrscheinlichkeit des ermittelten Modalwerts abgefragt werden. Im Fall einer Variablen mit metrischem Skalenniveau kann die Wahrscheinlichkeit eines Intervalls um den Modalwert herum aus der Verteilung ermittelt werden. Durch die Integration von Unsicherheit zweiter Ordnung in die Wahrscheinlichkeitsverteilung ergibt sich also für Unsicherheitsangaben zum wahrscheinlichsten Fall als Prognoseergebnis kein Erweiterungsbedarf gegenüber dem Basissystem.

Im restlichen Teil des Abschnittes ist an manchen Stellen von der Wahrscheinlichkeitsverteilung einer Variablen die Rede. Wenn keine genaueren Angaben gemacht werden, ist damit die Verteilung gemeint, die sich aus den prognostizierten Werten dieser Variablen über mehrere Durchläufe der Prognose hinweg ergibt und die man im Basissystem erhält, wenn man eine Zustandsprognose über diese Variable durchführen lässt.

Um Unsicherheitsangaben nutzen zu können, müssen diese zunächst ermittelt werden. Um dies erreichen zu können, wird im Folgenden darauf eingegangen, welche Rolle die unterschiedlichen Arten von Unsicherheit für das Prognosesystem spielen.

Verfälschungen durch Variations-, Schätz- und Methodenunsicherheit

Da das den Variablen im Prognosemodell zugrunde liegende, die prinzipielle Unsicherheit ausdrückende Wahrscheinlichkeitsmaß im Allgemeinen von Zeitraum zu Zeitraum variieren kann, geht über den Vorgang des regelmäßigen Lernens neben prinzipieller auch Variationsunsicherheit in prognostizierte Wahrscheinlichkeitsverteilungen ein. Die Variation kann durch äußere Umstände erklärt werden, einschließlich Variablen, die Teil des Prognosemodells sind, jedoch keine Kanten zu allen Variablen aufweisen, mit denen sie zusammenhängen. Da das Prognosesystem nicht nur vom Wahrscheinlichkeitsmaß lernt, das die prinzipielle Unsicherheit für einen bestimmten Zeitraum ausdrückt (z.B. April 2009 - Juli 2009 als Zeitraum ohne Wechsel der Arbeitsstätte in Beispiel 4.2), sondern für einen potentiell längeren Zeitraum, stellt sich die Frage, welches Wahrscheinlichkeitsmaß stattdessen gelernt wird.

Zur Beantwortung dieser Frage anhand eines Beispiels seien Z_4 als abhängige Variable und X_1 und Y_3 als unabhängige Variablen angenommen, wobei die Zeitpunkte 1, 3 und 4 jeweils auf den aktuellen Messzeitpunkt der Werte der Variablen als Zeitpunkt 1 bezogen

sind. Das Lernen und die Prognose werden hier ähnlich wie zum Teil auch in Abschnitt 2.4 auf einer eher theoretischen Ebene als ein Gesamtvorgang betrachtet, um von konkreten Methoden abstrahieren zu können, die den Gesamtvorgang unterschiedlich auf Lernen und Prognose aufteilen ↑. Bei dem Gesamtvorgang wird durch das Prognosesystem eine Wahrscheinlichkeitsverteilung p(z_4 | x_1,y_3) geschätzt.

Solange die Reihenfolge der Datensätze beim Lernen keine Rolle spielt, kann man eine Zufallsvariable T mit einer Wertemenge $D(T)$, der absoluten Zeit als Wert (z.B. 01.08.2009 15:32) und $\forall_{t\in D(T)\setminus ?, t'\in D(T)\setminus ?} p(t) = p(t')$ und $p(?) = 0$ zum Wahrscheinlichkeitsmodell hinzufügen und sich vorstellen, dass ein Datensatz aus einer Lostrommel gezogen wird, der eine Variablenwertkombination x_1, y_3, z_4 enthält und außerdem mit einem Messzeitpunkt t versehen ist. Dann gibt es ein festes, nicht variierendes Wahrscheinlichkeitsmaß des Wahrscheinlichkeitsmodells, das durch eine Verteilung p(x_1,y_3,z_4,t) dargestellt werden kann. Nun ergibt sich eine Interpretation für Verteilungen wie p(x_1,y_3,z_4 | t), mit der Wahrscheinlichkeiten für einen festen Zeitpunkt ermittelt werden können, und die obige Verteilung p(x_1,y_3,z_4) = $\sum_{t\in D(T)} p(x_1, y_3, z_4, t)$, die sich als Summe über die Zeitpunkte ergibt.

In Prognosen wird also solchen Bereichen der möglichen Variablenwertkombinationen eine hohe Wahrscheinlichkeit zugeordnet, die (unter Berücksichtigung der gegebenen Informationen) zu vielen Zeitpunkten eine hohe Wahrscheinlichkeit aufweisen. Es findet somit eine Art Ausmittelung der prinzipiellen Unsicherheit für die unterschiedlichen Zeitpunkte statt. Wenn der Nutzer in Beispiel 4.2 z.B. zu nah gelegenen Arbeitsstätten immer mit dem Fahrrad fährt und zu entfernter gelegenen Arbeitsstätten immer mit dem Auto, werden Prognosen ohne besonderen Umgang mit Variationsunsicherheit mit einer gewissen Wahrscheinlichkeit das Fahrrad und mit einer gewissen Wahrscheinlichkeit das Auto als Transportmittel vorhersagen. Dass zu unterschiedlichen Zeitpunkten unterschiedliche äußere Umstände vorliegen, geht somit in sinnvoller Weise als Unsicherheit in die Wahrscheinlichkeitsverteilung ein. Dennoch ist es nützlich zu wissen, ob der Nutzer in Beispiel 4.2 sich gerade nach den Zusammenhängen verhält, die bei nah gelegenen Arbeitsstätten gelten, oder nach den Zusammenhängen, die bei entfernter gelegenen Arbeitsstätten gelten, worauf dieser Abschnitt noch zurückkommen wird.

Im Gegensatz zu Variationsunsicherheit kann man bei Schätzunsicherheit und Methodenunsicherheit schon eher von Verfälschungen prognostizierter Verteilungen sprechen. Wenn eine Wahrscheinlichkeitstabelle für eine bestimmte Wertekombination der unabhängigen Variablen erst einen Datensatz gelernt hat, wird sie bei Prognosen immer den Wert der abhängigen Variablen im gelernten Datensatz prognostizieren, so dass diesem Wert bei einer Zustandsprognose über diese abhängige Variable eine Wahrscheinlichkeit von 100% zugeschrieben wird. Schätzunsicherheit spielt bei Methoden wie Wahrscheinlichkeitstabellen, die wenig generalisieren, eine große Rolle. Bei stärker generalisierenden Methoden wie Regression steht dagegen die Methodenunsicherheit mehr im Vordergrund, wenn die Art der Generalisierung nicht vollständig der Realität entspricht, z.B. bei der Prognose eines nur näherungsweise linearen Zusammenhangs durch lineare Regression.

Grundlegende Herangehensweise zur Ermittlung von Unsicherheit

Eine wichtige Möglichkeit zum Umgang mit Unsicherheit zweiter Ordnung ist ihre Ermittlung und Angabe. Für die Ermittlung ist entsprechend obiger Ausführungen ihre Methodenabhängigkeit zu berücksichtigen. Unsicherheit wird außerdem durch Variablenwerte beeinflusst. Im Beispiel der von einer Wahrscheinlichkeitstabelle prognostizierten Verteilung, die auf nur einem gelernten Datensatz basiert, ist es durchaus möglich, dass die Wertekombination der unabhängigen Variablen selten auftritt und dass für andere, häufigere Wertekombinationen über 100 Datensätze in die prognostizierten Verteilungen eingehen. Bei Regression kann der Zusammenhang zwischen der abhängigen und der unabhängigen Variablen z.B. für kleine und mittlere Werte der unabhängigen Variablen gut durch eine Regressionsgerade beschreibbar sein und für größere Werte könnte der Graph abflachen und sich einer oberen Schranke annähern. Da Unsicherheit von Variablenwerten abhängt, sollte sie erst beim Prognostizieren in der Prognoseschicht und individuell für jede einzelne Prognose bestimmt werden. Der Einfluss von Variablenwerten und die Methodenabhängigkeit von Unsicherheit sprechen außerdem dafür, Unsicherheit zunächst lokal für jede Variable einzeln zu ermitteln. Man kann hier ähnlich argumentieren wie für den Einsatz mehrerer, austauschbarer Prognosemethoden, die lokal arbeiten (vgl. Abschnitt 3.2.4). Eine denkbare globale Ermittlung in Abhängigkeit von den Werten der gegebenen Variablen der Prognose durch eine einzelne Methode, die speziell zur Unsicherheitsermittlung eingesetzt wird, würde das Potential der Berücksichtigung von Wissen über die einzelnen Methoden nicht nutzen und müsste mit einer potentiell riesigen Menge möglicher Wertekombinationen von Variablen als gegebene Informationen der Prognose umgehen können.

Im Bezug auf die lokale Unsicherheitsermittlung kann man sich fragen, ob eine Methode einer Variablen selbst die Unsicherheit bestimmen soll oder ob dafür eine separate Einheit erwünscht ist, die einer Variablen wie eine Prognosemethode zugeordnet ist, jedoch unabhängig von Prognosemethoden arbeitet. Eine direkte Ermittlung durch die Methoden hat den Vorteil, dass das Wissen über das Verhalten der jeweiligen Methoden bezüglich Unsicherheit voll genutzt werden kann, was die Unsicherheitsermittlung erleichtert und somit auch einen positiven Faktor für ihre Effizienz darstellt. Eine direkte Ermittlung durch die Methoden weist jedoch auch die Nachteile auf, dass Anwendungsentwickler bei der Implementierung neuer Methoden zusätzlich belastet werden und dass eine Ermittlung von Unsicherheit vermutlich wegen der Heterogenität von Methoden nicht bei allen Methoden einfach durchführbar ist. Daher sollte ein methodenspezifischer Umgang mit Unsicherheit optional sein für Methoden.

Separate Einheiten für die Unsicherheitsermittlung als Alternative zur Ermittlung durch die Methoden besitzen, wenn der Anwendungsentwickler voll von der Unsicherheitsermittlung entlastet sein soll, kein Wissen über Methoden und müssen daher die Prognosen der Methoden überwachen und lernen, mit wie hoher Unsicherheit zweiter Ordnung sie bei unterschiedlichen Wertekombinationen der unabhängigen Variablen arbeiten. Das Ausmaß der Unsicherheit zweiter Ordnung der Prognose einer Methode ist aber nicht leicht von außen ermittelbar, da im Endeffekt Wahrscheinlichkeitsverteilungen prognostiziert werden. In der Realität, die in Frage kommt zur Prüfung der Genauigkeit

einer Prognose im Nachhinein, also zur Prüfung, wie stark eine prognostizierte Verteilung durch Unsicherheit zweiter Ordnung verfälscht ist, tritt aber nur ein einzelner Wert ein. Es kann also nicht einfach durch eine separate Einheit ermittelt werden, ob die Prognose einer Methode für eine Variable die prinzipielle Unsicherheit gut widerspiegelt. Eines von weiteren Problemen separater Einheiten liegt darin, dass die Unsicherheitsermittlung mit separaten Einheiten selbst mit Unsicherheit behaftet ist. Wenn z.B. wenig Prognosen durchgeführt werden, kann die Unsicherheitsermittlung unter Schätzunsicherheit leiden. Daher wird eine direkte Ermittlung von Unsicherheit durch die Methoden als primärer Ansatz gewählt.

Fortpflanzung von Unsicherheit im Netz

Da Unsicherheit nach obiger Entscheidung lokal von den Methoden der einzelnen Variablen ermittelt werden soll, bedarf es eines Ansatzes, um in der Prognoseschicht, wenn eine Prognose durchgeführt wird, die Angaben zur Unsicherheit im Prognosenetz zu propagieren. Wenn beim Vorgang der Prognose die über mehrere Durchläufe der Prognose hinweg prognostizierten Werte einer Variablen (bzw. die sich daraus ergebende Verteilung) durch Unsicherheit zweiter Ordnung verfälscht sind und daraus wiederum weitere Werte prognostiziert werden, wirkt sich die Verfälschung auch auf diese aus. So kommt man zu der Frage, in welcher Weise die Prognose einer abhängigen Variablen verfälscht wird, wenn die Werte der unabhängigen Variablen mit Unsicherheit zweiter Ordnung behaftet sind und bei der Prognose der abhängigen Variablen durch ihre Methode ebenfalls Unsicherheit zweiter Ordnung auftritt. Ein wesentliches Problem liegt darin, dass man nicht oder nur vage weiß, in welcher Weise die Werte der unabhängigen Variablen verfälscht sind. Wenn man es genauer wüsste, könnte man sie gleich besser prognostizieren. Die Unsicherheit zweiter Ordnung ist jedoch abhängig von den Werten der unabhängigen Variablen ↑. Somit kann die Genauigkeit von Unsicherheitsangaben bei Prognosen in die ferne Zukunft deutlich abnehmen.

Ein einfacher Mechanismus zur Fortpflanzung von Unsicherheitsangaben im Netz besteht darin, die oben beschriebene Integration von Angaben zur Unsicherheit zweiter Ordnung in eine Wahrscheinlichkeitsverteilung bei einer Prognose für jede einzelne Variable durchzuführen, indem in jedem Durchlauf des Prognosealgorithmus mit einer gewissen Wahrscheinlichkeit, die sich aus dem Ausmaß ihrer lokal ermittelten Unsicherheit zweiter Ordnung ergibt, der prognostizierte Wert verworfen und stattdessen ein zufälliger Wert entsprechend einer Gleichverteilung gewählt wird. Die sich aus mehreren Durchläufen des Prognosealgorithmus ergebende Verteilung der unabhängigen Variablen wird dadurch gleichmäßiger (vgl. Abbildung 4.15), so dass in den meisten Fällen auch die Werte der abhängigen Variablen gleichmäßiger verteilt sind. Auf diese Weise kommt die Unsicherheit in den Verteilungen der unabhängigen Variablen zur Unsicherheit der abhängigen Variablen hinzu, so dass sich die Unsicherheit, die zu erwarten ist, über mehrere Prognoseschritte hinweg vergrößert. Es wird außerdem unterbunden, dass verfälschte Werte der unabhängigen Variablen die von diesen Werten abhängige lokale Unsicherheitsermittlung durch die Methode der abhängigen Variablen in eine bestimmte, falsche Richtung lenken. Ein Problem liegt allerdings darin, dass die Schätzunsicherheit

der Methode der abhängigen Variablen recht groß wird, wenn durch die Überlagerung mit einer Gleichverteilung Werte auftreten, die in der Realität fast nie eintreten. Alternativ zu einer Gleichverteilung könnte man auch über die Verwendung der unbedingten Verteilung $p(x)$ einer Variablen X nachdenken. Diese Verteilung muss aber nicht unbedingt einen hohen Grad an Unsicherheit aufweisen. Zu prüfen wären daher Möglichkeiten der sinnvollen Kombination einer Gleichverteilung mit der unbedingten Verteilung.

Zu berücksichtigen ist jedoch, dass beim Coprocessing als eine Nutzungsmöglichkeit von Unsicherheitsangaben der Unterschied zwischen Unsicherheit zweiter Ordnung und prinzipieller Unsicherheit, die Eigenschaft des Kontextes ist und daher nicht verringert, sondern beschrieben werden muss, eine Rolle spielt. Daher ist die Integration von Angaben zur Unsicherheit zweiter Ordnung in Wahrscheinlichkeitsverteilungen dafür nicht ausreichend. Das wird auch an Beispiel 4.2 deutlich. Beim Einsatz von Coprocessing in diesem Beispiel könnte etwa eine der parallel eingesetzten Methoden durch Wahrscheinlichkeiten die prinzipielle Unsicherheit ausdrücken, dass der Nutzer manchmal in der Stimmung für das Auto, manchmal in der Stimmung für die Bahn und manchmal in der Stimmung für das Fahrrad ist. Eine andere Methode könnte einer einzelnen der drei Fortbewegungsmöglichkeiten eine Wahrscheinlichkeit von 100% zuordnen und die prinzipielle Unsicherheit einfach ignorieren. Bei der Ergebnisintegration ist in diesem Fall gerade nicht die weniger gleichmäßige Verteilung, die einem Wert 100% Wahrscheinlichkeit zuordnet, zu bevorzugen, da es sich bei der Unsicherheit in der Verteilung nur um die zu beschreibende prinzipielle Unsicherheit und keinen Mangel der Prognose handelt. Eine Ergebnisintegration kann außerdem auch aus dem Grund nicht nur auf Unsicherheitsangaben basieren, die in die Verteilung integriert sind, da in einem Durchlauf des Prognosealgorithmus nur ein Wert und keine Verteilung vorliegt.

Die dafür vorgesehene Lösung ist, dass alle Methoden in jedem Durchlauf bei der Prognose zwei Werte prognostizieren. Der erste ist der gleiche wie im Basissystem. Beim zweiten erfolgt zusätzlich eine Integration der Unsicherheit zweiter Ordnung wie gerade beschrieben. Es liegen dann bei jeder Prognose einer Methode für jede unabhängige Variable diese beiden Werte vor. Wenn die Methode den Wert zu ihrer Variablen bestimmt, in den die Unsicherheit zweiter Ordnung integriert ist, verwendet sie die Werte der unabhängigen Variablen, in die die Unsicherheit zweiter Ordnung integriert ist. Wenn sie den Wert bestimmt, in den keine Unsicherheit zweiter Ordnung integriert ist, verwendet sie die Werte der unabhängigen Variablen, in die keine Unsicherheit zweiter Ordnung integriert ist. Beim Coprocessing liegen dann auch für jede der parallel arbeitenden Methoden jeweils zwei solche Werte vor. Zur Ermittlung der Unsicherheit zweiter Ordnung bei der Ergebnisintegration werden die beiden Werte miteinander verglichen. Im Fall einer nominal oder ordinal skalierten Variablen können die Werte gleich oder unterschiedlich sein. Bei hoher Unsicherheit zweiter Ordnung sind die Werte in vielen Durchläufen der Prognose unterschiedlich, so dass das Ergebnis der Methode häufig weniger stark bei der Ergebnisintegration einfließt. Bei metrisch skalierten Variablen kann in jedem Durchlauf eine Abweichung zwischen den beiden Werten berechnet werden. Eine große Abweichung deutet auf eine hohe Unsicherheit zweiter Ordnung hin.

Lokale Ermittlung von Schätzunsicherheit durch Methoden

Grundlage aller Unsicherheitsangaben ist die lokale Ermittlung von Unsicherheit zweiter Ordnung für die Methoden der einzelnen Variablen. Wegen der Abhängigkeit der Unsicherheit von Variablenwerten erfolgt die Ermittlung erst beim Prognostizieren in der Prognoseschicht und individuell für jede einzelne Prognose. Die Ermittlung von Schätzunsicherheit durch Methoden kann z.B. so erfolgen, dass diese einfach wissen, wie viele Datensätze sie gelernt haben müssen, um eine geringe Schätzunsicherheit zu erreichen. Bei Regression und der empirischen Verteilungsfunktion genügt das schon. Für Wahrscheinlichkeitstabellen bietet es sich darüber hinausgehend an zu unterscheiden, wie viele Datensätze beim Lernen in die jeweils zur Prognose herangezogene Verteilung $p(x_i|x_1, ..., x_{i-1})$ bei festen Werten $x_1, ..., x_{i-1}$ eingegangen sind. Offen ist noch die Frage, wie man Schätzunsicherheit angibt. Man könnte z.B. einen Minimalwert definieren, der aussagt, dass weiteres Lernen bei Abwesenheit von anderen Unsicherheitsarten zu keiner Verbesserung mehr führt, und einen Maximalwert, der aussagt, dass noch nichts gelernt wurde. Die Werte dazwischen könnten z.B. durch den Erwartungswert der Abweichung der aktuell gelernten Verteilung von der tatsächlichen Verteilung unter Annahme der Abwesenheit anderer Arten von Unsicherheit sein. Es ist jedoch zu befürchten, dass ein wohldefiniertes Maß für Schätzunsicherheit das Problem mit sich bringt, dass die Schätzunsicherheit damit von vielen Methoden nicht effizient berechnet werden kann.

Angaben zur Schätzunsicherheit können nicht nur genutzt werden, indem sie in Angaben zur Unsicherheit für die Anwendung und für die Ergebnisintegration beim Coprocessing eingehen, sondern auch zur Regulierung der Häufigkeit, mit der Daten akquiriert und als Datensätze gelernt werden (vgl. Abschnitt 2.5.7). Da Angaben zur Schätzunsicherheit dann sowohl in der Prognoseschicht, als auch in der Lernschicht benötigt werden, macht es Sinn, dass die Ermittlung in die Vorwissenschicht verlagert wird. Bei hoher Schätzunsicherheit kann dann mehr gelernt werden als bei geringer. Allerdings macht häufiges Lernen nur Sinn, wenn sich die Werte auch entsprechend häufig ändern. Zu berücksichtigen ist außerdem, dass es durch Änderung der Lernhäufigkeit dazu kommen kann, dass bei Gegenwart von Variationsunsicherheit bestimmte Varianten der Zusammenhänge intensiver gelernt werden und dadurch stärker in Prognosen eingehen. Es findet dann eine Verzerrung der Stichprobe statt.

Lokale Ermittlung von Methodenunsicherheit durch Methoden

Eine methodenspezifische Ermittlung von Methodenunsicherheit kann bei Regression z.B. so erfolgen, dass für unterschiedliche Bereiche der Wertekombination der unabhängigen Variablen gespeichert wird, wie groß der quadratische Fehler ist. Bei Wahrscheinlichkeitstabellen und der empirischen Verteilungsfunktion erübrigt sich die Ermittlung, da sie keine nennenswerte Methodenunsicherheit aufweisen. Wenn man Entscheidungsbäume verwenden würde, könnte man beispielsweise darüber nachdenken, die Blätter im Baum mit Angaben zur Methodenunsicherheit zu versehen. Bei Methodenunsicherheit stellt sich ebenso wie bei Schätzunsicherheit das noch offene Problem der Definition eines Maßes für die Unsicherheit.

Lokale Ermittlung von Variationsunsicherheit durch Methoden

Variationsunsicherheit ist dadurch gekennzeichnet, dass sich die beobachteten Zusammenhänge zwischen unterschiedlichen Zeiträumen unterscheiden. Oben in diesem Abschnitt wurde ausgeführt, dass Variationsunsicherheit im Basissystem in sinnvoller Weise in prognostizierte Wahrscheinlichkeitsverteilungen eingeht und dabei eine Art Ausmittelung über die sich ändernden Zusammenhänge stattfindet. Dadurch werden jedoch prinzipielle Unsicherheit und Variationsunsicherheit vermischt und sind nicht mehr leicht zu unterscheiden. Für das Coprocessing ist die Unterscheidung zwischen prinzipieller Unsicherheit und Unsicherheit zweiter Ordnung jedoch wichtig, da eine hohe prinzipielle Unsicherheit des Prognoseergebnisses einer Methode nicht zu dessen Benachteiligung bei der Ergebnisintegration führen darf, denn prinzipielle Unsicherheit ist unvermeidbar und zu beschreiben statt zu verringern ↑. Am Ende von Abschnitt 2.4.9 wurde festgestellt, wie nah prinzipielle Unsicherheit und Variationsunsicherheit beieinander liegen. Es wurde dort vereinbart, dass anwendungsspezifisch festzulegen ist, was als prinzipielle Unsicherheit anzusehen ist, also als unvermeidbare Unsicherheit, die im Kontext und nicht in der Prognose begründet liegt ↑ ↑. In Beispiel 4.2 über die Prognose des Transportmittels für den Weg zur Arbeit ist relativ klar, dass man die Unsicherheit in prognostizierten Wahrscheinlichkeitsverteilungen, die durch die unterschiedlichen Arbeitsstätten entsteht, als Variationsunsicherheit ansieht. Bei einer Prognose in Abhängigkeit vom Wetter würde man dagegen die Unsicherheit des prognostizierten Wetters eher als prinzipielle Unsicherheit auffassen. Die Frage ist nun, wie man prinzipielle Unsicherheit und Variationsunsicherheit trennen kann.

Zur Laufzeit schwer zu erkennen sind mehrere schnelle Änderungen der Zusammenhänge in kurzen Abständen, da Zusammenhänge in der Regel, unter Anderem durch prinzipielle Unsicherheit, erst nach dem Lernen einiger Datensätze adäquat abgebildet werden können, so dass keine schnelle Erkennung von Änderungen möglich ist. Ein wichtiger Fall ist jedoch, dass die Unsicherheit, die in der Ausgewogenheit der Wahrscheinlichkeitsverteilung der Variablen Ausdruck findet, komplett oder überwiegend als Variationsunsicherheit angesehen wird. Dieser Fall liegt häufig vor, wenn Coprocessing eingesetzt wird und jede Methode nur einen Teil der unabhängigen Variablen verwendet (vgl. Abbildung 4.9 rechts). Dann stellen für die einzelnen Methoden schon Variablen, die Teil des Prognosemodells sind, äußere Einflüsse dar, die zu Variationen der beobachteten Zusammenhänge führen. Ob der Fall vorliegt, dass die komplette Unsicherheit der Wahrscheinlichkeitsverteilung als Variationsunsicherheit angesehen wird, muss für jede Variable im Prognosemodell angegeben werden. Wenn der Fall vorliegt, kann das Ausmaß der Variationsunsicherheit daran abgelesen werden, wie ausgewogen die prognostizierte Wahrscheinlichkeitsverteilung der Variablen ist. Wenn sie einem Wert eine Wahrscheinlichkeit von 100% zuordnet, liegt z.B. gar keine Variationsunsicherheit vor. Die Wahrscheinlichkeitsverteilung kann dann außerdem in gewisser Weise um die Variationsunsicherheit bereinigt werden, indem immer der wahrscheinlichste Wert durch die Methode prognostiziert wird. Das heißt, dass Methoden, die grundsätzlich Wahrscheinlichkeitsverteilungen prognostizieren, nicht wie bisher im Basissystem zufällig einen Wert entsprechend dieser Verteilung wählen, sondern stattdessen den Modelwert der Vertei-

lung prognostizieren. Wenn man nun in Beispiel 4.2 das Transportmittel durch mehrere Methoden im Coprocessing prognostiziert und eine Methode das Transportmittel mit dem Transportmittel des Vortages als unabhängiger Variablen prognostiziert, ergibt sich ein Nutzen durch dieses Vorgehen. Wenn z.B. bei der Bahn als Transportmittel des Vortages eine Wahrscheinlichkeitsverteilung prognostiziert wird, die der Bahn am aktuellen Tag 38%, dem Auto 30% und dem Fahrrad 32% Wahrscheinlichket zuordnet, wird die hohe Unsicherheit zweiter Ordnung der Prognose dieser Methode bei der Ergebnisintegration berücksichtigt, so dass sie weniger stark in das Gesamtergebnis eingeht. Wenn jedoch beim Fahrrad als Transportmittel des Vortags dem Fahrrad am aktuellen Tag 85% Wahrscheinlichkeit zugeordnet wird, hat die Prognose der Methode gute Chancen, stark in das Endergebnis einzugehen.

Für die Erkennung seltener Änderungen von Zusammenhängen bietet sich im Gegensatz zum Fall schneller Änderungen ein anderes Vorgehen an, das nicht nur angewendet werden kann, wenn kaum oder keine prinzipielle Unsicherheit vorliegt. Das Vorgehen basiert auf einem Vergleich des Methodenvorwissens auf der Instanzebene für unterschiedliche Zeiträume, um Änderungen der durch das Vorwissen abgebildeten Zusammenhänge zu erkennen. Dazu wird neben dem für die Prognose genutzten Methodenvorwissen auf der Instanzebene separates Methodenvorwissen auf der Instanzebene gespeichert und beim Lernen einbezogen, das sich nur auf die jüngere Vergangenheit bezieht.

Die Verwaltung dessen kann durch den von konkreten Methoden unabhängigen Koordinationsteil der Vorwissenschicht durchgeführt werden. Wenn das Methodenvorwissen auf der Instanzebene schon zu viele alte Informationen enthält, muss es verworfen werden. Es bietet sich an, schon eine Weile vorher neues Methodenvorwissen auf der Instanzebene zu erstellen, das das verworfene ablösen kann. Dieser Ansatz, für die jüngere Vergangenheit Methodenvorwissen auf der Instanzebene mehrfach zu ermitteln und zu speichern, erfordert zwar zusätzlichen Speicherplatz, vermeidet aber das Problem, dass selektiv nur auf den jüngeren Teil von Methodenvorwissen auf der Instanzebene zugegriffen werden muss, in das auch ältere Datensätze eingegangen sind. Ein solcher Zugriff ist nicht möglich bei Methoden, die beim Lernen von den Zeitpunkten der Datensätze der Trainingsdaten abstrahieren.

Methoden müssen allerdings ein Ähnlichkeitsmaß für ihr Methodenvorwissen auf der Instanzebene bereitstellen, um das zur Prognose verwendete mit dem sich auf die jüngere Vergangenheit beziehenden vergleichen zu können. Für Wahrscheinlichkeitstabellen könnte ein solches Ähnlichkeitsmaß z.B. den Betrag der Differenz von den Werten beider Tabellen für $p(x_i|x_1, ..., x_{i-1})$ berechnen, dies über die möglichen Wertekombinationen von $X_1, ..., X_i$ aufaddieren und das Ergebnis normieren. Es dürfen allerdings nur Wertekombinationen mit nicht zu hoher Schätzunsicherheit einbezogen werden, denn leere Bereiche der Tabellen, in die noch keine Datensätze eingegangen sind und die insbesondere in der Tabelle mit den jungen Daten zu erwarten sind, bieten keine geeignete Grundlage für einen Vergleich.

Anpassung an Änderungen von Zusammenhängen

Die Ermittlung von Variationsunsicherheit kann zu einer Anpassung an Änderungen von Zusammenhängen erweitert werden. Dies wird möglich durch Wechsel des Methodenvorwissens auf der Instanzebene bei einem Zusammenhangswechsel. Man muss also Methodenvorwissen auf der Instanzebene für unterschiedliche Zusammenhänge unterscheiden. Ein solcher Ansatz wird auch bei Wittig erwähnt [Wit02, S. 65-66]. Man kann dann z.B. für die Situation eines Urlaubs anderes Methodenvorwissen auf der Instanzebene verwenden. Es muss dann allerdings erkannt werden, welches Methodenvorwissen auf der Instanzebene das jeweils aktuelle ist. Das kann durch Vergleich mit dem Methodenvorwissen auf der Instanzebene für die jüngere Vergangenheit erfolgen. Die Prognose muss immer mit dem aktuellen Methodenvorwissen auf der Instanzebene erfolgen. Das Lernen muss ebenso für das aktuelle Methodenvorwissen auf der Instanzebene und zusätzlich auch für das Methodenvorwissen auf der Instanzebene, das die jüngere Vergangenheit beschreibt, durchgeführt werden. Wenn kein Methodenvorwissen auf der Instanzebene zu den aktuellen Zusammenhängen passt, muss neues erstellt werden. Alternativ kann man auch auf universelles Methodenvorwissen auf der Instanzebene zurückgreifen, das immer beim Lernen einbezogen wird und bei der Prognose verwendet wird, wenn kein anderes Methodenvorwissen auf der Instanzebene besser passt. Es stellt dann variierende Zusammenhänge in der gleichen Weise wie ohne Anpassung an Änderungen von Zusammenhängen dar.

Insgesamt wird eine Anpassung an Änderungen von Zusammenhängen sowohl temporären, langsamen Änderungen von Zusammenhängen, als auch permanenten, langsamen Änderungen von Zusammenhängen gerecht. Durch frühzeitiges Einführen von neuem Methodenvorwissen auf der Instanzebene für bestimmte Zusammenhänge kann auch schleichenden Änderungen von Zusammenhängen begegnet werden im Gegensatz zum Basissystem, bei dem vor langer Zeit gelernte Trainingsdaten ebenso stark in Prognosen eingehen wie vor kurzer Zeit erlernte, sofern die jeweiligen Methoden nicht selbst z.B. die jüngere Vergangenheit stärker gewichten können. Das Konzept des Wechsels von Methodenvorwissen auf der Instanzebene ist auch von Nutzen, wenn die Anwendungsentwickler a priori bekanntes, zur Entwicklungszeit aufbereitetes Methodenvorwissen auf der Instanzebene verwenden möchten und Nutzergruppen existieren, in denen sich die Nutzer ähneln (vgl. Abschnitt 2.5.4). Es kann dann für jede Nutzergruppe zur Entwicklungszeit Methodenvorwissen auf der Instanzebene erstellt werden. Die Auswahl des zum jeweiligen Nutzer passenden Vorwissens kann automatisch durch den Mechanismus zum Wechsel von Methodenvorwissen auf der Instanzebene durchgeführt werden. Eine weitere Möglichkeit, die der Mechanismus bietet, ist das Vorgeben von Methodenvorwissen auf der Instanzebene durch die Anwendungsentwickler, das ganz grob zu allen Nutzern passt. Dadurch wird die hohe Schätzunsicherheit bei Inbetriebnahme des Prognosesystems vermieden. Das Prognosesystem kann dann zur Laufzeit neues Methodenvorwissen auf der Instanzebene anlegen und dieses statt des von den Anwendungsentwicklern erstelltem für die Prognose verwenden, sobald genug Trainingsdaten gelernt sind.

5 Implementierung eines Prognoseframeworks

In diesem Kapitel erfolgt eine Umsetzung des in Kapitel 4 erarbeiteten Verfahrens der strukturierten Kontextdatenprognose als Prognoseframework. Im Rahmen dieser Arbeit erfolgt eine prototypische Implementierung des Basissystems. Manche Funktionalitäten von Erweiterungen sind jedoch im Zuge der Anpassbarkeit und Erweiterbarkeit durch die Anwendungsentwickler rudimentär realisierbar. Da ein Ziel dieser Arbeit die Prognose der Verfügbarkeit von Diensten im Rahmen des DEMAC-Projektes ist ↑, basiert die Implementierung auf dem *Personal Profile* (JSR-216) der *Java Micro Edition*™, für das auch die DEMAC-Middleware entwickelt ist. Die Implementierung ist so gestaltet, dass sie als JAR-Datei von einer Anwendung benutzt werden kann.

Ein wesentliches Grundprinzip des Verfahrens liegt im Einsatz mehrerer Methoden, die von den Anwendungsentwicklern anwendungsspezifisch im Prognosemodell ausgewählt und bei Bedarf auch selbst implementiert werden können ↑. Auch Messadapter sollen von den Anwendungsentwicklern ausgewählt und implementiert werden können ↑. Dies wird durch die Realisierung des Verfahrens als Framework erreicht. Durch Interfaces und abstrakte Klassen werden Schnittstellen definiert, die von einer konkreten Methode bzw. einem konkreten Messadapter zu implementieren sind. Auf diese Weise benötigt das Prognosesystem keine Kenntnis von den konkreten Methoden, sondern nur von der allgemeinen Schnittstelle für Methoden. Methoden können dann wie Plugins vom Prognosesystem verwendet werden, ohne dass bestehender Quellcode des Prognosesystems modifiziert werden muss. Updates dieses bestehenden Quellcodes sind somit auch beim Einsatz für unterschiedlichste Anwendungen problemlos möglich.

Dieses Kapitel ist unterteilt in die Implementierung der einzelnen Elemente der Architektur des Prognosesystems und Aspekte, die das Gesamtsystem betreffen. Darauffolgend wird aus der Perspektive von Anwendungsentwicklern aufgezeigt, wie sich die Realisierung des Einsatzes des Prognosesystems in der DEMAC-Middleware ergibt.

5.1. Architekturelemente des Prognosesystems

Abbildung 5.1 gibt einen Überblick über die Ausgestaltung der Architektur des Basissystems. Sie stellt Vorwissen-, Datenakquisitions-, Lern- und Prognoseschicht des Prognosesystems mit den wichtigsten Klassen und Beziehungen dazwischen dar. Unterhalb des Prognosesystems befinden sich gespeicherte Daten im Dateisystem. Oberhalb des Prognosesystems befindet sich die Anwendung oder alternativ ein Kontextsystem, das mit Kontext in Verbindung stehende Aktivitäten für die Anwendung übernimmt. Die Klasse `PredictionSystem` bildet die zentrale Schnittstelle des Prognosesystems nach oben. Die einzelnen Bestandteile des Prognosesystems im Diagramm werden im Folgenden erläutert.

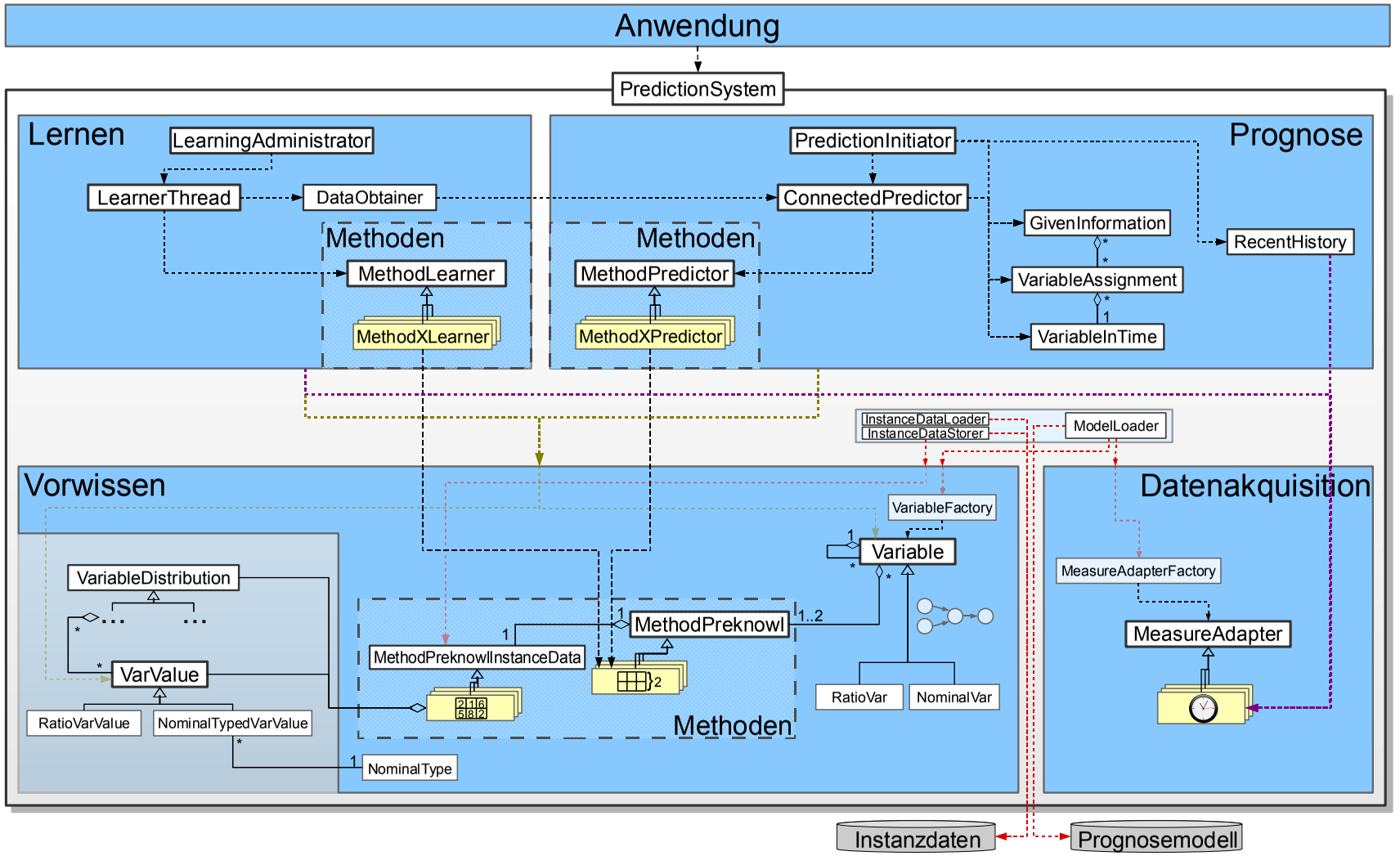

Abbildung 5.1.: Klassendiagramm zur Implementierung, aufbauend auf Abbildung 4.1 über die Architektur des Verfahrens, gelbe Klassen gehören zu konkreten Methoden und Messadaptern

5.1.1. Vorwissen

In Abschnitt 4.2.1 wird erstens unterschieden zwischen Methodenvorwissen, das zu konkreten Methoden gehört, und Koordinationsvorwissen, das im Wesentlichen durch das Prognosenetz gebildet wird. Zweitens wird unterschieden zwischen dem Prognosemodell, das zur Entwicklungszeit entsteht und sich zur Laufzeit nicht mehr ändert, sowie dem Vorwissen auf der Instanzebene, das beim adaptiven Online-Lernen entsteht und sich ändert. Im Basissystem existiert nur Methodenvorwissen auf der Instanzebene, jedoch kein Koordinationsvorwissen auf der Instanzebene ↑ ↑. Das Vorwissen wird zum einen als Objekte in der Vorwissenschicht im Prognosesystem dargestellt. Zum anderen existiert auch eine externe Repräsentation des Prognosemodells als Teil des Vorwissens, das die Anwendungsentwickler zur Entwicklungszeit erstellen müssen (vgl. Abbildung 5.1 unten). Es handelt sich um eine XML-Repräsentation. Dieser Abschnitt erläutert zunächst die Darstellung von Vorwissen in der Vorwissenschicht und geht dann auf die XML-Repräsentation des Prognosemodells ein.

Vorwissenschicht des Prognosesystems

Das Methodenvorwissen ist im Klassendiagramm in Abbildung 5.1 ebenso wie in Abbildung 4.1 über die Architektur in einem durch eine gestrichelte Linie begrenzten Kasten in der Vorwissenschicht dargestellt. Das Vorwissen auf der Instanzebene ist durch die abstrakte Klasse `MethodPreknowlInstanceData` bzw. deren Unterklassen repräsentiert. Die Klasse `MethodPreknowl` bzw. deren Unterklassen enthalten das Methodenvorwissen, das zum Prognosemodell gehört. Besonders wichtig ist auch die abstrakte Klasse `Variable` als Teil des Prognosemodells. Sie stellt eine Variable des Prognosenetzes dar. Das schon konzeptionell nicht zur Laufzeit änderbare Prognosemodell wird, soweit wie möglich, durch unveränderbare Objekte dargestellt.

Die Interfaces `VariableDistribution` und `VarValue` sowie die implementierenden Klassen zählen nur bedingt zum Vorwissen, also zum Wissen über die Zusammenhänge im Kontext für Prognosen, und können nicht eindeutig einer der oben wiederholten Arten von Vorwissen zugeordnet werden. Ein `VarValue` ist ähnlich wie eine `VariableDistribution` häufig Teil des Vorwissens auf der Instanzebene, kann aber auch Teil des Prognosemodells sein oder erst bei der Prognose entstehen, z.B. als Prognoseergebnis, oder in der Datenakquisitionsschicht als Messergebnis.

Die das Interface `VarValue` implementierenden Klassen `RatioVarValue` und `NominalTypedVarValue` stellen einen Variablenwert mit Ratio-Skalenniveau (z.B. Anzahl verpasster Telefonanrufe) bzw. dem für Kontext wichtigen nominalen Skalenniveau (z.B. diskreter Ort „zu Hause“) dar ↑. Die Implementierung folgt der üblichen Darstellung von Werten mit Ratio-Skalenniveau als reelle Zahlen. Die Klasse `NominalTypedVarValue` unterstützt die Repräsentation von diskreten Werten mit nominalem Skalenniveau aus einer endlichen Wertemenge. Als Werte sind prinzipiell beliebige serialisierbare Objekte erlaubt. Da eine Variable in der Regel aber nur bestimmte Werte annehmen kann, ist jedem `NominalTypedVarValue` ein `NominalType` zugeordnet, der die möglichen Werte beschränkt, indem er die Menge der durch diesen

Typen erlaubten Werte speichert. Auf diese Weise ergibt sich Typsicherheit. Neben einem `NominalType` können auch `RatioVarValue` und `NominalTypedVarValue` selbst als Typen angesehen werden. Das Skalenniveau eines Wertes definiert also dessen Typ, damit z.B. eine Methode wie Regression, die einen Wert mit Ratio-Skalenniveau als Eingabe erwartet, keinen Wert mit nominalem Skalenniveau übergeben bekommen kann.

Eine Unterscheidung zwischen Ratio-Skalenniveau und nominalem Skalenniveau durch Unterklassen erfolgt neben Variablenwerten auch bei Variablen und bei einigen anderen Klassen, für die dies aus Platzgründen nicht in Abbildung 5.1 dargestellt ist. Die Sicherstellung, dass die Typen der Variablen im Prognosemodell geeignet gewählt sind, erfolgt schon, wenn die Objekte in der Vorwissenschicht erzeugt werden. Eine Variable referenziert (indirekt über weitere Objekte) die Variablen, mit denen sie über Eingangskanten verbunden ist. Beim Festsetzen dieser Variablen erkundigt sich eine `Variable` bei ihrem `MethodPreknowl`, welche Art und Anzahl an unabhängigen Variablen die Methode unterstützt. Insgesamt wird bei der Erzeugung der Objekte des Prognosemodells in der Vorwissenschicht eine relativ weitreichende Konsistenzprüfung des Prognosemodells durchgeführt, um ein Scheitern beim Lernen und bei Prognosen durch Fehler im Prognosemodell zu verhindern. Einzig eine Prüfung auf die in Prognosenetzen verbotenen Zyklen, in denen alle Kanten mit dem Zeitversatz 0 versehen sind, ist nicht implementiert.

Das Erzeugen der Objekte des Prognosemodells in der Vorwissenschicht erfolgt regulär durch Laden der XML-Repräsentation des Prognosemodells mit Hilfe des `ModelLoader`. Dieser erzeugt über Fabriken Instanzen von Klassen wie z.B. von der Klasse `Variable` und auch von anderen Klassen, für die dies aus Platzgründen nicht in Abbildung 5.1 dargestellt ist. Für das Vorwissen auf der Instanzebene (Instanzdaten) wird auf eine XML-Repräsentation verzichtet, da das Editieren von Instanzdaten durch Menschen lediglich für ein mögliches Einbringen von a priori vorhandenem Wissen von Nutzen ist. Da sich das Prognosemodell nicht zur Laufzeit ändert, müssen XML-Daten dadurch nur geladen, jedoch nicht gespeichert werden. Die Instanzdaten werden unter Einsatz des Java-Serialisierungsmechanismus gespeichert und geladen.

XML-Repräsentation des Prognosemodells

Wenn man die Klassen `VariableDistribution` und `VarValue` nicht zum Vorwissen zählt, verbleiben `MethodPreknowlInstanceData` als Vorwissen auf der Instanzebene und der restliche Teil der Vorwissenschicht als Prognosemodell. Wesentliche Bereiche des Prognosemodells sind in der Darstellung der Vorwissenschicht die Klassen `Variable`, `MethodPreknowl` und `NominalType`. Diese drei Bereiche bilden auch die Grundlage der XML-Repräsentation des Prognosemodells. Die Variablen bilden das Prognosenetz. Den Variablen ist ihr Methodenvorwissen zugeordnet und Variablen mit nominalem Skalenniveau besitzen zusätzlich einen nominalen Typen, der die möglichen Werte der Variablen festlegt. Hinzu kommt die Auswahl und Konfiguration der Messadapter, die ebenfalls Teil der XML-Repräsentation des Prognosemodells sind. Sie werden im weitesten Sinne auch als Vorwissen und als Teil des Prognosemodells angesehen, da sie

auf Wissen über den Kontext und die darin enthaltenen Variablen basieren. Das folgende Listing zeigt einen Auszug eines Prognosemodells, das den Grundaufbau der XML-Repräsentation von Prognosemodellen deutlich macht.

```
<?xml version="1.0" encoding="UTF-8"?>
<mdl:predictionModel xmlns:mdl="http://contextprediction.vsis.org/
                                predictionModel"
                     xmlns:mdlma="http://contextprediction.vsis.org/
                                  predictionModel/measureAdapters"
                     xmlns:mdlme="http://contextprediction.vsis.org/
                                  predictionModel/methods">

<generalSettings timeStepLength="900000"
                 recentHistoryFrequency="3"
                 recentHistoryLength="5" />

<nominalTypeSet>
  ...
  <nominalType name="positionNominalType">
    <allowedValue value="home" />
    <allowedValue value="work" />
    <allowedValue value="way home-work" />
    <allowedValue value="supermarket" />
    <allowedValue value="park" />
    <allowedValue value="elsewhere" />
    ...
  </nominalType>
  ...
</nominalTypeSet>

<measureAdapterSet>
  ...
  <measureAdapter name="positionMeasureAdapter"
                  factoryClass="org.vsis.contextprediction.examplePhoneCall.
                                measuring.PositionDummyMeasureAdapterFactory"
                  timeTolerance="150000">
    <mdlma:measureAdapterSpecificSettings />
  </measureAdapter>
  ...
</measureAdapterSet>

<methodPreknowledgeSet>
  ...
  <methodPreknowledge
            name="positionByTimeAndPositBeforePreknowl"
            factoryClass="org.vsis.contextprediction.preknowledge.methods.
                          universal.ProbabTablePreknowlFactory"
            nominalTypeDependentVariable="positionNominalType">
    <mdlme:probabilityTableSettings>
      <property name="indepVarNominalTypes">
        <property name="nominalType" value="timeOfDayNominalType"/>
        <property name="nominalType" value="positionNominalType"/>
      </property>
```

```
50      </mdlme:probabilityTableSettings>
51    </methodPreknowledge>
52    ...
53  </methodPreknowledgeSet>
54
55  <variableSet>
56    ...
57    <variable name="position" scale="nominal" nominalType="positionNominalType"
58              measureAdapter="positionMeasureAdapter"
59              defaultLearningDistance="900000"
60              learningMethodPreknowlNoIndep="true"
61              learningMethodPreknowlWithIndep="true"
62              methodPreknowledgeNoIndep="positionNoIndepPreknowl"
63              methodPreknowledgeWithIndep="positionByTimeAndPositBeforePreknowl">
64      <parentEdge timeOffset="0" variable="timeOfDay"/>
65      <parentEdge timeOffset="1" variable="position"/>
66    </variable>
67    ...
68  </variableSet>
69
70  </mdl:predictionModel>
```

Listing 5.1: Auszug aus einem Prognosemodell in XML-Repräsentation

Die Tags `nominalType`, `measureAdapter`, `methodPreknowledge` und `variable` entsprechen den gleichnamigen Klassen in der Vorwissenschicht des Prognosesystems. Zu Beginn des XML-Dokuments befindet sich darüber hinaus der Tag `generalSettings`, der allgemeine Parameter wie die Länge eines Zeitschrittes angibt. Die Zeitschrittlänge sollte so gewählt werden, dass zur Minimierung des Ressourcenaufwands bei keiner benötigten Prognose zu viele Zeitschritte einbezogen werden müssen und dass gleichzeitig die Zeit für jede Prognose fein genug aufgelöst ist. Wenn sowohl Prognosen über nahe, als auch Prognosen über ferne Zukunft benötigt werden, kann auf zwei Prognosemodelle mit unterschiedlichen Zeitschrittlängen ausgewichen werden. Auf diesen Tag folgen die nominalen Typen. In obigem Beispiel sind eine Reihe diskreter Orte als mögliche Werte für Variablen mit dem nominalen Typen `positionNominalType` erlaubt. Dann sind die Messadapter und Methodenvorwissen definiert. Dazu muss vor allem jeweils eine Fabrik-Klasse zur Erzeugung des Messadapters bzw. des Methodenvorwissens angegeben werden, so dass das Prognosesystem sie ohne Modifizierungen bestehenden Quellcodes erzeugen kann. Über die Fabrik-Klasse hinaus können für einen Messadapter bzw. eine Methode Angaben erforderlich sein, die spezifisch für den jeweiligen Messadapter bzw. die Methode sind. Bei der Binning-Methode, die Werte mit Ratio-Skalenniveau zu Werten mit nominalem Skalenniveau diskretisiert, muss z.B. festgelegt werden, welche Wertintervalle auf welche diskreten Werte abgebildet werden. In obigem Listing sind ein Messadapter zur Messung der Position und Methodenvorwissen für die Prognose der Position aus der Zeit und aus der Position in der vorhergehenden Zeitscheibe definiert. Messadapter und Methodenvorwissen werden von den Variablen in einem Prognosemodell referenziert. Die Variablen bilden schließlich mit der Angabe ihrer Eingangskanten das Prognosenetz.

Es existiert ein dem Standard *XML Schema* des W3C folgendes XML-Schema, das die Syntax von Prognosemodellen als XML-Repräsentation spezifiziert. Dieses erlaubt auch Anwendungsentwicklern, mit XML Schema die Syntax von Angaben festzulegen, die speziell von ihren Messadaptern und Methoden benötigt werden. Die Syntax für konkrete Messadapter und Methoden wird dabei in zwei separaten Dateien definiert, die vom Hauptschema referenziert werden, so dass das bestehende Hauptschema nicht modifiziert werden muss. Anhang B zeigt das Hauptschema und Anhang C ein Beispiel für das ergänzende Schema einer konkreten Methode. Der Programmcode des Prognosesystems verwendet die XML-Schemata nicht, da der verwendete XML-Parser *kXML* für mobile Geräte dies nicht unterstützt. Die Schemata dienen jedoch der Spezifikation und erleichtern das Erstellen von Prognosemodellen.

5.1.2. Datenakquisition

Die Implementierung der Datenakquisitionsschicht entspricht der Beschreibung einer einfachen Ausgestaltung derselben in Abschnitt 4.2.4. Ein Messadapter misst auf Anforderung einen Wert. Ein Objekt vom Typ `MeasureAdapter` wird regulär beim Laden des Prognosemodells durch den `ModelLoader` erzeugt (vgl. Abbildung 5.1). Da Anwendungsentwickler neue Messadapter implementieren können, ist grundsätzlich auch eine einfache Form entfernter Datenakquisition realisierbar. Die im Rahmen dieser Arbeit entwickelte Implementierung enthält zwei universell einsetzbare Messadapter zur Messung der Zeit und der Position, die als primärer Kontext besonders wichtig sind. Die Zeit wird durch Abfragen der Systemzeit gemessen. Die Position auf der Erde wird in Form von Koordinaten bestimmt. Dazu greift der Messadapter auf die *Location API* für die Java Micro Edition (JSR-179) zurück, die die Heterogenität mobiler Geräte überwindet und die verwendete Messmethode verbirgt. Neben GPS besteht z.B. die Möglichkeit, die Zellen eines drahtlosen Netzwerks zur Positionsbestimmung zu verwenden.

5.1.3. Lernen

Das Lernen erfolgt wie in Abschnitt 4.2.3 beschrieben. Der `LearningAdministrator` startet und stoppt das Lernen für Variablen auf Wunsch der Anwendung und entsprechend den Vorgaben im Prognosemodell (vgl. Abbildung 5.1). Das Lernen erfolgt pro Methodenvorwissen der Variablen. Dazu werden Datensätze als Trainingsdaten durch den `MethodLearner` der Methode gelernt. Dieser benutzt das `MethodPreknowl`, das wiederum auf die `MethodPreknowlInstanceData` zugreift, die letztendlich beim Lernen aufgebaut und verbessert werden sollen. Die Datensätze werden regelmäßig pro Methodenvorwissen durch einen `LearnerThread` gebildet und dem `MethodLearner` übergeben. Die Werte der abhängigen Variablen und den unabhängigen Variablen werden dazu jeweils über einen `MeasureAdapter` bezogen. Ein `LearnerThread` setzt einen `DataObtainer` ein, um die Werte der unabhängigen Variablen zu prognostizieren, wenn sie nicht messbar sind, jedoch über Methoden prognostiziert werden, die Repräsentationsabbilder sind, also nur Repräsentationen in andere überführen.

5.1.4. Prognose

Der Kern der Prognose ist in der Klasse `ConnectedPredictor` umgesetzt. Sie führt sowohl die Generierung des relevanten Ausschnittes des aufgefalteten Prognosenetzes aus dem Prognosenetz, als auch die auf dem relevanten Ausschnitt des aufgefalteten Prognosenetzes basierende Prognose durch. Ein `ConnectedPredictor` prognostiziert den Wert einer Variablen $V_{i,j}$ des aufgefalteten Prognosenetzes und benutzt dazu einen zur Variablen gehörenden `MethodPredictor` sowie die `ConnectedPredictor`-Objekte der unabhängigen Variablen.

Die Generierung des relevanten Ausschnittes des aufgefalteten Prognosenetzes findet bei der Erzeugung eines `ConnectedPredictor`-Objekts statt. Dabei fragt dieses die Eingangskanten seiner Variablen V_i im Prognosenetz in der Vorwissenschicht ab und erzeugt für diese rekursiv `ConnectedPredictor`-Objekte (vgl. Abschnitt 4.2.2). Eine Variable V_i des Prognosenetzes wird in der Vorwissenschicht durch ein `Variable`-Objekt repräsentiert. Eine Variable $V_{i,j}$ des aufgefalteten Prognosenetzes entspricht in der Implementierung einem `VariableInTime`-Objekt.

Bei der Prognose wird das in Abschnitt 4.2.2 ausgearbeitete Grundgerüst für einen Algorithmus verwendet, bei dem rekursiv auf die `ConnectedPredictor`-Objekte der unabhängigen Variablen zurückgegriffen wird. Für die Prognose des Wertes der abhängigen Variablen aus den Werten der unabhängigen Variablen benutzt der `ConnectedPredictor` einer Variablen den entsprechenden `MethodPredictor` der Variablen.

Der `PredictionInitiator` realisiert über einen `ConnectedPredictor` hinaus die beiden Prognosemodi der Zustandsprognose zur Prognose eines Zustandes zu einem gegebenen Zeitpunkt und der Zeitprognose zur Prognose der Zeit, zu der ein gegebener Zustand eintritt. Er wandelt außerdem Zeitangaben in Zeitscheiben um. Der `PredictionInitiator` kann `GivenInformation` über die Gegenwart oder jüngere Vergangenheit von der Anwendung als Ausgangspunkt für eine Prognose entgegen nehmen. Ein `GivenInformation`-Objekt enthält `VariableAssignment`-Objekte, die `VariableInTime`-Objekten jeweils einen `VarValue` zuordnen. Regulär nimmt der `PredictionInitiator` jedoch die `GivenInformation` nicht von der Anwendung entgegen, sondern fragt sie bei der `RecentHistory` ab. Diese besitzt einen eigenen Thread und lässt regelmäßig von `MeasureAdapter`-Objekten die Werte aller messbarer Variablen messen.

5.1.5. Methoden

Die in Abschnitt 4.2.5 als Referenzkonfiguration gewählten Methoden sind alle im Rahmen dieser Arbeit implementiert. Dabei wird großer Wert auf Effizienz gelegt. Der Kern vieler Methoden liegt in effizienten Datenstrukturen, die zum Teil schon von der API des Personal Profile der Java Micro Edition bereitgestellt werden und zum Teil selbst implementiert sind.

Für die Binning-Methode wird z.B. eine `TreeMap` aus der API verwendet, um effizient das Intervall zu finden, in dem ein Wert liegt. Für die empirische Verteilungsfunktion

als Methode zur Prognose ohne Werte unabhängiger Variablen ist dagegen ein binärer Suchbaum im Rahmen dieser Arbeit implementiert. Dieser speichert in jedem Knoten die Anzahl an Nachkommen auf der linken und auf der rechten Seite und ermöglicht so den effizienten Zugriff auf das i-te Element im Baum mit besserer als linearer Zeitkomplexität. Dies ermöglicht das für die Prognose notwendige, effiziente Wählen von Zufallszahlen, die der Verteilung entsprechen.

Der Speicherbedarf der meisten Methoden ist moderat. Lediglich Wahrscheinlichkeitstabellen benötigen bei zu vielen unabhängigen Variablen oder zu großen Wertemengen der Variablen viel Speicherplatz, was beim Erstellen von Prognosemodellen zu beachten ist.

Wenn Anwendungsentwickler über die bestehende Referenzkonfiguration hinaus Methoden implementieren möchten, müssen sie zu einer Reihe abstrakter Klassen bzw. Interfaces jeweils eine Klasse schreiben, die von der Klasse erbt bzw. das Interface implementiert. Dabei muss in der Typhierarchie von den tiefsten Typen ausgegangen werden. Es ist also z.B. keine direkte Unterklasse von `MethodPreknowl`, sondern von `NominalMethodPreknowl` zu schreiben. Es folgenden die abstrakten Klassen und Interfaces.

- `MethodPreknowl`
- `MethodPreknowlFactory`
- `MethodPreknowlInstanceData`
- `MethodLearner`
- `MethodLearnerFactory`
- `MethodPredictor`
- `MethodPredictorFactory`

Die Fabrik-Klassen werden verwendet, um zu erreichen, dass bei der Implementierung neuer Methoden kein bestehender Quellcode des Prognosesystems modifiziert werden muss. Von den Fabrik-Klassen einer Methode ist im Gegensatz zu den weniger bedeutsamen Klassen vom Typ `MethodLearnerFactory` und `MethodPredictorFactory` die Klasse vom Typ `MethodPreknowlFactory` am komplexesten, da sie von der Methode benötigte methodenspezifische Angaben in der XML-Repräsentation des Prognosemodells in abstrakter Form verarbeiten muss. Die Funktionen der restlichen abstrakten Klassen und Interfaces in obiger Auflistung wurden bereits erläutert.

Weiterführend bietet es sich an, das Implementieren einfacher Methoden zu erleichtern. Die Mittelwert-Methode zur Ergebnisintegration beim Coprocessing benötigt z.B. im Grunde kein Vorwissen und kein Lernen. Eine Möglichkeit zur Vereinfachung der Implementierung solcher Methoden liegt darin, (abstrakte) Klassen bereitzustellen, die von den oben aufgelisteten Klassen erben bzw. die Interfaces implementieren und sich für bestimmte Zwecke eignen. Man kann z.B. einen `MethodLearner` und eine `MethodLearnerFactory` implementieren, die im Grunde keine Funktionalität aufweisen und von Methoden wie der Mittelwert-Methode verwendet werden können.

5.2. Aspekte des Gesamtsystems

Dieser Abschnitt geht auf Gesichtspunkte ein, die nicht isoliert für einzelne Schichten des Prognosesystems betrachtet werden können. Diese betreffen zum Teil die Entwicklung des Systems und zum Teil auch dessen Einsatz.

5.2.1. Synchronisation

Im Prognosesystem kommen mehrere Threads zum Einsatz, die synchronisiert werden müssen. Es kann unterschieden werden zwischen Threads, die im Prognosesystem erstellt und gestartet werden, und Threads, die von der Anwendung stammen. Innerhalb des Prognosesystems wird ein Thread erstellt, der die `RecentHistory` aktualisiert. Zusätzlich wird beim Lernen pro Methodenvorwissen ein Thread erstellt. Threads der Anwendung dringen in erster Linie bei Prognosen in das System ein. Die Threads erfüllen unterschiedliche Aufgaben, zwischen denen nur schwache zeitliche Abhängigkeiten bestehen. Sie greifen jedoch auf gemeinsame Ressourcen zu.

Der Thread der `RecentHistory` und die Lernthreads greifen alle auf `MeasureAdapter` zu. Da der Synchronisationsbedarf abhängig von den Messadaptern ist, liegt die Synchronisation an dieser Stelle im Verantwortungsbereich der Messadapter.

Die zweite gemeinsame Ressource, auf die unterschiedliche Threads zugreifen, ist das Vorwissen. Für das Prognosemodell als Teil des Vorwissens besteht wegen dessen Unveränderbarkeit kein Synchronisationsbedarf zur Wahrung der Datenkonsistenz. Auf `MethodPreknowlInstanceData` erfolgen im Gegensatz dazu auch Schreibzugriffe, so dass durch ungünstige Verschränkungen von Zugriffen inkonsistente Daten gelesen werden können. Daher werden pro Methodenvorwissen Sperren verwendet. Ein `MethodLearner`, der in einem der Lernthreads ausgeführt wird, sperrt dann das Methodenvorwissen für den Zeitraum des Lernens. Dies ist bereits in der abstrakten Klasse implementiert. Andere Lernthreads werden dadurch nicht beeinträchtigt, da sie auf anderes Methodenvorwissen zugreifen. Die Sperren bewirken jedoch einen wechselseitigen Ausschluss zwischen Lernthreads und prognostizierenden Threads der Anwendung. Wenn eine Variable für eine Prognose nicht relevant ist, wird das Lernen für das Methodenvorwissen dieser Variablen aber nicht beeinträchtigt.

Verklemmungen beim konkurrierenden Zugriff von Lernthreads und prognostizierenden Threads der Anwendung auf das Methodenvorwissen sind ausgeschlossen, da Lernthreads nie eine weitere Sperre anfordern, wenn sie schon eine halten. Somit kann es nicht passieren, dass ein Thread auf einen Lernthread wartet, weil der Lernthread eine Sperre hält, und der Lernthread wiederum seinerseits auf einen Thread wartet, weil er eine weitere Sperre anfordert. Deshalb kann ein Lernthread in einem Wartegraphen nur entweder eine Eingangskante oder eine Ausgangskante besitzen, aber nie eine Ein- und eine Ausgangskante, so dass keine Zyklen möglich sind, an denen ein Lernthread beteiligt ist. Prognostizierende Threads der Anwendung fordern jedoch mehrere Sperren an, da sie auf Methodenvorwissen mehrerer Variablen zugreifen. Um Verklemmungen zwischen prognostizierenden Threads vorzubeugen, wird durch eine zentrale Sperre im Prognosesystem verhindert, dass mehrere Threads gleichzeitig prognostizieren können.

5.2.2. Tests

Es existieren eine Reihe von JUnit-Tests zum Prognosesystem, die einzelne Teile des Systems testen. Darüber hinaus ist das Gesamtsystem anhand einer Umsetzung von Beispiel 1.2 über die Prognose der Nutzung eines Mobiltelefons erprobt. Das Testen des Prognosesystems gestaltet sich jedoch aufgrund von drei Problemen relativ schwierig.

Das erste Problem liegt darin, dass das Prognoseverfahren nach Abschnitt 4.2.2 vorsieht, dass die Prognosen von Methoden von Zufallszahlen abhängen. Daher können Tests mit einer gewissen Wahrscheinlichkeit fehlschlagen, obwohl das System korrekt arbeitet. Diesem Problem wird durch hohe Anzahlen an Prognoserunden entgegengewirkt.

Das zweite Problem entsteht dadurch, dass insbesondere zur Prognose des Gesamtsystems Daten von den Messadaptern gemessen können werden müssen. Wenn ein schnell durchführbarer, automatischer Test ausgeführt werden soll, mit dem man nach Änderungen am System augenblicklich testen kann, ob dadurch Fehler im System entstanden sind, können jedoch in der Regel keine geeigneten, realen Daten aktuell gemessen werden. Wenn man sich beim Testen z.B. nur an einem Ort befindet, macht das Messen der Position kaum Sinn. Dieses Problem ist für die Erprobung des Gesamtsystems anhand von Beispiel 1.2 so gelöst, dass Dummy-Messadapter eingesetzt werden, deren Messungen im Abrufen jeweils eines Wertes aus einer Historie fiktiver Daten bestehen.

Daran anschließend ergibt sich das folgende, dritte Problem. Wenn die Historie Daten mehrerer Tage beinhaltet, muss die Zeit in der Historie beschleunigt werden, so dass eine Sekunde realer Zeit beispielsweise einer Stunde in der Historie entspricht. Im Prognosemodell muss dann eine kürzere Zeitschrittlänge gewählt werden und bei Messungen mit Dummy-Messadaptern muss eine Umrechnung zwischen der realen Zeit und der Zeit in der Historie erfolgen. Außerdem wird eine spezielle Variante der Zeitinkrementierer-Methode benutzt, die der Prognose des zukünftigen Zeitpunktes dient, wenn dieser im Prognosemodell als Variable verwendet wird.

5.2.3. Einsatz

Möglichkeiten wie das automatische Messen von Daten durch Messadapter ohne Beteiligung der Anwendung wirken für Tests eher erschwerend, erleichtern jedoch den produktiven Einsatz des Prognosesystems in einer Anwendung. Wenn das Prognosemodell erstellt und das Prognosesystem eingerichtet ist, beschränkt sich die Nutzung desselben in vielen Fällen auf wenige Zeilen Programmcode in der Anwendung, wie das folgende Listing als Beispiel für die Nutzung des Prognosesystems zeigt.

```
1 // start application
2 PredictionSystem predictionSystem = new PredictionSystem(
3         predictionModelFile, true, true, new ConsoleLogAdapter() );
4 predictionSystem.loadInstanceData(instanceDataFile);
5 predictionSystem.startLearning();
6 predictionSystem.startUpdatingRecentHistory();
7
8 ...
9
```

```
// predict value of a variable after 60000 milliseconds with 300 iterations
FrequencyNominalVarDistrib predictedDistribution = (FrequencyNominalVarDistrib)
        predictionSystem.predictState("variableName",
                                      new PointOfRealTime(60000), 300);
double variableValueProbability = predictedDistribution.getProbabOfVarValue(
        predictionSystem.getNominalValue("nominalTypeName", "variableValue"));

...

// stop application
predictionSystem.stopUpdatingRecentHistory();
predictionSystem.stopLearning();
predictionSystem.storeInstanceData(instanceDataFile);
```

Listing 5.2: Beispiel für Nutzung des Prognosesystems in einer Anwendung

Das Listing beginnt mit dem Erstellen des Prognosesystems, das das Laden des Prognosemodells einschließt, dem Laden des Vorwissens auf der Instanzebene und dem Starten des automatischen Lernens sowie des Aktualisierens der kurzen Historie mit potentiellen gegebenen Informationen für Prognosen. Dann folgt die Prognose der Verteilung der möglichen Werte einer Variablen durch Zustandsprognose. Vor dem Beenden der Anwendung werden schließlich alle Threads gestoppt und das durch adaptives Lernen angepasste Vorwissen auf der Instanzebene in einer Datei gespeichert.

5.3. Einsatz in der DEMAC-Middleware

Ein Ziel dieser Arbeit liegt darin, für die DEMAC-Middleware die Möglichkeit der Prognose der Verfügbarkeit von Diensten für ein Gerät bereitzustellen (vgl. Beispiel 1.1 und Abschnitt 2.3.7). Die DEMAC-Middleware realisiert die Konzepte kontextbasierter Kooperation und mobiler Prozesse, indem sie langlebige, komplexe Aufgaben in mobilen Umgebungen in Form von Prozessen kooperativ ausführt [Kun08] ↑. Dabei nutzt sie die unterschiedlichen Potentiale mehrerer Geräte, die in den Ausführungskontexten der Geräte Ausdruck finden, indem sie die Migration von Prozessen unterstützt [Kun08] ↑. Wichtige Entscheidungsgrundlage bei der Wahl eines Gerätes als Migrationsziel bilden daher die Ausführungskontexte der Geräte, die die Ausführung bestimmter als Dienste angesehener Teilaufgaben erlauben und die nicht-funktionalen Rahmenbedingungen bilden [Kun08] ↑. Ein wichtiger Aspekt des Ausführungskontextes eines Gerätes als Grundlage zur Planung von Migrationen ist also, auf welche Dienste es lokal oder entfernt zugreifen kann. Durch eine Prognose der Dienstverfügbarkeit wird der hohen Dynamik mobiler Umgebungen Rechnung getragen, die zu Änderungen der Verfügbarkeit von Diensten für Geräte während der Ausführung eines langlebigen Prozesses führen kann ↑ ↑ [Kun08].

Konkret ist für das DEMAC-Projekt momentan eine Prognose der Verfügbarkeit eines Dienstes für Geräte als potentielle Migrationsziele erwünscht, wenn der Dienst aktuell für keines dieser Geräte verfügbar ist. In diesem Abschnitt wird zur Erfüllung dieser Aufgabe die Rolle eines Anwendungsentwicklers eingenommen, der das Prognosesystem

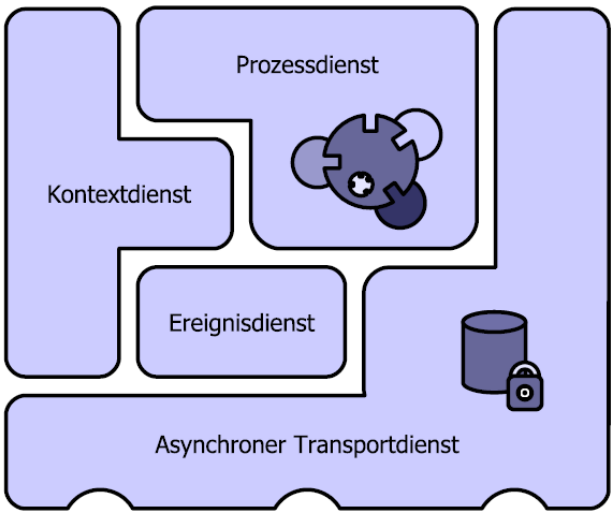

Abbildung 5.2.: Ausschnitt aus der Architektur der DEMAC-Middleware (übernommen aus [Kun08, S. 195])

für eine Anwendung benutzen möchte. Die Rolle der Anwendung nimmt in diesem Fall die DEMAC-Middleware ein bzw. im engeren Sinne deren Teil, der in der Architektur oberhalb des Prognosesystems angesiedelt ist und dieses benutzt. Dabei handelt es sich um den sogenannten Prozessdienst, der für die Ausführung und Migration von Prozessen verantwortlich ist (vgl. Abbildung 5.2). Bestandteile der Architektur der DEMAC-Middleware wie der Prozessdienst werden selbst als Dienste angesehen. Die Prognose der Verfügbarkeit von Diensten soll ebenfalls als ein Dienst unterhalb des Prozessdienstes platziert werden. Das Prognosesystem wird darin als JAR-Datei eingebunden.

Dieser Abschnitt beginnt mit dem obligatorischen Schritt der Erstellung eines Prognosemodells und fährt dann mit der Realisierung von Funktionalität fort, die über das Basissystem hinausgehend für die Prognose der Verfügbarkeit von Diensten in der DEMAC-Middleware erforderlich ist.

5.3.1. Erstellung des Prognosemodells

Dieser Abschnitt folgt dem systematischen Vorgehen zur Erstellung von Prognosemodellen in Abschnitt 4.3.1.

Variablen

Den ersten Schritt der Erstellung eines Prognosemodells stellt die Auswahl der Variablen dar (vgl. Abbildung 4.8). Wenn man berücksichtigt, dass das Konzept mobiler Prozesse im Bereich der Vernetzung den Einsatz von Ad-Hoc-Netzwerken in den Mittelpunkt stellt, hängt die Verfügbarkeit eines Dienstes für ein Gerät davon ab, ob sich Geräte in der Umgebung befinden, die diesen anbieten. Weitere Einflussfaktoren sind, ob Dienstanbieter und potentielle Dienstnutzer eine gemeinsame Netzwerktechnik wie z.B. Bluetooth unterstützen und ob der Dienst über eine Verteilungstechnik wie z.B. Webservices angeboten wird, die der potentielle Dienstnutzer unterstützt. Für das Prognosemodell werden

neben der Verfügbarkeit des Dienstes als binäre Variable die Tageszeit und die Position des Gerätes gewählt, da diese als primärer Kontext eine besondere Bedeutung besitzen, messbar sind und weil es vom Ort und der Position des Gerätes abhängt, ob sich ein Gerät in der Umgebung befindet, das den Dienst anbietet. An manchen Orten und zu manchen Zeiten wird ein Dienst häufiger als an anderen Orten und zu anderen Zeiten angeboten. Es ist z.B. möglich, dass ein Dienst immer auf der Arbeit zu bestimmten Zeiten genutzt werden kann, weil sich eine Person mit einem Gerät, das diesen Dienst anbietet, regelmäßig zu diesen Zeiten dort befindet. Manche Dienste wie z.B. ein Druckdienst werden sogar ausschließlich an bestimmten Orten angeboten. Als weitere Variable die Netzgröße verwendet. Damit ist die Anzahl an Geräten im von Netzwerktechniken unabhängigen Overlay-Netzwerk gemeint, das durch den Transportdienst der DEMAC-Middleware gebildet wird [Kun08, S. 198] (vgl. Abbildung 5.2). Bei vielen Geräten im Netzwerk, z.B. in der U-Bahn, ist eher als z.B. im Wald davon auszugehen, dass eines der Geräte in der Umgebung den Dienst anbietet.

Zusammenhangsstruktur

Der zweite Schritt der Erstellung eines Prognosemodells besteht in der Ermittlung der Zusammenhangsstruktur. Da die Zusammenhangsstruktur nicht sehr kompliziert ist, werden keine automatischen Ansätze zur Analyse eingesetzt. Die Dienstverfügbarkeit zu einem bestimmten Zeitpunkt in der Zukunft hängt kausal von der Tageszeit, der Position und der Netzgröße zu diesem Zeitpunkt ab. Darüber hinaus wird die Position zu einem bestimmten Zeitpunkt von der Tageszeit zu diesem Zeitpunkt und der Position davor beeinflusst.

Prognosestruktur und Methoden für Dienstverfügbarkeit

Die Ableitung der Prognosestruktur und die parallele Auswahl und Implementierung von Methoden als weitere Schritte gestalten sich etwas komplexer. Abbildung 5.3 zeigt das Prognosenetz als Ergebnis der Ableitung der Prognosestruktur. Diese entspricht in Wesentlichen der Zusammenhangsstruktur, da nur die Dienstverfügbarkeit entlang der Richtung der kausalen Zusammenhänge zu prognostizieren ist. Die Prognosestruktur stellt eine Verfeinerung der Zusammenhangsstruktur dar, die stark durch den Einsatz von Coprocessing gelenkt ist. Dabei wird die Variante des Coprocessing gewählt, bei der jede der parallel eingesetzten Methoden nur einen Teil der unabhängigen Variablen berücksichtigt, die bei einer Prognose durch eine einzelne Methode ohne Coprocessing verwendet würden (vgl. Abbildung 4.9). Das macht deshalb Sinn, weil Wahrscheinlichkeitstabellen als einzige universelle Methode der Referenzkonfiguration für nominales Skalenniveau bei zu vielen unabhängigen Variablen viel Speicherplatz für das Methodenvorwissen auf der Instanzebene benötigen (vgl. Abschnitt 4.2.5).

Wahrscheinlichkeitstabellen werden eingesetzt zur Prognose der *Dienstverfügbarkeit nach Netzgröße*, der *Dienstverfügbarkeit nach Tageszeit*, der *Dienstverfügbarkeit nach Ort* und der *Dienstverfügbarkeit nach Tageszeit und Ort* (vgl. Abbildung 5.3). Die Prognose der *Dienstverfügbarkeit nach Tageszeit und Ort* dominiert dabei wegen der Anzahl unab-

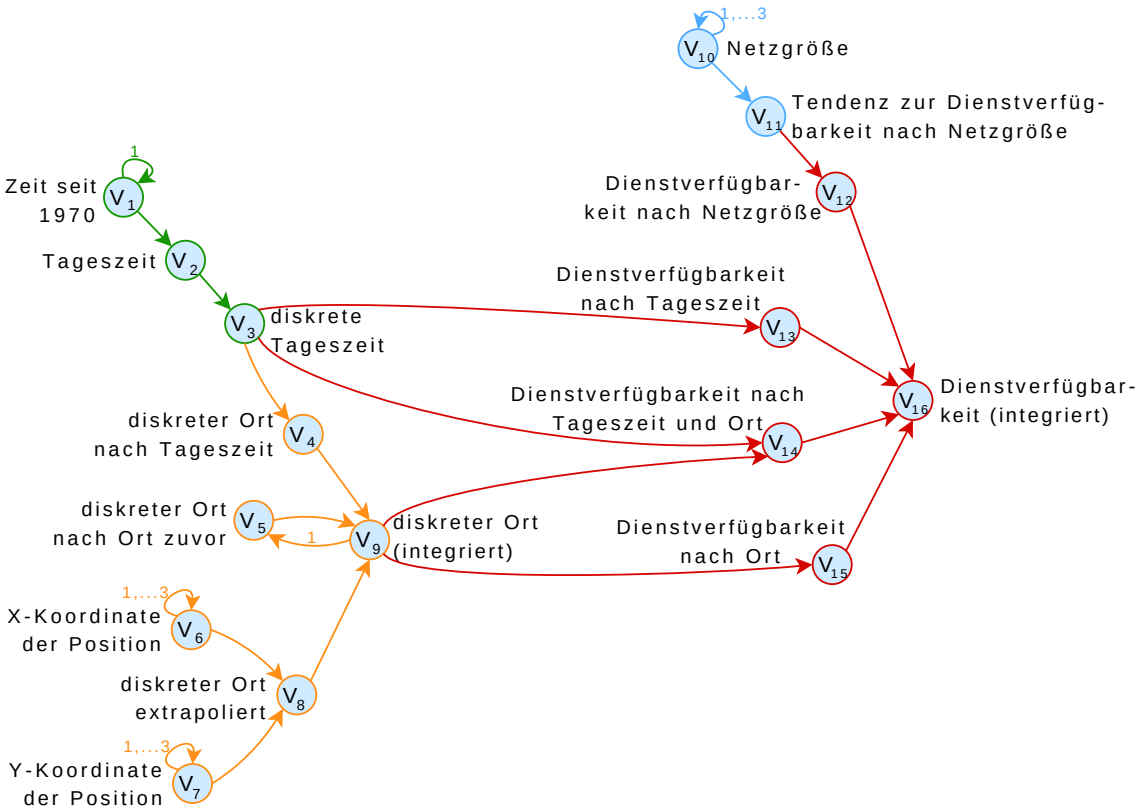

Abbildung 5.3.: Prognosenetz für die Prognose der Dienstverfügbarkeit in der DEMAC-Middleware

hängiger Variablen den Speicherbedarf. Das Entfernen dieser Prognose stellt eine Möglichkeit der Skalierung dar, durch die das Prognosemodell auch auf Geräten mit nur recht geringer Speicherkapazität eingesetzt werden kann. Die Prognose der *Dienstverfügbarkeit nach Netzgröße* basiert auf Regression, da diese recht ressourcenschonend ist, die Netzgröße Ratio-Skalenniveau aufweist, und zu vermuten ist, dass die Chance auf Verfügbarkeit des Dienstes in etwa linear mit der Netzgröße wächst. Da Regression einen Wert mit Ratio-Skalenniveau als Ergebnis bestimmt, die Dienstverfügbarkeit jedoch kein Ratio-Skalenniveau besitzt, wird das Ergebnis als Tendenz zur Dienstverfügbarkeit interpretiert und mittels der Binning-Methode auf einen binären Wert abgebildet. Die Integration der Prognoseergebnisse für die Dienstverfügbarkeit erfolgt durch die Mehrheitsentscheid-Methode, da sie nominales Skalenniveau unterstützt.

Prognosestruktur und Methoden für Netzgröße und Zeit

Da die Dienstverfügbarkeit zu einem bestimmten Zeitpunkt aus der Tageszeit, dem Ort und der Netzgröße zu diesem Zeitpunkt prognostiziert wird, muss für diese eine Zukunftsprognose durchgeführt werden, um die zukünftigen Werte aus den gegenwärtigen und vergangenen zu bestimmen. Für die Netzgröße erfolgt dies durch die Polynomextrapolation-Methode, die sich wegen des Ratio-Skalenniveaus der Netzgröße als Methode mit geringem Ressourcenbedarf anbietet. Weiterführend ist auch das Einbe-

ziehen von Zeit und Ort zur Prognose der Netzgröße denkbar. Die zukünftige Zeit wird in ihrer Darstellung als Anzahl Millisekunden seit 1970 durch die Zeitinkrementierer-Methode ermittelt. Zur Bestimmung der Tageszeit kommt die Zeitperiodisierer-Methode zum Einsatz. Durch die Nutzung der Tageszeit werden periodische Zusammenhänge mit einem Tag als Periode unterstützt. Es kann so z.B. berücksichtigt werden, dass nachts die meisten Menschen schlafen, so dass an den meisten Orten kaum mobile Geräte als Anbieter von Diensten in der Umgebung zu erwarten sind. Schließlich wird die Tageszeit durch Binning diskretisiert, so dass sie als unabhängige Variable für Wahrscheinlichkeitstabellen verwendet werden kann. Wahrscheinlichkeitstabellen eigenen sich in Verbindung mit der Zeit gut, da es sich um eine nichtparametrische Methode handelt, die so gut wie keine Voraussetzungen bezüglich der zu lernenden Zusammenhänge macht. Die starke Annahme von z.B. Linearität, die bei Regression gemacht wird, wäre hier unangebracht, denn man kann nicht grundsätzlich davon ausgehen, dass die Tendenz zur Verfügbarkeit eines Dienstes im Laufe des Tages immer weiter steigt oder sinkt.

Prognosestruktur und Methoden für Position / Ort

Zur Prognose der zukünftigen Position werden drei Methoden parallel angewendet. Die erste basiert auf der Darstellung der Position auf der Erde durch Koordinaten als Variablen mit Ratio-Skalenniveau. Die X- und die Y-Koordinate werden separat über die Polynomextrapolation-Methode prognostiziert. So werden Trends in der Bewegung erfasst, ohne dass dafür Vorwissen auf der Instanzebene gesammelt sein muss. Eine Bewegung in Richtung Süden wird z.B. durch diesen Ansatz als Bewegung in Richtung Süden in der Zukunft fortgesetzt. Die weiteren zwei Methoden, die zur Verbesserung der Prognosegenauigkeit eingesetzt werden, basieren auf der Darstellung der Position durch diskrete Orte wie z.B. „zu Hause“, „auf der Arbeit“ oder „im Supermarkt“. Es handelt sich um Wahrscheinlichkeitstabellen. Die erste prognostiziert den Ort in Abhängigkeit von der Tageszeit und nutzt damit periodische Zusammenhänge. Die zweite verwendet den Ort in der vorhergehenden Zeitscheibe als unabhängige Variable. Dies entspricht einer Markovkette und erfasst sequentielle Zusammenhänge. Regelmäßig wiederkehrende Abfolgen von Orten wie z.B. der Weg zur Arbeit morgens können damit abgebildet werden.

Um die Ergebnisintegration der unterschiedlichen Prognosen für den Ort / die Position durchführen zu können, wird die prognostizierte zukünftige Position in Koordinatendarstellung als Prognoseergebnis der Polynomextrapolation-Methode mit der 2D-Binning-Methode in einen diskreten Ort umgewandelt. Die Ergebnisintegration wird dann von der Mehrheitsentscheid-Methode durchgeführt.

Eine Umwandlung der Position als Koordinaten in einen diskreten Ort ist auch für die anderen beiden Methoden erforderlich. Sie benötigen den momentanen diskreten Ort zum Lernen und als gegebene Informationen bei der Prognose, was allerdings mit einem Problem verbunden ist. Es sei als Beispiel angenommen, dass die Position in Koordinatendarstellung zum Zeitpunkt 0 als gegenwärtigem Zeitpunkt durch Messung als gegebene Informationen bekannt ist. Wenn nun eine Prognose durchgeführt wird, soll bei der Methode für die Prognose des Ortes aus dem Ort in der vorhergehenden Zeitscheibe

natürlich die für den Zeitpunkt 0 gegebene Position verwendet werden, anstatt dass der Ort zum Zeitpunkt 0 wiederum aus früheren Werten prognostiziert wird. Grundsätzlich wird bei Prognosen ein gegebener Wert für eine Variable verwendet, anstatt den Wert der Variablen zu prognostizieren. Wenn die Position als Koordinaten und der Ort als diskreter Wert als unterschiedliche Variablen im Prognosemodell dargestellt sind und nur die Position als Koordinaten als gegebene Informationen gemessen wird, hilft dies jedoch nicht (vgl. auch Abschnitt 4.2.3). Im Fall der Prognose der Dienstverfügbarkeit für das DEMAC-Projekt ist das Problem so gelöst, dass neben dem Messadapter für die Position als Koordinaten auch ein Messadapter für den Ort als diskrete Größe existiert, der die vom anderen Messadapter ermittelte Position in einen diskreten Ort umwandelt. Die Vermeidung einer doppelten Implementierung dieser Umwandlung erfordert allerdings, dass der Kern der 2D-Binning-Methode zur gemeinsamen Verwendung durch diese und den Messadapter als separate Klasse implementiert ist.

Unabhängig davon weist der Ansatz der Verwendung diskreter Orte für Prognosen zunächst den Nachteil auf, dass eine Zuordnung von der Position als Koordinaten zu diskreten Orten festgelegt werden muss. Dabei wird vorausgesetzt, dass der Ort als diskreter Wert nicht mit entsprechenden Techniken direkt gemessen wird, z.B. in einem Firmengebäude. Diese Zuordnung muss momentan durch die Anwendungsentwickler festgelegt werden, so dass sich alle Nutzer an den gleichen Orten aufhalten müssen. Dieser Nachteil lässt sich aber überwinden, indem die aktuelle Position eines Nutzers als Koordinaten zur Laufzeit regelmäßig gespeichert und ein Clustering der Positionen durchgeführt wird. Die Cluster können dann als diskrete Orte verwendet werden. Mayrhofer hat mit seinem Verfahren gezeigt, dass grundsätzlich sogar adaptives Clustering möglich ist, bei dem die Cluster kontinuierlich zur Laufzeit angepasst werden [May04].

Abschlussbemerkungen und verbleibende Schritte

Das entwickelte Prognosenetz verdeutlicht erneut den Sinn der Erweiterung zum verbesserten Umgang mit Unsicherheit für das Coprocessing (vgl. Abschnitt 4.4.3). So macht es z.B. Sinn, den Ort über den Ort in der vorhergehenden Zeitscheibe zu prognostizieren, wenn der Nutzer sich gerade auf einem häufigen beschrittenen Pfad befindet, also die Variationsunsicherheit für den nächsten Schritt eher gering ist. An unbekannten Orten sollte das Ergebnis der Extrapolation verwendet werden. Eine Berücksichtigung, ob einer Position als Koordinaten ein diskreter Ort zugeordnet ist, gelingt auch ohne die Erweiterung zum verbesserten Umgang mit Unsicherheit. Dazu ist ein diskreter Ort „unbekannt“ eingeführt und die Mehrheitsentscheid-Methode so modifiziert, dass ein Prognoseergebnis „unbekannt“ nicht in den Mehrheitsentscheid eingeht.

Nachdem das Prognosenetz erstellt ist, verbleiben im Vorgehen zur Erstellung eines Prognosemodells in Abschnitt 4.3.1 die Festlegung des Methodenvorwissens, das Einbringen von a priori vorhandenem Vorwissen und die Validierung. A priori vorhandenes Wissen wird jedoch in diesem Fall keines eingebracht und eine Validierung erfolgt nicht im Rahmen dieses Kapitels. Die Festlegung des Methodenvorwissens wird an dieser Stelle nicht ausgeführt, da es dabei um Details geht. Das vollständige Prognosemodell ist jedoch in Anhang D in XML-Repräsentation zu finden.

5.3.2. Realisierung zusätzlich benötigter Funktionalität

Zur Realisierung zusätzlich benötigter Funktionalität kommen zunächst anwendungsspezifische Methoden und Messadapter in Frage. Durch den Einsatz des Prognosesystems motivierte, auch für andere Anwendungen verwendbare Methoden und Messadapter sind als fester Bestandteil in das Prognosesystem und die Referenzkonfiguration verwendeter Methoden eingeflossen und werden daher in diesem Abschnitt nicht erneut aufgegriffen. Lediglich für die Ermittlung der Netzgröße ist ein einfacher anwendungsspezifischer Messadapter implementiert, der die Netzgröße beim Transportdienst der DEMAC-Middleware abfragt. Von der 2D-Binning-Methode wird aus den oben genannten Gründen eine variierte Version verwendet.

Die Aufgabe der Prognose der Dienstverfügbarkeit für das DEMAC-Projekt stellt jedoch auch Anforderungen, die nicht allein durch das implementierte Basissystem und zusätzliche Methoden und Messadapter erfüllt werden. Diese Anforderungen können in drei Bereiche unterteilt werden, die im Verlauf dieses Abschnittes genauer erörtert werden. Die Anforderungen der ersten zwei dieser Bereiche werden durch die Erweiterungen des Prognosesystems um Verteilung und Netzfragmentinstanzen erfüllt. Das unterstreicht die Bedeutung dieser Erweiterungen. Da die Erweiterungen jedoch nicht im Rahmen dieser Arbeit implementiert sind, werden die benötigten Funktionalitäten auf der Anwendungsebene, also in der Architektur oberhalb des Prognosesystems realisiert.

Verteilung

Der erste Bereich besonderer Anforderungen betrifft die Verteilung. Wenn sich ein Prozess auf einem Gerät befindet und dieses plant, zu welchem Gerät der Prozess migrieren soll, muss es entfernte Geräte einbeziehen. Es benötigt für jedes Gerät, das ein mögliches Migrationsziel darstellt, eine Prognose über die Dienstverfügbarkeit für das Gerät. Da die Verfügbarkeit eines Dienstes für ein Gerät hauptsächlich von diesem Gerät abhängt, bietet es sich an, dass jedes Gerät im Grundsatz die Verfügbarkeit von Diensten für sich selbst übernimmt. Ein Gerät, das die Migration eines Prozesses plant, muss daher die möglichen Migrationsziele kontaktieren, so dass die Art der Verteilung letztendlich auf eine entfernte Nutzung von Prognosen hinausläuft (vgl. Abschnitt 4.4.1). Die Umsetzung einer solchen entfernten Nutzung von Prognosen ist unabhängig von der Existenz einer entsprechenden Erweiterung des Prognosesystems relativ anwendungsspezifisch. Das ergibt sich aus der Voraussetzung in Abschnitt 2.5.8, dass in der Anwendung eingesetzte, bestehende Ansätze und Techniken zur Verteilung von den Anwendungsentwicklern auch auf das Prognosesystem angewendet werden, so dass es zu keinem konzeptionellen oder technologischen Bruch kommt.

Es kommt weiterführend auch in Frage, dass an der Gesamtaufgabe der Prognose der Verfügbarkeit eines Dienstes für ein Gerät auch andere Geräte als das Gerät selbst beteiligt werden, denn die Verfügbarkeit des Dienstes hängt nicht ausschließlich von dem Gerät selbst ab. Es kommt z.B. nicht nur darauf an, wo sich das Gerät als Dienstnutzer befindet, sondern auch, wo sich mögliche Dienstanbieter befinden. So könnte ein Dienstnutzer z.B. von Dienstanbietern in der Umgebung eine Prognose darüber abfra-

gen, wie sich ihre Position in der Zukunft ändern wird. Dieser Ansatz ist jedoch dadurch beschränkt, dass nicht anwesende, später hinzu stoßende Dienstanbieter nicht berücksichtigt werden können. Eine weitere Möglichkeit besteht darin, dass Geräte zwar allein ihre zukünftige Position und Netzgröße prognostizieren, aber zusätzlich Vorwissen darüber mit anderen Geräten austauschen, an welchen Orten, zu welchen Zeiten und bei welchen Netzgrößen im Allgemeinen bestimmte Dienste verfügbar sind. Das entspricht einem Austausch von Vorwissen, das den rechten Teil des Prognosenetzes in Abbildung 5.3 betrifft. Diese Möglichkeit ist jedoch mit den Einschränkungen verbunden, dass die Geräte sich möglicherweise an unterschiedlichen diskreten Orten häufig aufhalten und dass eine Integration des Methodenvorwissens eines entfernten Gerätes in das lokale nicht bei allen Methoden möglich ist. Wegen dieser Limitierungen beschränkt sich diese Arbeit darauf, dass jedes Gerät Prognosen über die Verfügbarkeit von Diensten für sich selbst auch selbst durchführt.

Unterstützung mehrerer Dienste

Der zweite große Anforderungsbereich neben der Verteilung liegt darin, dass Prognosen über die Verfügbarkeit mehrerer Dienste statt nur eines Dienstes benötigt werden. Es existieren also mehrere Dienstinstanzen wie z.B. ein Druckdienst, ein Verkehrsdienst und ein Nachrichtendienst zu dem allgemeinen Konzept „Dienst“ in der Realität. Für jede Dienstinstanz muss eigenes Vorwissen erlernt werden, denn ein Druckdienst wird in der Regel z.B. nicht genau an den gleichen Orten wie ein Verkehrsdienst anzutreffen sein.

Der hier verwendete Begriff einer Dienstinstanz, der sich aus Abschnitt 4.4.2 ergibt, entspricht in der Terminologie von Kunze in [Kun08, S. 164-166] über kontextbasierte Kooperation und mobile Prozesse einer sogenannten abstrakten Dienstklasse. Im Folgenden wird in dieser Arbeit statt von einer Dienstinstanz auch einfach die Rede von einem Dienst sein.

Eine Dienstinstanz entspricht nach Abschnitt 4.4.2 über Netzfragmentinstanzen einer Instanz des rechten Netzfragmentes des Prognosenetzes in Abbildung 5.3 ($V_{11} - V_{16}$). Entsprechend der Unterteilung von Abbildung 5.3 in eine linke und eine rechte Hälfte kann die gesamte Aufgabe der Prognose der Dienstverfügbarkeit in einen dienstspezifischen und einen dienstunabhängigen Teil unterteilt werden.

Zum Umgang mit Dienstinstanzen wird die Lösung gewählt, das Vorwissen auf der Instanzebene für alle Variablen in der rechten Hälfte des Prognosenetzes in Abbildung 5.3 vor einer Prognose oder dem Lernen von Trainingsdaten in der Vorwissenschicht auszutauschen. Dies erfolgt auf die gleiche Weise wie das Speichern und Laden von Vorwissen auf der Instanzebene (Instanzdaten) auf einen persistenten Speicher, so dass das Prognosesystem nur in vorgesehener Weise benutzt wird und keine Architekturverletzungen oder Ähnliches nötig sind. Diese Lösung nutzt aus, dass im Gegensatz zum allgemeinen Fall von Netzfragmentinstanzen jede Prognose nur eine einzelne Dienstinstanz und nie mehrere betrifft. Die Prognosestruktur bleibt daher unberührt. Wenn man z.B. mit dem Prognosesystem die Anzahl verfügbarer Dienstinstanzen prognostizieren wollen würde, wäre das nicht der Fall.

Eine wichtige Aufgabe ist die Auswahl der Dienstinstanzen, für die Prognose ange-

boten werden soll. Da erlerntes Vorwissen Speicherplatz benötigt, können im Allgemeinen nicht für jede Dienstinstanz Prognosen angeboten werden. Um eine Auswahl an Dienstinstanzen zu treffen, wird die Anzahl an Abfragen über Dienstinstanzen überwacht und gespeichert. Prognosen werden dann grundsätzlich für die am häufigsten abgefragten Dienstinstanzen angeboten. Wenn für eine Dienstinstanz schon Prognosen angeboten werden und für eine andere noch nicht, wird jedoch die erste bei der Auswahl bevorzugt. Dies hemmt den Austausch den Dienstinstanzen, für die bereits Prognose angeboten wird, durch andere Dienstinstanzen, damit erlerntes Vorwissen nicht zu leichtfertig verworfen wird. Insbesondere in dem Fall, dass Dienstinstanzen ähnlich häufig frequentiert werden, würde es sonst dazu kommen, dass diese Dienstinstanzen häufig gegeneinander ausgetauscht würden, so dass im Endeffekt für keine dieser Dienstinstanzen gutes Vorwissen verfügbar wäre und gute Prognosen möglich wären.

Akquisition von Daten über Dienstverfügbarkeit

Der dritte Anforderungsbereich liegt grundsätzlich im Rahmen der Möglichkeiten des Basissystems und ist in diesem explizit berücksichtigt, erfordert aber trotzdem in gewissem Maße die Implementierung zusätzlicher Funktionalität oberhalb der Prognosesystems. Die Rede ist von der Akquisition von Daten über die Verfügbarkeit der Diensten zum Lernen. Für die Prognose werden keine Angaben über die Verfügbarkeit eines Dienstes als gegebene Informationen benötigt, da diese die gesuchten Informationen bei der Prognose darstellt und keine anderen Variablen von ihr abhängen. Im Gegensatz zur Verfügbarkeit von Diensten erfolgen die Akquisition der Zeit und der Position vollautomatisch über Messadapter. Für die Verfügbarkeit von Diensten würde eine automatische, regelmäßige Messung durch Messadapter jedoch zu einem regelmäßigen Nachrichtenaustausch zwischen Geräten führen, der für mobile Geräte nach Möglichkeit zu vermeiden ist, um deren knappe Energiereserven zu schonen und drahtlose Netze nicht unnötig zu belasten.

Daher werden keine Daten über die Dienstverfügbarkeit speziell für das Prognosesystem gemessen, sondern nur bestehende Daten verwendet. Da solche Daten nicht unbedingt regelmäßig anfallen, werden sie der Lernschicht des Prognosesystems von der Anwendung von oben zugeführt, was im Prognosesystem als Möglichkeit vorgesehen ist ↑.

Als bestehende Daten über die Verfügbarkeit von Diensten werden Anfragen über die Verfügbarkeit von Diensten an die verteilte Registratur für Dienste der DEMAC-Middleware und die Antworten darauf verwendet [Kun08, S. 175-183]. Bei diesem Vorgehen ist zwar keine große Menge an Trainingsdaten für das Lernen zu erwarten. Dem entgegen steht jedoch, dass gerade Angaben über die Verfügbarkeit solcher Dienste am häufigsten vertreten sind, deren Verfügbarkeit am häufigsten in der verteilten Registratur abgefragt wird. Es ist zu erwarten, dass auch über die Verfügbarkeit dieser Dienste am häufigsten Prognosen abgefragt werden. Somit stehen gerade für die bei Prognosen am häufigsten frequentierten Dienste am meisten Trainingsdaten zur Verfügung.

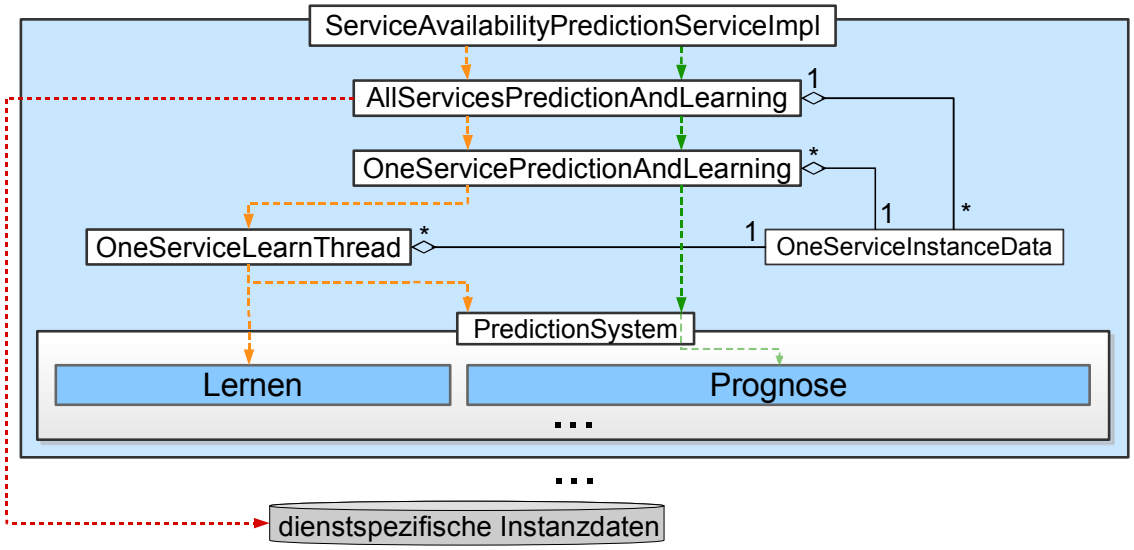

Abbildung 5.4.: Klassendiagramm zum `ServiceAvailabilityPredictionService`, grüne Pfeile: Prognose, orange Pfeile: Lernen, roter Pfeil: Speichern und Laden

Implementierung eines Dienstverfügbarkeits-Prognosedienstes

Die konkrete Realisierung der Funktionalität, die über das Prognosesystem hinaus für die Prognose der Dienstverfügbarkeit notwendig ist, wird durch das Klassendiagramm in Abbildung 5.4 dargestellt. Die Prognose der Dienstverfügbarkeit wird über den `ServiceAvailabilityPredictionService` in der DEMAC-Middleware angeboten.

Die Schnittstelle zur Benutzung des Dienstes bildet die Klasse `ServiceAvailabilityPredictionServiceImpl`. Sie führt die komplette Funktionalität zusammen und realisiert zusätzlich die verteilte Nutzung von Prognosen. Sie ist in erster Linie auf den Anwendungsfall abgestimmt, für den die Prognose momentan erwünscht ist, also die Ermittlung eines Gerätes als Migrationsziel, wenn aktuell ein bestimmter Dienst für kein potentielles Migrationsziel verfügbar ist. Daher kann über die Klasse `ServiceAvailabilityPredictionServiceImpl` prognostiziert werden, für welches Gerät die Wahrscheinlichkeit am höchsten ist, dass der Dienst in einem bestimmten Zeitraum für dieses verfügbar werden wird. Dazu wird über den Transportdienst der DEMAC-Middleware bei den Geräten jeweils die Prognose abgefragt, mit welcher Wahrscheinlichkeit der Dienst für sie in dem Zeitraum verfügbar werden wird. Dazu wird der Prognosemodus der Zeitprognose verwendet, der eine Wahrscheinlichkeitsverteilung der möglichen Zeitpunkte prognostiziert, zu denen ein Zustand eintritt. Diese Prognose über die Zeit wird auch lokal von `ServiceAvailabilityPredictionServiceImpl` angeboten, ebenso wie eine Prognose über den Zustand zu einem bestimmten Zeitpunkt (Zustandsprognose). So wird auch als möglicher zukünftiger Anwendungsfall unterstützt, dass proaktiv vor der Migration eines Prozesses geplant wird. Es kann dann nämlich bei der Planung die Prognose berücksichtigt werden, ob ein Dienst zu seinem voraussichtlichen Aufrufzeitpunkt im

Prozess für ein mögliches Migrationsziel verfügbar sein wird. Von Nutzen wäre dafür allerdings auch eine Prognose, die den voraussichtlichen Aufrufzeitpunkt eines Dienstes oder zumindest die Ausführungsdauer vorhergehender Dienste prognostiziert.

Von der Klasse `AllServicesPredictionAndLearning` existiert nur ein Exemplar und sie wird von `ServiceAvailabilityPredictionServiceImpl` benutzt. Sie realisiert die Verwaltung der Dienstinstanzen und hält das Methodenvorwissen auf der Instanzebene für die einzelnen Dienstinstanzen bereit, speichert es auf Wunsch in einer Datei und kann es auch wieder aus der Datei laden. Die Klasse `OneServiceInstanceData` kapselt das Methodenvorwissen auf der Instanzebene von einer Dienstinstanz. Wenn für eine Dienstinstanz eine Prognose durchgeführt oder die gemeldete Verfügbarkeit der Dienstinstanz gelernt werden soll, erzeugt `AllServicesPredictionAndLearning` ein Objekt des Typs `OneServicePredictionAndLearning` und übergibt diesem ein Objekt vom Typ `OneServiceInstanceData`. Vor Prognose oder Lernen wird das Vorwissen auf der Instanzebene im Prognosesystem ausgetauscht. Für Prognosen greift die Klasse `OneServicePredictionAndLearning` direkt auf die Klasse `PredictionSystem` als Teil des Prognosesystems zu. Zum Lernen startet sie einen `OneServiceLearnThread`, der im Hintergrund aus der übergebenen Information über die Verfügbarkeit eines Dienstes und den über Messadapter ermittelten Werten für Tageszeit, Ort und Netzgröße Datensätze zum Lernen für die Methoden bildet. Nicht betroffen davon ist das Lernen für den linken, dienstunabhängigen Teil des Prognosenetzes in Abbildung 5.3, das vollautomatisch im Prognosesystem erfolgt.

6 Diskussion und Evaluation

Dieses Kapitel evaluiert und reflektiert das entwickelte Verfahren der strukturierten Kontextdatenprognose und dessen Implementierung als zentrale Resultate dieser Arbeit. Dazu werden nicht-funktionale Eigenschaften wie Genauigkeit und Effizienz zunächst quantitativ untersucht. Im zweiten Teil des Kapitels erfolgt schließlich eine qualitative Überprüfung auf Erfüllung der Anforderungen des Anforderungskatalogs in Abschnitt 2.6 und eine kritische Bewertung im Hinblick auf die Ziele der Arbeit und im Vergleich zu bestehenden Verfahren.

6.1. Evaluation

Die Evaluation wird anhand der Prognose der Dienstverfügbarkeit für das DEMAC-Projekt durchgeführt, da für diese bereits ein Prognosemodell existiert (vgl. Abschnitt 5.3). Alle Messungen erfolgen auf einem ca. fünf Jahre alten Notebook, das über einen Pentium M - Prozessor mit 1,5 GHz und 1 GB Arbeitsspeicher verfügt. Es wird die Java Virtual Machine *J9* für die Java Micro Edition™ zur Ausführung verwendet.

Dabei wird entsprechend Abschnitt 5.2.2 vorgegangen, der sich mit Vorgehensweisen zum Testen des implementierten Prognosesystems befasst. Es wird eine Historie fiktiver, jedoch realitätsnaher Daten zugrunde gelegt. Die Daten beschreiben über fünf Tage hinweg den Kontext eines Nutzers mit einem mobilen Gerät wie z.B. einem Smartphone. Sie gibt für Intervalle von 15 Minuten jeweils die Position, die Netzgröße und die Verfügbarkeit diverser Dienste an. Abgesehen von der Beschleunigung der Zeit zur Evaluation in einem akzeptablen Zeitrahmen beträgt die Zeitschrittlänge im Prognosemodell ebenfalls 15 Minuten.

6.1.1. Effizienz

Effizienz bedeutet, dass gute Ergebnisse erzielt werden bei geringem Speicherbedarf und geringer Prozessorlast ↑. Zumindest zu Zeitpunkten, an denen keine Prognosen stattfinden, ergibt sich der Speicherbedarf im Wesentlichen aus dem Prognosemodell, das festlegt, wie umfangreiches Methodenvorwissen auf der Instanzebene zu speichern ist. Bei Verwendung von Wahrscheinlichkeitstabellen, die leicht den Speicherbedarf dominieren können, kann dieser aus der Größe der gespeicherten Instanzdaten abgelesen werden. Das ist möglich, weil die im Rahmen dieser Arbeit entwickelte Implementierung von Wahrscheinlichkeitstabellen von Anfang an den Speicherplatz für die komplette Tabelle inklusive ungenutzter Bereiche reserviert. Dies ist recht nützlich bei der Evaluation und der Erstellung von Prognosemodellen. Der dienstunabhängige Teil des Methodenvorwissens auf der Instanzebene, der unter Anderem der Prognose des Ortes dient, fordert in etwa 5-10 KB. Für das Methodenvorwissen auf der Instanzebene für einen Dienst werden ca. 12 KB benötigt.

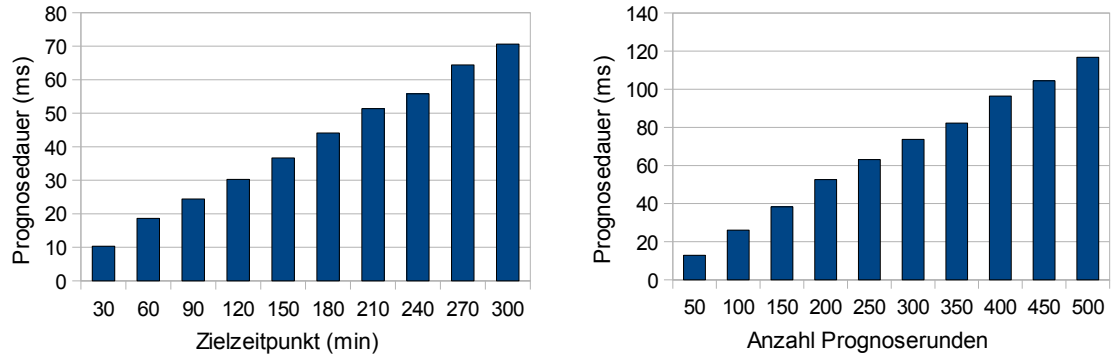

Abbildung 6.1.: links: Dauer einer Zustandsprognose bei 70 Prognoserunden in Abhängigkeit von der Entfernung des Zeitpunktes, über den die Prognose durchgeführt wird, zur Gegenwart; rechts: Dauer einer Zustandsprognose über den Zeitpunkt in 60 Minuten in Abhängigkeit von der Anzahl an Prognoserunden; jede Prognosedauer gemittelt über 176 Prognosen

Rechenaufwand entsteht sowohl bei der Prognose, als auch beim Lernen. Der Rechenaufwand beim Lernen ist jedoch vernachlässigbar. Obwohl die Zeit bei der Evaluation im 900-fachen Tempo läuft, also 900 mal so häufig gelernt wird wie sonst, bewegt sich die Prozessorauslastung durch den Prozess der Java Virtual Machine in der Regel im Bereich von 0% bis 3%.

Abbildung 6.1 zeigt Messergebnisse über die benötigte Zeit für eine Prognose. Diese betrifft zum einen den Rechenaufwand und zum anderen auch die Reaktionszeiten. Die benötigte Zeit für eine Prognose ist auf dem Testrechner entsprechend der Abbildung so gering, dass zu erwarten ist, dass eine Prognose auf leistungsschwächeren Geräten wie PDAs weniger als eine Sekunde dauert. Die Messergebnisse untermauern die in Abschnitt 4.2.2 theoretisch hergeleitete lineare Komplexität ↑ ↑. Es wird bestätigt, dass die Dauer einer Prognose linear abhängt von der Anzahl an Prognoserunden und von der zeitlichen Ferne des Zeitpunktes, über den die Prognose durchgeführt wird.

6.1.2. Genauigkeit

Die Ermittlung der Genauigkeit von Prognosen ist nicht einfach. Da das in dieser Arbeit entwickelte Prognosesystem Wahrscheinlichkeitsverteilungen als Ausdruck prinzipieller Unsicherheit prognostiziert, müsste man die entsprechenden Verteilungen der Realität ermitteln können. Dabei stößt man jedoch auf ähnliche Probleme wie bei der Prognose selbst. Daher beschränkt sich dieser Abschnitt auf eine einfache Evaluation, die Prognosen über die Verfügbarkeit eines Dienstes im Sinne von „verfügbar“ oder „nicht verfügbar“ mit der tatsächlichen zukünftigen Verfügbarkeit in der Historie vergleicht.

Dabei werden drei Einflussfaktoren auf die Genauigkeit berücksichtigt. Abbildung 6.2 stellt Ergebnisse von Genauigkeitsmessungen dar und differenziert im Bereich dieser drei im Folgenden erklärten Faktoren. Erstens wird durch die Aufteilung in ein linkes und ein rechtes Diagramm berücksichtigt, wie weit die Prognosen in die Zukunft reichen. Zweitens werden durch die unterschiedlichen Linien in den Diagrammen verschiedene

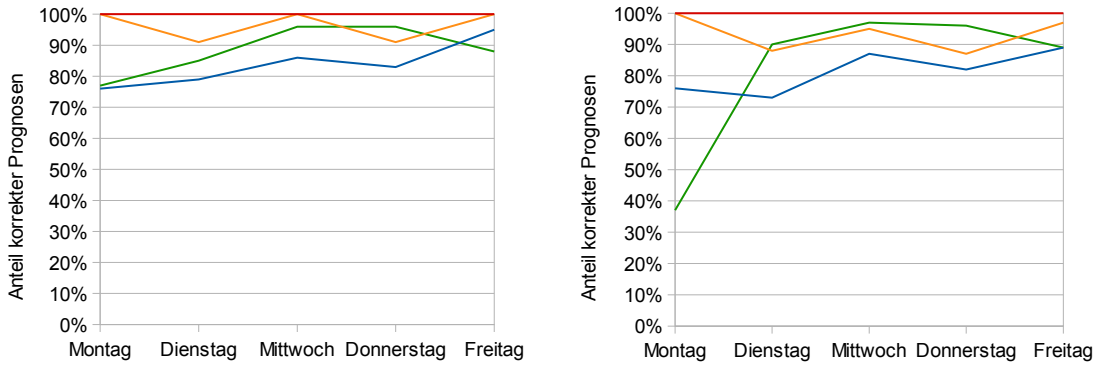

Abbildung 6.2.: Anteil korrekter Zustandsprognosen der Dienstverfügbarkeit pro Tag in der Historie für unterschiedliche Dienste; grün: Druckdienst, orange: Stadtführerdienst, rot: MMS-Nachrichtendienst, blau: Dateiaustauschdienst; Prognosen werden in kurzen Abständen hintereinander ausgeführt; links: Prognosen über Zeitpunkt in 30 Minuten, rechts: Prognosen über Zeitpunkt in 180 Minuten

Dienste unterschieden. Drittens fließt ein, an welchem Tag der Historie die Prognosen durchgeführt werden, denn das adaptive Online-Lernen beginnt am ersten Tag der Historie. Die Zusammenhänge müssen dann erst erlernt werden. Umso erstaunlicher ist, dass schon am ersten Tag eine Prognosegenauigkeit erreicht wird, die das Niveau der nachfolgenden Tage erreicht.

Dies lässt sich so erklären, dass Abhängigkeiten der Dienstverfügbarkeit von Zeit und Ort sofort gelernt werden. Wenn der Nutzer an einem Ort verbleibt, kann das entsprechende Vorwissen schon bei Prognosen genutzt werden. Für die Tageszeit gilt dies zum Teil auch, da die Tageszeit im Prognosemodell nicht genauer als nach Stunden unterschieden wird. Eine Prognose über einen Zeitpunkt in der aktuellen Stunde nutzt also schon das gewonnene Vorwissen über die Stunde. Die geringere Genauigkeit beim Ausdruckdienst am ersten Tag in der rechten Hälfte von Abbildung 6.2 kann dadurch entstehen, dass die Prognosen sich auf weiter in der Zukunft liegende Zeitpunkte beziehen.

Die insgesamt hohe Genauigkeit der Prognosen wird dadurch begünstigt, dass manche Dienste mit Abstand die meiste Zeit verfügbar und manche Dienste mit Abstand die meiste Zeit nicht verfügbar sind. Trivial verhält es sich z.B. beim MMS-Nachrichtendienst, der lokal vom Gerät selbst angeboten wird und deshalb in der Historie immer zur Verfügung steht. Daher sind die Prognosen für diesen immer korrekt. Der Stadtführerdienst ist nur an zentralen Orten verfügbar, die der Nutzer am Abend des Dienstags und des Donnerstags besucht. Daher ist die Genauigkeit für diesen Dienst am Dienstag und Donnerstag geringer. Ein möglicher Ausgangspunkt für Verbesserungen besteht in dem Ziel, die Prognosegenauigkeit für den Dienst am Donnerstag gegenüber Dienstag zu verbessern. Der Ausdruckdienst ist immer auf der Arbeit verfügbar, solange der Nutzer sich nicht zu weit vom Gerät entfernt, das ihn anbietet. Der Dateiaustauschdienst wird von mobilen Geräten angeboten, die andere Menschen bei sich tragen und denen der Nutzer begegnen kann. Für ihn ist eine leichte Steigerung der Genauigkeit über die Woche

hinweg zu erkennen.

Für das DEMAC-Projekt ist die erzielte Genauigkeit recht zufriedenstellend, da die Anteile der Prognosen, die zu richtigen Migrationsentscheidungen führen, bei über 80% liegen. Positiv hervorzuheben ist auch das schnelle Lernen. Allerdings erscheinen die verwendeten Daten und die Auswertung eher noch zu simpel, um sichere Aussagen abzuleiten. Die erzielten Ergebnisse sind jedoch vielversprechend.

Für eine genauere Evaluation kommt eine Fokussierung auf Situationen in Frage, in denen Prognose nicht zu einfach ist. Auch ein Vergleich mit trivialen Prognosen, die für einen bestimmten Dienst immer vorhersagen, dass er verfügbar ist, oder immer vorhersagen, dass er nicht verfügbar ist, macht Sinn. Interessant wäre ebenso eine Unterscheidung, wie stark die prognostizierte Wahrscheinlichkeit für die Verfügbarkeit eines Dienstes von der tatsächlichen Verfügbarkeit abweicht. Anzustreben ist außerdem eine Evaluation mit Daten aus der Context Database [May], mit denen auch andere Verfahren evaluiert wurden, so dass ein Vergleich möglich wird.

Es ist zu erwarten, dass weitere Verbesserungen der Genauigkeit durch die Implementierung der Erweiterung um verbesserten Umgang mit Unsicherheit erzielt werden können. Diese ermöglicht eine verbesserte Ergebnisintegration beim Coprocessing, so dass die Prognoseergebnisse mit der geringsten Unsicherheit bevorzugt werden.

6.2. Diskussion

In Abschnitt 2.6 wurde ein Anforderungskatalog entwickelt, der die wesentlichen Ergebnisse der kompletten Analyse des Problemfelds in Kapitel 2 zusammenfasst. Daher wurden in Abschnitt 3.2.1 und 3.2.2 bereits die Verfahren von Mayrhofer und Sigg mit Hilfe dieses Anforderungskatalogs bewertet. Auch in diesem Abschnitt wird er zugrunde gelegt.

6.2.1. Erfüllung von Anforderungen

Die folgende Tabelle zeigt, welche Anforderungen durch das Verfahren der strukturierten Kontextdatenprognose und dessen Implementierung erfüllt („✔“), mit Einschränkungen erfüllt („(✔)“) oder nicht erfüllt sind („✘“). Die Spalten „**M**“ und „**S**“ geben zum Vergleich die aus Abschnitt 3.2.1 und 3.2.2 übernommenen Ergebnisse der Verfahren an, die im Rahmen der Dissertationen von Mayrhofer und Sigg entstanden sind.

Nr.	Anforderung		Begründung / Kommentar	M	S
Kontextdaten als historische Daten					
1.1	Unterstützung von Zusammenhangsarten:				
1.1.1	Trends	(✔)	Polynomextrapolation-Methode als einfacher Ansatz	(✔)	✔
1.1.2	Sequentielle Muster	✔	durch Kanten der Form $X_{j-k} \rightarrow X_j$ im Prognosenetz	✔	✔
1.1.3	Periodische Muster	(✔)	bei fester Periode durch z.B. Tageszeit oder Wochentag als unabhängige Variable	✘	?

Nr.	Anforderung		Begründung / Kommentar	M	S
1.1.4	Lineare Zusammenhänge	✔	durch Regression	-	✔
1.1.5	Nichtlineare Zusammenhänge	✘	durch Implementierung neuer Methoden realisierbar	-	✔
1.1.6	Gemischte Zusammenhänge	✔	durch Coprocessing	✘	(✔)
1.2	Direkte Unterstützung von Skalenniveaus:				
1.2.1	Nominal	✔	vgl. Abschnitt 5.1.1	?	✔
1.2.2	Ordinal	✘	nachrüstbar	✔	?
1.2.3	Metrisch	(✔)	Ratio-Skalenniveau unterstützt (vgl. Abschnitt 5.1.1)	✔	✔
1.3	Unterstützung von Entitätsarten:				
1.3.1	Menschen	(✔)	keine besonderen Optimierungen	✔	(✔)
1.3.2	Technik	(✔)	keine besonderen Optimierungen	✔	(✔)
1.4	Optimierungen primärer Kontext	(✔)	Messadapter für Position und Zeit sowie Methoden für Zeit implementiert, für DEMAC Ansätze zur Ortsprognose erarbeitet (vgl. Abschnitt 5.1.2, 4.2.5, 5.3.1)	✘	✘
1.5	Nutzung des Zusammenhangs prim. Kont. → sekund. Kont.	(✔)	möglich durch entsprechendes Prognosenetz wie z.B. bei dem für DEMAC (vgl. Abbildung 5.3)	✘	✘
1.6	Horizontale Verarbeitung auf unterschiedlichen Abstraktionsebenen:				
1.6.1	Low-Level-Kontext	✔	keine Einschränkungen bei Abstraktionsebenen	✘	✔
1.6.2	High-Level-Kontext	✔	keine Einschränkungen bei Abstraktionsebenen	✔	✔
1.7	Vertikale Verarbeitung	✘	nachrüstbar als Methode	✔	✘
Umgang mit Dynamik von Kontext					
2.1	Instanzen	(✔)	entsprechende Erweiterung vorgesehen, jedoch nicht implementiert (vgl. Abschnitt 4.4.2)	(✔)	(✔)
2.2	Aktive Elemente	✔	möglich durch entsprechendes Prognosenetz (vgl. Abschnitt 4.3.2)	✘	?
Anpassung an Nutzer					
3.1	Adaptives Online-Lernen	✔	als Hauptlernart fokussiert (vgl. Abschnitt 4.2.3)	✔	(✔)
3.2	Unterstützung bei Erstellung von Prognosemodell	(✔)	systematisches Vorgehen entwickelt, Werkzeugunterstützung angedacht (vgl. Abschnitt 4.3.1)	-	-
3.3	Einbringen von A Priori - Wissen:				
3.3.1	Durch Offline-Batch-Lernen	(✔)	möglich, jedoch nicht speziell dafür optimale Methoden betrachtet (vgl. Abschnitt 4.2.3)	✘	?
3.3.2	Durch deduktives Lernen	(✔)	möglich, keine besonderen Optimierungen, kein Editieren von Instanzdaten von Hand möglich (vgl. Abschnitt 5.1.1)	✘	?
Integration in Anwendungsszenarien					

Nr.	Anforderung		Begründung / Kommentar	M	S
4.1	Plattformunabhängigkeit	✔	durch Implementierung in Java, mögliche zusätzliche Messadapter evtl. nicht	(✔)	?
4.2	Universelle Schnittstellen	(✔)	kein Publish-Subscribe implementiert, mehrere Möglichkeiten der Datenakquisition zum Lernen (vgl. Abschnitt 4.1.2)	(✔)	?
4.3	Universelles Kontext-Metamodell	✔	Variablen als allgemeines Konzept	✘	✔
4.4	Unterstützung von Rollen der Zeit in gesuchten Informationen:				
4.4.1	Informationen über Variablen/Ereignisse gesucht	✔	Modus der Zustandsprognose	✔	✔
4.4.2	Zeitpunkt gesucht	✔	Modus der Zeitprognose	(✔)	✔
4.5	Unterstützung von Informationen über Variablen:				
4.5.1	Wahrscheinlichkeitsverteilung	✔	vgl. Abschnitt 4.2.2	(✔)	✘
4.5.2	Modalwert	✔	von Implementierung aus Verteilung berechnet	✔	✔
4.5.3	Wahrscheinlichkeit zu logischer Formel	(✔)	durch Variable in Prognosenetz, die den Wahrheitswert der Formel besitzt, und entsprechende zu implementierende Methode	✘	✘
4.5.4	Erwartungswert und Varianz	✘	Berechnung aus Verteilung leicht nachrüstbar	✘	?
Genauigkeit					
5.1	Fundierte Angabe prinzipieller Unsicherheit	✔	als Wahrscheinlichkeitsverteilung, die jedoch im Basissystem auch andere Unsicherheitsarten beinhaltet (vgl. Abschnitt 4.4.3)	(✔)	✘
5.2	Hohe Generalisierungsfähigkeit	(✔)	abhängig von Methoden, gegeben z.B. bei Regression	(✔)	(✔)
5.3	Genug Trainingsdaten lernen	(✔)	bei Wahrscheinlichkeitstabellen Bedarf hoch, Lernfrequenz im Prognosemodell einstellbar	✔	(✔)
5.4	Fundierte Angabe Schätzunsicherheit	(✔)	als Erweiterung bedacht, nicht implementiert, Angabe nicht fundiert wegen Heterogenität von Methoden (vgl. Abschnitt 4.4.3)	✘	?
5.5	Lernen stabiler Inseln von Zusammenhängen	✔	durch geeignetes Prognosenetz in verschiedensten Anwendungsdomänen	(✔)	(✔)
5.6	Berücksichtigung räumlich und zeitlich nahen Kontextes	(✔)	bei geeignetem Prognosenetz	(✔)	(✔)
5.7	Einbeziehen zusätzl. Inform.	(✔)	manuell möglich durch Variablenwahl	(✔)	(✔)
5.8	Angabe Variationsunsicherheit	(✔)	als Erweiterung bedacht, nicht implementiert (vgl. Abschnitt 4.4.3)	✘	?
5.9	Anpassung an Änderungen von Zusammenhängen	(✔)	als Erweiterung bedacht, nicht implementiert (vgl. Abschnitt 4.4.3)	(✔)	(✔)

Nr.	Anforderung		Begründung / Kommentar	M	S
5.10	Geringe Methodenunsicherheit	(✔)	abhängig von Methoden, sehr gut z.B. bei Wahrscheinlichkeitstabellen	(✔)	(✔)
5.11	Angabe Methodenunsicherheit	(✔)	als Erweiterung bedacht, nicht implementiert (vgl. Abschnitt 4.4.3)	✘	?
5.12	Umgang mit Mängeln in Trainingsdaten:				
5.12.1	Fehlerhafte Werte	✘	keine besondere Behandlung	✘	?
5.12.2	Fehlende Werte	(✔)	Prognosen trotz fehlender Werte möglich, Lernen nur bei vollständigen Datensätzen (vgl. Abschnitt 3.4.2, 4.2.2)	✔	✔
5.12.3	Subjektive Einschätzungen	✘	keine spezielle Berücksichtigung im Verfahren (vgl. Abschnitt 3.1.4, 3.1.5)	?	?
5.13	Konsistenz mehrerer Prognosen	✘	eher gering durch Zufallszahlen bei Prognosen (vgl. Abschnitt 4.2.2)	✔	?
Effizienz					
6.1	Abstraktion beim Lernen	(✔)	abhängig von Methoden, sehr gut z.B. bei Regression	✔	(✔)
6.2	Prognosebedarf berücksichtigen	✔	Abstimmung auf benötigte Prognosen durch Prognosemodell	?	?
6.3	Angepasste Aufwandsverteilung auf Lernen und Prognose	✔	durch Wahl von Methoden	✔	✔
6.4	Begrenzbarkeit Ressourcennutzung	(✔)	abhängig von Methoden, bei Methoden der Referenzkonfiguration möglich, Zeitaufwand des koordinierenden Prognosealgorithmus begrenzbar (vgl. Abschnitt 4.2.2)	✘	?
6.5	Kein unnötiges Lernen	✘	keine automatische Steuerung der Lernfrequenz, nachrüstbar	?	?
6.6	Stabil geringe Reaktionszeiten	(✔)	in der Regel eher gering, Variation möglich durch Sperren (vgl. Abschnitt 4.2.2, 5.2.1)	✔	(✔)
6.7	Effizienz bei vielen Variablen	(✔)	linear steigender Aufwand bei gleichartigen Variablen (vgl. Abschnitt 4.2.2, 4.2.3)	✔	(✔)
6.8	Skalierbarkeit	✔	zur Laufzeit durch Anzahl Prognoserunden, zur Entwicklungszeit durch Prognosemodell	(✔)	(✔)
Verteilte Prognose					
7.1	Umgang mit heterogenen Trainingsdaten	(✔)	durch Implementierung entsprechender Messadapter, zeitliche Abstimmung vorgesehen, jedoch nicht implementiert (vgl. Abschnitt 4.4.1)	(✔)	?
7.2	Angabe von Fähigkeiten	✘	nachrüstbar	?	?
7.3	Wahlmöglichkeiten Verteilung	(✔)	Verteilung liegt zu großen Teilen in Händen der Anwendung (vgl. Abschnitt 2.5.8, 4.4.1)	(✔)	✔

Nr.	Anforderung		Begründung / Kommentar	M	S
7.4	Ansätze gegen instabile Vernetzung	✘	nachrüstbar	?	?
7.5	Effizienzoptimierungen abonnierte Prognosen	✘	nachrüstbar	?	?
Benutzbarkeit					
8.1	Geringer Vorbereitungsaufwand für Anwendungsentwickler	✘	Erstellung von Prognosemodell und evtl. Implementierung von Messadaptern und Methoden nötig	✔	✔
8.2	Geringer Aufwand beim Einsatz für Anwendungsentwickler	✔	vgl. Abschnitt 5.2.3	?	?
8.3	Unaufdringlichkeit für Nutzer	✔	keine Einbeziehung des Nutzers nötig, solange Anwendung es nicht erfordert	(✔)	✔
8.4	Nachvollziehbarkeit für Nutzer	(✔)	graphischer Ansatz mit Netzen eröffnet Möglichkeit der Erklärung von Prognosen (vgl. z.B. [Wit02, S. 66-67])	?	?

Ziel dieser Arbeit ist die Entwicklung eines generischen Verfahrens zur Kontextdatenprognose auf mobilen Geräten. Der Anforderungskatalog gibt Kriterien zur Prüfung an, inwieweit das Ziel erreicht ist. Dabei kann zwischen funktionalen und nicht-funktionalen Anforderungen unterschieden werden. Die funktionalen Anforderungen sind durch die Forderung nach Generik bestimmt. Das Verfahren soll entsprechend den Anforderungsgruppen des Anforderungskatalog durch seine Funktionalität möglichst viele Arten von Kontextdaten unterstützen, mit Dynamik im Kontext umgehen können, Verteilung unterstützen und sich in verschiedenste Anwendungsszenarien fügen sowie an Nutzer individuell anpassen. Bei all diesen Funktionalitäten wird eine möglichst hohe Güte angestrebt, wie die nicht-funktionalen Anforderungen in den Bereichen Genauigkeit, Effizienz und Benutzbarkeit fordern. Die Effizienz spielt für den Einsatz auf mobilen Geräten eine wichtige Rolle.

Die funktionalen und nicht-funktionalen Anforderungen werden zum Teil durch grundlegende Konzepte und zum Teil durch die Implementierung einzelner Funktionalitäten erfüllt. Das Verfahren und die Implementierung bieten gute Möglichkeiten zur Erweiterung um einzelne Funktionalitäten. Eine gute Erweiterbarkeit ist konkret insbesondere für die Anwendungsentwickler realisiert. Das gelingt durch das Prinzip des Einbringens von Wissen über die Anwendungsdomäne zur Entwicklungszeit durch die Anwendungsentwickler (vgl. Abschnitt 4.1.1). Die Anwendungsentwickler erstellen ein Prognosemodell und können im Rahmen der Implementierung des Verfahrens als Framework Messadapter und Methoden implementieren (vgl. Abschnitt 5). Durch eine Erweiterung der Referenzkonfiguration verwendeter Methoden können Erweiterungen für alle Anwendungsentwickler erfolgen, die das Prognosesystem nutzen. Für Erweiterungen in Form zusätzlicher Funktionalität oberhalb des Prognosesystems auf der Anwendungsebene einer Anwendung besteht genug Flexibilität, wie im Zusammenhang mit DEMAC gezeigt wurde ↑. Im Bereich der Verteilung ist regulär zu großem Teil eine Umsetzung durch die Anwendungsentwickler vorgesehen, so dass es zu keinem Bruch mit den Strategien und Techniken der Anwendung für Verteilung kommt ↑↑.

6.2.2. Bewertung der Funktionalität

Auch jenseits der Erweiterbarkeit wird das Ziel der Generik erfüllt, das die funktionalen Anforderungen bestimmt. Das gelingt zu einem großen Teil durch das Prinzip der Einbringens von Wissen über die Anwendungsdomäne zur Entwicklungszeit durch die Anwendungsentwickler in Kombination mit dem Prinzip des hybriden Einsatzes von austauschbaren Methoden (vgl. Abschnitt 4.1.1). So wird z.B. Anforderung 1.1.6 nach der Unterstützung gemischter Zusammenhänge durch Coprocessing erfüllt. Durch ein geeignetes Prognosenetz kann entsprechend Anforderung 1.5 die Abhängigkeit sekundären Kontextes von primärem ausgenutzt werden oder auf Wunsch entsprechend Anforderung 2.2 eine ereignisorientierte Sichtweise eingenommen werden.

Durch die Implementierung neuer Messadapter und Methoden können von Anwendungsentwicklern auch zusätzliche Funktionalitäten realisiert werden, die in obiger Tabelle mit „✘“ gekennzeichnet sind. Das gilt unter Anderem für die Unterstützung von Zusammenhangsarten (Anforderung 1.1), die Unterstützung von Entitätsarten (Anforderung 1.3), Optimierungen für primären Kontext (Anforderung 1.4), vertikale Verarbeitung (Abbildung 1.7) und eine gute Unterstützung für das Einbringen von a priori vorhandenem Wissen (Anforderung 3.3). Solche Erweiterungen sind grundsätzlich, z.B. bei der Unterstützung von Zusammenhangsarten, auch bei Mayrhofer und Sigg durch die Implementierung neuer Methoden möglich ↑ ↑. Im Gegensatz dazu bietet das Verfahren der strukturierten Kontextdatenprognose durch den Einsatz mehrerer Methoden jedoch die Möglichkeit, die Vorzüge mehrerer Methoden zu kombinieren und sich nicht auf die Eigenschaften einer Methode festlegen zu müssen ↑.

Das Ziel der Generik wird über das Einbringen von Wissen über die Anwendungsdomäne hinaus durch adaptives Online-Lernen, also kontinuierliches Lernen zur Laufzeit beim Nutzer erreicht (Anforderung 3.1). Das Prognosesystem passt sich auf diese Weise zur Laufzeit an die individuellen Zusammenhänge im Kontext des Nutzers an ↑. Durch die Fokussierung auf Methoden, die adaptives Online-Lernen unterstützen, wird eine Einschränkung auf bestimmte Methoden akzeptiert ↑, die für die Erfüllung weiterer funktionaler und nicht-funktionaler Anforderungen von Nachteil ist. Der durch die Anpassung an den Nutzer zur Laufzeit entstehende Nachteil wird jedoch dadurch in Grenzen gehalten, dass die Prognosestruktur, also die Menge der Verbindungen zwischen den Methoden, schon zur Entwicklungszeit durch das Prognosenetz festgelegt wird und nicht ebenso beim Nutzer zu erlernen ist. Dadurch wird allerdings vorausgesetzt, dass die Zusammenhangsstruktur bei allen Nutzern so beschaffen ist, dass sich eine einzige Prognosestruktur für alle Nutzer eignet, was die Generik in gewissem Maße einschränkt.

Die Funktionalität des Prognosesystems wird auch durch den koordinierenden Prognosealgorithmus beeinflusst. Er spielt seine Stärken bei der Unterstützung von Rollen der Zeit in Prognoseergebnissen (Anforderung 4.4) durch die Modi der Zustands- und Zeitprognose und die Unterstützung von Wahrscheinlichkeitsverteilungen (Anforderung 4.5.1) aus. Einfachere Informationen wie der Modalwert werden aus einer prognostizierten Verteilung berechnet. Die Erfüllung funktionaler Anforderungen geschieht auch durch die Erweiterung um Verteilung (Anforderung 7) und die Erweiterung um Netzfragmentinstanzen (Anforderung 2.1).

6.2.3. Bewertung nicht-funktionaler Eigenschaften

Im Bereich der nicht-funktionalen Anforderungen geht es um die Prognosegenauigkeit, die Effizienz und die Benutzbarkeit. Grundsätzlich besteht hier die Gefahr, dass ein recht generisches System zwar funktional verschiedensten Anwendungsbereichen gerecht wird, jedoch nur mit mäßiger Qualität. Mayrhofer umgeht diese Gefahr, indem er sich auf High-Level-Kontext, also Kontext auf hoher Abstraktionsebene beschränkt ↑. So beschränkt es jedoch die Generik erheblich. Das Verfahren der strukturierten Kontextdatenprognose unterstützt dagegen auch Low-Level-Kontext (Anforderung 1.6.1) ↑ und ist insgesamt recht generisch (vgl. oben), vermeidet aber trotzdem stärkere Einschränkungen im nicht-funktionalen Bereich. Das gelingt durch das Prinzip des Einbringens von Wissen über die Anwendungsdomäne zur Entwicklungszeit durch die Anwendungsentwickler (vgl. Abschnitt 3.2.4). Durch ein Prognosemodell und eigene Messadapter und Methoden können Anwendungsentwickler das Prognosesystem so genau auf den Anwendungsbereich abstimmen, dass es nicht unbedingt erheblich schlechter angepasst sein muss als eine Eigenentwicklung. Das Verfahren von Sigg bietet durch die Austauschbarkeit von Methoden in gewissem Maße auch das Einbringen von Wissen über die Anwendungsdomäne, setzt jedoch auf den Einsatz einer einzelnen Methode ↑. Da es keine universelle Methode ohne Schwächen bezüglich bestimmter Anforderungen gibt, ist fraglich, ob mit einer einzelnen Methode Genauigkeit und Effizienz in für mobile Geräte zufriedenstellendem Maße erreicht werden können (vgl. Abschnitt 3.2.2, 3.2.4). Das Verfahren der strukturierten Kontextdatenprognose bietet dagegen die Möglichkeit des hybriden Einsatzes mehrerer Methoden, um der Vielfalt von Kontext für die Erfüllung nicht-funktionaler Anforderungen ebenso wie für die Erfüllung funktionaler Anforderungen gerecht zu werden. Insgesamt werden die Vorteile für Genauigkeit und Effizienz durch einen im Vergleich zum Verfahren von Mayrhofer hohen Vorbereitungsaufwand für die Anwendungsentwickler und damit eine erschwerte Benutzbarkeit erkauft (Anforderung 8.1).

Durch hybriden Einsatz mehrerer, austauschbarer Methoden kann für einen Zusammenhang zwischen einzelnen Variablen eine möglichst optimale Methode gewählt werden, so dass sie z.B. Zusammenhangsart und Skalenniveaus gerecht wird, die auch für die Prognosegenauigkeit eine Rolle spielen. Darüber hinaus können viele weitere Faktoren in die Wahl von Methoden einfließen. Von Bedeutung sind unter Anderem die Anzahl der auftretenden Wertekombinationen der unabhängigen Variablen, der vorhandene Speicherplatz und die von den Anwendungsentwicklern akzeptierte Zeit für das Lernen bis zu den ersten guten Ergebnissen. Es kann dann z.B. die Wahl einer Methode wie Regression in Frage kommen, die gekennzeichnet ist durch eine hohe Generalisierungsfähigkeit (Anforderung 5.2), eine hohe Methodenunsicherheit (Anforderung 5.10) und eine starke Abstraktion von den Trainingsdaten (Anforderung 6.1). Statt Regression könnte auch eher eine weniger stark generalisierende Methode mit geringer Methodenunsicherheit, jedoch hoher Schätzunsicherheit wie Wahrscheinlichkeitstabellen Sinn machen. Eine Wahl von Regression beeinflusst auch den durch fehlerhafte gelernte Werte entstehenden Schaden (Anforderung 5.12.1) ↑. Wenn besonders kurze Reaktionszeiten bei der Anforderung von bestimmten Prognosen erwünscht sind (Anforderung 6.6), bietet sich für die

beteiligten Variablen die Wahl von Methoden an, die die wesentliche Arbeit beim Lernen statt bei der Prognose durchführen (Anforderung 6.3). Durch Wahl entsprechender Methoden und eines geeigneten Prognosenetzes kann auch der Ressourcenverbrauch insgesamt reguliert werden. Durch diese Skalierbarkeit (Anforderung 6.8) ist der Einsatz auf unterschiedlich leistungsstarken Geräten möglich.

Es wird deutlich, wie unterschiedlich die Prioritäten für einzelne Anforderungen in verschiedenen Anwendungsbereichen sein können und wie unterschiedlich die Vorzüge und Defizite von Methoden ausfallen. Das betrifft zum Teil die ganze Prognose und zum Teil punktuell einzelne Variablen. Auch Abschnitt 3.3.4 zeigt auf, wie unterschiedlich die Charakteristika von zwei konkreten Methoden sein können. Deshalb sind die Wählbarkeit und der hybride Einsatz mehrerer Methoden von großem Nutzen.

Unabhängig von der Wahl konkreter Methoden bietet die Verknüpfung der Methoden durch ein Prognosenetz den wesentlichen Vorteil mit sich, dass bei einem guten Prognosenetz stabile Inseln von Zusammenhängen erlernt werden (Anforderung 5.5). Variablen, zwischen denen kein Zusammenhang besteht, brauchen im Prognosenetz nicht verbunden zu werden. Durch geschickte Wahl der Variablen und der Prognosestruktur, also der Verbindungen zwischen den entsprechenden Methoden kann die Variationsunsicherheit, also die Variation der zu lernenden Zusammenhänge durch äußere Umstände minimiert werden. Neben den Zusammenhängen im Kontext wird durch die Erstellung eines Prognosenetzes auch berücksichtigt, was überhaupt prognostizierbar sein soll (Anforderung 6.2). Das erleichtert den effizienten Umgang mit Ressourcen, ist aber nicht selbstverständlich. Beim Verfahren von Sigg, das die Werte von Variablen zu einem Zeitpunkt als mehrdimensionalen Wert behandelt ↑, ist unklar, ob der Prognosebedarf überhaupt in irgendeiner Weise Berücksichtigung findet.

Der die Methoden koordinierende Prognosealgorithmus weist durch die Wahlmöglichkeit der Anzahl an Prognoserunden eine sehr gute Skalierbarkeit bezüglich Rechenzeit auf. Er zeichnet sich außerdem durch seine durchgängig lineare Komplexität aus (vgl. Abschnitt 4.2.2 ↑ ↑). Hervorzuheben ist auch seine Unterstützung von Wahrscheinlichkeitsverteilungen, durch die prinzipielle Unsicherheit ausgedrückt wird (Anforderung 5.1), im Basissystem jedoch vermischt mit anderen Unsicherheitsarten. Abhilfe soll die Erweiterung um verbesserte Behandlung von Unsicherheit schaffen, die einen großen Beitrag zur Prognosegenauigkeit leisten kann. Dies betrifft eine ganze Reihe von Anforderungen des Anforderungskatalogs, die mit Einschränkungen erfüllt sind („(✔)“), da die Erweiterung im Rahmen dieser Arbeit nicht implementiert ist.

Wenige Anforderungen wurden bei der Entwicklung des Verfahrens der strukturierten Kontextdatenprognose gar nicht direkt berücksichtigt. Dazu zählen der Umgang mit fehlerhaften Werten in akquirierten Daten (Anforderung 5.12.1) und die Unterstützung subjektiver Einschätzungen (Anforderung 5.12.3). Zum Umgang mit subjektiven Einschätzungen wurden in Kapitel 3 Ansätze genannt (vgl. Abschnitt 3.1.4, 3.1.5). Inwieweit diese im Prognosesystem nachrüstbar sind, ist unklar. Prinzipiell mit dem entwickelten Verfahren kaum erfüllbar ist wegen der Nutzung von Zufallszahlen bei der Prognose die Konsistenz mehrerer aufeinander folgender Prognosen (Anforderung 5.13).

6.2.4. Fazit

Das Verfahren der strukturierten Kontextdatenprognose erfüllt entsprechend obigen Ausführungen das Ziel der Generik aus dem im Wesentlichen die funktionalen Anforderungen abgeleitet sind. Dabei werden gleichzeitig auch Anforderungen im Bereich der Genauigkeit und Effizienz in zufriedenstellendem Maße erfüllt. Dafür werden Einbußen bei der Benutzbarkeit für die Anwendungsentwickler durch einen hohen Vorbereitungsaufwand in Kauf genommen.

Das Verfahren der strukturierten Kontextdatenprognose und dessen Implementierung eignen sich aufgrund ihrer Eigenschaften besonders für bestimmte Anwendungen, in denen Kontextdatenprognose eingesetzt werden soll. Die Generik kommt Anwendungen entgegen, die eine hohe Flexibilität erfordern, z.B. durch mehrere Prognosen unterschiedlicher Art. Falls dagegen ausschließlich Prognosen mit High-Level-Kontext nötig sind, bietet sich das Verfahren von Mayrhofer an. Insbesondere in dem Fall, dass sich der Kontext der Nutzer in den zu prognostizierenden Bereichen unterscheidet und auch dort Generik gefordert ist, kann das Verfahren der strukturierten Kontextdatenprognose seine Stärke des adaptiven Online-Lernens ausspielen. Wenn außerdem eine möglichst hohe Genauigkeit, eine hohe Effizienz und evtl. auch der Einsatz auf mobilen Geräten erwünscht sind, spricht Vieles für den Einsatz des Verfahrens. Wenn die Anwendungsentwickler bereit sind, einen gewissen Aufwand zu investieren, stellen das Verfahren der strukturierten Kontextdatenprognose und dessen Implementierung eine vielversprechende Lösung dar, um die Ziele der Anwendungsentwickler für ihre Anwendung zu erreichen.

7 Zusammenfassung und Ausblick

Dieses Kapitel fasst zunächst die Inhalte der Arbeit zusammen. Dann gibt es einen Überblick über die erbrachten Beiträge der Arbeit zur Forschung und zeigt auf, welche Aufgaben und Probleme noch offen sind.

7.1. Zusammenfassung

Diese Arbeit ist motiviert durch die Vision des Ubiquitous Computing und den Forschungsbereich der Context-Awareness. Eine allgegenwärtige, möglichst unsichtbare Unterstützung des Nutzers durch Informationstechnik wie durch einen Assistenten wird durch die Berücksichtigung des Kontextes, also der kompletten Nutzungssituation in verstärktem Maße ermöglicht ↑. Einen weiteren Schritt stellen Prognosen über Kontext dar, durch die auch proaktives, vorausschauendes Handeln von Informationstechnik möglich wird ↑. Diese Arbeit stellt sich der Aufgabe, eine generische Systemunterstützung für Kontextdatenprognose bereitzustellen, um Anwendungsentwickler von der Entwicklung kompletter Eigenlösungen zu entlasten ↑. Dabei wird auch die besondere Bedeutung mobiler Geräte im Ubiquitous Computing berücksichtigt, auf denen die Kontextdatenprognose ausgeführt werden soll ↑. Da das Problem der Bereitstellung von Systemunterstützung zur Kontextdatenprognose bisher in der Forschung nur zum Teil gelöst ist, liegt der Schwerpunkt dieser Arbeit mehr darin, gute Prognosen durchzuführen, als darauf aufbauend weiterführende Funktionalitäten zu bieten.

Kapitel 2 dieser Arbeit analysiert den Problemkomplex, der von Ubiquitous Computing, Context-Awareness und mobilen Geräten auf der einen Seite und Prognose auf der anderen Seite ausgeht ↑. Besonders wichtig sind die Themen der Context-Awareness und der Prognose. Die besonderen, für die Realisierung von Kontextdatenprognose relevanten Eigenschaften von Kontext werden anhand des Forschungsbereichs der Context-Awareness analysiert. Im Vordergrund stehen unter Anderem Arten von Kontext, Darstellungen von Kontext und Aktivitäten in Verbindung mit Kontext. Der Themenbereich der Prognose ist im Gegensatz zur Context-Awareness sehr zersplittert und erstreckt sich über verschiedene Wissenschaften. Damit die Betrachtungen über Prognose keine Lösungsmöglichkeiten zur Realisierung von Prognose vorweg nehmen, wird eine universelle, probabilistische Perspektive eingenommen ↑. Um einer Anwendung Angaben zur Genauigkeit einzelner Prognosen anbieten zu können und die Genauigkeit zu verbessern, werden verschiedene Arten von Unsicherheit unterschieden ↑.

Die Analysen in den Bereichen Ubiquitous Computing, Context-Awareness, mobile Geräte und Prognose münden in ein Spektrum von Anforderungen an ein Prognosesystem zur Kontextdatenprognose auf mobilen Geräten, das insbesondere aufgrund der geforderten Generik recht breit ausfällt. Wichtige Bereiche funktionaler Anforderungen bestehen in der Unterstützung der Eigenschaften von Kontextdaten bei der Prognose, der

Anpassung an die Anwendung und den Nutzer sowie der Unterstützung von verteilter Prognose. In diese Anforderungen gehen z.B. die unterschiedlichen Arten von Kontext und dessen Eigenschaft der Dynamik ein. Von besonderer Bedeutung ist die Feststellung, dass wegen inter-individueller Unterschiede zwischen Nutzern zur Laufzeit beim Nutzer adaptiv gelernt können werden sollte ↑. Die nicht-funktionalen Anforderungen betreffen die Gebiete Prognosegenauigkeit, Effizienz und Benutzbarkeit. Die Benutzbarkeit bezieht sich in erster Linie darauf, welcher Aufwand für die Anwendungsentwickler durch das Prognosesystem entsteht. Die Anforderungen bezüglich Genauigkeit und Effizienz ergeben sich überwiegend aus der Analyse des Problembereichs der Prognose. Eine hohe Effizienz ist im Grundsatz besonders durch den Einsatz auf mobilen Geräten motiviert. Die Anforderungen sind in einem Anforderungskatalog als wesentliches Ergebnis des Kapitels zusammengefasst.

In Kapitel 3 werden bestehende Lösungsmöglichkeiten zur Realisierung von Kontextdatenprognose auf mobilen Geräten untersucht. Dazu werden zunächst relevante Wissenschaftszweige sondiert, die vielen Lösungsmöglichkeiten zugrunde liegen. Dann werden komplette Verfahren zur Kontextdatenprognose betrachtet und anhand des Anforderungskatalogs bewertet. Da eine ganze Reihe von Ansätzen im Zusammenhang mit verschiedenen Forschungsaktivitäten im Bereich der Context-Awareness existieren und sich diese Arbeit eine generische Lösung zum Ziel gesetzt hat, werden nur wissenschaftliche Arbeiten herangezogen, die sich ausschließlich und in allgemeiner Weise mit Kontextdatenprognose befassen. Es stellt sich jedoch heraus, dass sich keines der untersuchten Verfahren als Basis für diese Arbeit eignet. Die Verfahren von Mayrhofer und Petzold weisen durch eine Beschränkung auf bestimmte Arten von Kontext eine zu geringe Generik auf ↑ ↑. Beim Verfahren von Sigg ist fraglich, ob eine einzelne Methode den an sie gestellten Anforderungen gerecht werden kann ↑. Das Verfahren von Petzold beinhaltet aber den viel versprechenden Ansatz des hybriden, parallelen Einsatzes mehrerer Methoden zur Prognose eines Wertes ↑. Zur Planung des weiteren Vorgehens im Kapitel fällt in einer weiterführenden Diskussion bereits die Entscheidung für insbesondere zwei zentrale Prinzipien. Erstens sollen mehrere, austauschbare Methoden hybrid eingesetzt werden. Dadurch reduzieren sich die Anforderungen an die Methoden im Vergleich zum Einsatz einer einzelnen Methode und die Vorzüge unterschiedlicher Methoden können kombiniert werden ↑. Zweitens soll Wissen über die Anwendungsdomäne von den Anwendungsentwicklern zur Entwicklungszeit eingebracht werden ↑. Diese Entscheidung wird getroffen, um trotz Generik durch Anpassung an die Anwendungsdomäne gute Leistungen im Bereich nicht-funktionaler Anforderungen wie Genauigkeit und Effizienz zu erreichen ↑. Dabei werden Einbußen bei der Benutzbarkeit durch erhöhten Aufwand für die Anwendungsentwickler in Kauf genommen ↑.

Da die betrachteten Verfahren nicht als Basis für diese Arbeit in Frage kommen, werden Bausteine für die Ausgestaltung eines eigenen Verfahrens gesammelt. Dazu werden zunächst eine Reihe von Methoden gesichtet, die zur Prognose in Frage kommen. Dabei werden Methoden ausgeschlossen, die sich nicht für adaptives Online-Lernen eigenen oder einen zu hohen Ressourcenaufwand verursachen ↑. Von den verbleibenden Methoden werden solche gewählt, die sich gut ergänzen und damit als Auswahl für einen

hybriden Einsatz anbieten ↑. Die gewählten Methoden bilden den Ausgangspunkt für eine Referenzkonfiguration von Methoden ↑.

Schließlich werden zur Umsetzung des Prinzips des hybriden Einsatzes von Methoden Ansätze zur Hybridität betrachtet. Chain- und Coprocessing stellen einen sequentiellen bzw. parallelen Einsatz von Methoden dar. Bayes'sche Netze werden als Verallgemeinerung dieser Konzepte angesehen. Es finden sich unter den Ansätzen zum Lernen in bayes'schen Netzen sowie zur Erstellung bayes'scher Netze und in stärkerem Maße bei der Prognose in bayes'schen Netzen Anknüpfungspunkte für das zu entwickelnde Verfahren ↑.

In Kapitel 4 wird das Verfahren der strukturierten Kontextdatenprognose ausgearbeitet. Kern des Verfahrens ist eine Kombination der zwei oben genannten zentralen Prinzipien ↑. Es werden mehrere, austauschbare Methoden hybrid eingesetzt. Die Methoden führen also lokal Prognosen durch und sind zur Bestimmung von Gesamtprognosen miteinander verbunden. Das Wissen über die Anwendungsdomäne wird als Wissen über eine geeignete Zusammenstellung von Methoden zur Entwicklungszeit als Teil eines Prognosemodells eingebracht ↑. Wenn die im Rahmen dieser Arbeit benutzte Referenzkonfiguration von Methoden nicht ausreicht, können Anwendungsentwickler neue Methoden implementieren ↑. Die Zusammenstellung der Methoden, also die Struktur, wird durch ein Prognosenetz dargestellt. Prognosenetze werden im Rahmen dieser Arbeit in Anlehnung an bayes'sche Netze definiert. Während die Wahl der Struktur und der Methoden als schwierige Aufgaben zur Entwicklungszeit durch Menschen geschehen ↑, erfolgt eine Anpassung an die individuellen Nutzer zur Laufzeit mittels adaptivem Online-Lernen durch die einzelnen Methoden der Variablen ↑. Für die Erstellung von Prognosenetzen wird ein systematisches Vorgehen entwickelt.

Ein wichtiger Ausgangspunkt des Verfahrens der strukturierten Kontextdatenprognose ist seine Architektur für ein Prognosesystem, das nach dem Verfahren arbeitet. Sie enthält die vier Schichten „Datenakquisition", „Vorwissen", „Prognose" und „Lernen", die die Eckpfeiler des Verfahrens bilden und separat ausgearbeitet werden. Innerhalb der Schichten werden die Methoden wegen ihrer Austauschbarkeit als separate Bestandteile behandelt ↑. Die Vorwissenschicht enthält das Vorwissen, das die Zusammenhänge im Kontext beschreibt und damit als Grundlage zur Prognose dient. Beim Lernen werden den Methoden akquirierte Daten zugeführt, die von ihnen als Trainingsdaten gelernt werden ↑. Zur Entlastung der Anwendung erfolgt das Lernen regulär automatisch ↑. Prognosen werden dagegen von der Anwendung mit den Informationen darüber initiiert, was prognostiziert werden soll ↑. Um aus den lokalen Ergebnissen der Methoden der einzelnen Variablen zu einem Gesamtergebnis zu kommen, wird ein koordinierender Algorithmus eingesetzt ↑. Als Basis wird ein Algorithmus für bayes'sche Netze gewählt, weil er die Prognose von Wahrscheinlichkeitsverteilungen unterstützt, eine günstige Komplexität aufweist und die Realisierung höherwertiger Prognosemodi mit unterschiedlichen Arten von Prognoseergebnissen ermöglicht ↑ ↑.

Zur Beherrschung der Komplexität bei der Entwicklung werden Erweiterungen des Verfahrens, die verschiedene Schichten des Prognosesystems betreffen, separat im Anschluss an das bisher beschriebene Basissystem ausgearbeitet ↑. Die erste Erweiterung

schafft Möglichkeiten der Verteilung . Die zweite ermöglicht den Umgang mit Instanzen von Entitäten wie beispielsweise Personen, die in beliebiger Anzahl im Kontext erscheinen können . Die dritte Erweiterung zum verbesserten Umgang mit Unsicherheit dient der Verbesserung der Prognosegenauigkeit und der Angabe von Unsicherheit für die Anwendung bei Prognosen . Dazu werden zusätzliche Arten von Unsicherheit erkannt und es wird prinzipielle Unsicherheit von anderen Arten von Unsicherheit getrennt .

Die Implementierung des Verfahrens der strukturierten Kontextdatenprognose erfolgt in Kapitel 5. Sie beschränkt sich auf das Basissystem. Das Grundprinzip der Austauschbarkeit von Methoden wird softwaretechnisch durch eine Implementierung als Framework realisiert . Erwähnenswert ist außerdem die Einführung einer XML-Repräsentation für Prognosemodelle.

Nach der Implementierung des Prognosesystems selbst erfolgt der Einsatz desselben zur Prognose der Verfügbarkeit von Diensten für das DEMAC-Projekt . Dabei spielt ein Teil der DEMAC-Middleware die Rolle einer Anwendung, die das Prognosesystem benutzt, und es wird die Perspektive der Anwendungsentwickler eingenommen, die das Prognosesystem für die Anwendung einrichten möchten . Dazu wird zunächst nach dem ausgearbeiteten systematischen Vorgehen ein Prognosemodell erstellt . Über die Erstellung des Prognosemodells hinaus wird zusätzliche Funktionalität oberhalb des Prognosesystems implementiert . Der Hauptgrund dafür liegt darin, dass die Erweiterungen des Verfahrens der strukturierten Kontextdatenprognose nicht im Rahmen dieser Arbeit implementiert sind .

Kapitel 6 führt eine kurze Evaluation und eine Diskussion zur Bewertung des Verfahrens der strukturierten Kontextdatenprognose und dessen Implementierung durch. Zur Diskussion wird der in Kapitel 2 erstellte Anforderungskatalog herangezogen . Es stellt sich heraus, dass mit Ausnahme der Benutzbarkeit funktionale und nicht-funktionale Anforderungen in zufriedenstellendem Maße erfüllt sind bzw. durch Erweiterungen von Anwendungsentwicklern entsprechend den Bedürfnissen des Anwendungsbereichs erfüllt werden können . Die Benutzbarkeit für die Anwendungsentwickler wird durch den verhältnismäßig hohen Vorbereitungsaufwand zur Ermöglichung von Prognosen beeinträchtigt . Dafür wird jedoch entsprechend dem Ziel der Arbeit eine hohe Generik bei guter Prognosegenauigkeit und Effizienz erreicht .

7.2. Beiträge zur Forschung

Diese Arbeit enthält eine Reihe von Resultaten, die dem Wissen des Autors nach über den bisherigen Stand der Forschung im Bereich der Kontextdatenprognose hinausgehen. Im Wesentlichen werden die folgenden Ergebnisse zur Forschung beigetragen.

- Ausarbeitung eines umfassenden und detaillierten Anforderungskatalogs
- Systematische Unterscheidung von Unsicherheitsarten
- Entwicklung eines Verfahrens zur Kontextdatenprognose mit den folgenden herausragenden Eigenschaften:

  - Verbindung einer hohen Generik mit positiven nicht-funktionalen Charakteristika durch das Einbeziehen von Wissen über die Anwendungsdomäne und den hybriden Einsatz mehrerer Methoden als in diesem Ausmaß dem Wissen des Autors nach neuer Ansatz zur Kontextdatenprognose
  - Berücksichtigung der Anforderungen mobiler Geräte im kompletten Verfahren
  - Angaben und Behandlung von Unsicherheit in besonderem Maße
  - Unterstützung von adaptivem Online-Lernen in Kombination mit obigen Eigenschaften
- Prototypische Implementierung des Verfahrens als Framework, das Anwendungen Systemunterstützung zur Kontextdatenprognose bietet

7.3. Ausblick

Diese Arbeit schlägt einen Weg ein, zu dessen Fortsetzung an vielen Stellen Möglichkeiten bestehen. Es werden im Folgenden die wichtigsten davon aufgezeigt.

Die drei Erweiterungen des Verfahrens der strukturierten Kontextdatenprognose sollten im Zuge einer genaueren Ausarbeitung und Erprobung implementiert werden. Dadurch sind Verbesserungen beim Einsatz des Prognosesystems und der Genauigkeit zu erwarten ↑.

Für die Definition von Prognosenetzen kommt das Ergänzen höherwertiger elementarer Netzbestandteile in Frage ↑. Dadurch wird der Reichtum an Ausdrucksmitteln erhöht und die Erstellung von Prognosenetzen erleichtert.

Das Konzept der Nutzung mehrerer Variablen im Prognosenetz zur Darstellung verschiedener Repräsentationen von oder Prognosen über eine einzelne Variable in der Realität sollte noch einmal überdacht werden ↑. Eine Verfeinerung des Konzeptes oder ein optimierter Umgang damit durch das Prognosesystem können möglicherweise zu einer grundsätzlichen Lösung führen (vgl. Abschnitt 4.2.3 und 5.3.1). Dann würden sich die bisherigen Einzellösungen erübrigen, die nur dort ansetzen, wo die Probleme zu Tage treten.

Die Referenzkonfiguration verwendeter Methoden lässt noch viel Raum für weitere Methoden. Abschnitt 3.3.1, 3.3.4 und 3.4.1 liefern Ausgangspunkte für eine Implementierung weiterer Methoden, deren genauere Untersuchung in dieser Arbeit den Rahmen gesprengt hätte. Auch nach der Implementierung weiterer Messadapter besteht Bedarf ↑. Zur Unterstützung bei der Erstellung von Prognosemodellen durch die Anwendungsentwickler sollten Werkzeuge bereitgestellt werden ↑.

Die Möglichkeit vertikaler Verarbeitung, um High-Level-Kontext aus Low-Level-Kontext zu bestimmen, kann durch die Implementierung von Clustering als Methode angeboten werden. Dafür kommt die Verwendung der Ergebnisse von Mayrhofer in Frage. Der Anforderungskatalog zeigt weitere verbleibende Verbesserungsmöglichkeiten auf.

Nicht zuletzt stellt eine intensivere Evaluation des Verfahrens der strukturierten Kontextdatenprognose und dessen Implementierung eine wichtige Aufgabe dar. Dabei ist auch der Vergleich mit anderen Verfahren von Interesse.

Diese Arbeit stellt im Bereich der Kontextdatenprognose einen Vorstoß in eine neue Richtung dar und bietet somit viele Anknüpfungspunkte für weitere Arbeiten. Der Autor hofft, mit dieser Arbeit weitere Untersuchungen anregen zu können und so einen kleinen Beitrag zu einer verbesserten Unterstützung des Menschen in einer mit Informationstechnik angereicherten Welt zu leisten.

A

Beweis für die Zyklenfreiheit aufgefalteter Prognosenetze

Dieser Beweis bezieht sich auf die Definition von aufgefalteten Prognosenetzen (Definition 4.2) und die Definition von Prognosenetzen (Definition 4.1).

Behauptung: Jedes aufgefaltete Prognosenetz ist zyklenfrei.

Der Beweis erfolgt indirekt, indem das Gegenteil der Behauptung angenommen und zum Widerspruch geführt wird. Sei also $\mathcal{N}_\mathcal{A}$ ein aufgefaltetes Prognosenetz zu einem Prognosenetz $\mathcal{N}$. $\mathcal{N}_\mathcal{A}$ enthalte einen Zyklus. Dann gibt es einen Pfad $V_{i,j} \longrightarrow V_{i'_1,j'_1} \longrightarrow ... \longrightarrow V_{i'_p,j'_p} \longrightarrow V_{i,j}$ in $\mathcal{N}_\mathcal{A}$, der den Zyklus bildet. i'_0 und i'_{p+1} seien beide definiert als i. j'_0 und j'_{p+1} seien beide definiert als j.

Für zwei aufeinander folgende Variablen $V_{i'_k,j'_k}$ und $V_{i'_{k+1},j'_{k+1}}$ im Pfad gilt nach Definition 4.2, dass es eine Kante $V_{i'_k} \xrightarrow{\Delta} V_{i'_{k+1}}$ mit $\Delta = j'_{k+1} - j'_k$ in $\mathcal{N}$ gibt. Nach Definition 4.1 ist $j'_{k+1} - j'_k \geq 0$. Also muss $j'_{k+1} \geq j'_k$ gelten. Wenn man dem Pfad folgt, kann man also nur zu Variablen zum gleichen oder einem späteren als dem vorherigen Zeitpunkt gelangen, jedoch nie zu einem früheren.

Da der Pfad einen Zyklus bildet, der bei $V_{i,j}$ beginnt und bei $V_{i,j}$ endet, kommt man am Ende des Pfades zum gleichen Zeitpunkt, an dem man am Anfang war. Da man im Pfad nie zu einem früheren Zeitpunkt als dem vorherigen Zeitpunkt kommt, kann man auch nie zu einem späteren als dem vorherigen kommen, denn sonst könnte der Pfad nicht bei dem Zeitpunkt enden, bei dem er beginnt. Also müssen alle Zeitpunkte $j'_k = j$ sein. Dann gilt $j'_{k+1} - j'_k = 0$ für alle k und es gibt einen Pfad $V_i \xrightarrow{0} V_{i'_1} \xrightarrow{0} ... \xrightarrow{0} V_{i'_p} \xrightarrow{0} V_i$ in $\mathcal{N}$. Bei einem solchen Pfad handelt es sich um einen Zyklus mit 0 als Summe der Kantenbeschriftungen. Solche Zyklen sind jedoch nicht erlaubt in (nicht aufgefalteten) Prognosenetzen, so dass $\mathcal{N}$ kein Prognosenetz sein kann.

Dies ist ein Widerspruch zur Annahme, dass $\mathcal{N}$ ein (nicht aufgefaltetes) Prognosenetz ist. Damit ist die Behauptung bewiesen. □

XML-Hauptschema für Prognosemodelle

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xml:lang="en"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            finalDefault="#all"
            targetNamespace="http://contextprediction.vsis.org/predictionModel"
            xmlns="http://contextprediction.vsis.org/predictionModel"
            xmlns:methods="http://contextprediction.vsis.org/predictionModel/
                           methods"
            xmlns:measureAdapters="http://contextprediction.vsis.org/
                                   predictionModel/measureAdapters">

<!-- import method - specific and measure adapter - specific parts -->
<xsd:import namespace="http://contextprediction.vsis.org/predictionModel/
                       methods"
            schemaLocation="predictionModelMethods.xsd" />
<xsd:import namespace="http://contextprediction.vsis.org/predictionModel/
                       measureAdapters"
            schemaLocation="predictionModelMeasureAdapters.xsd" />

<!-- this schema defines the syntax of a prediction model as xml,
     the schema is not used at runtime because the small parser for mobile
     environments does not support this, it is implicitly incorporated into
     the program code, the schema can be used for validation, code-
     completion and reading documentation while creating a prediction model
     in an xml-editor, most constraints that are not captured by this
     schema are checked when loading the model into the prediction system,
     some constraints concerning adaptive online-learning are checked when
     starting learning,
     prediction models in xml-format should be utf-8-encoded because
     automatic detection of encoding is not supported by the parser, don't
     use entities in prediction models because they are not fully supported
     by the parser -->


<xsd:element name="predictionModel">
  <xsd:complexType>
    <xsd:sequence>

      <xsd:element name="generalSettings">
        <xsd:complexType>
          <xsd:attribute name="timeStepLength" type="xsd:positiveInteger"
                         use="required">
            <xsd:annotation><xsd:documentation>
              length of a time step in milliseconds, can't simply be changed
              after first learning because method previous knowledge instance
              data is based on the time step length
            </xsd:documentation></xsd:annotation>
```

```
48          </xsd:attribute>
49          <xsd:attribute name="recentHistoryLength" type="xsd:positiveInteger"
50                         use="required">
51            <xsd:annotation><xsd:documentation>
52              length of the history with recent data as given information
53              for predictions, expressed as number of time steps, values
54              older than timeStepLength*length are discarded, the length
55              determines how far a prediction can go back into the past to
56              reach given information, a long history can lead to slow
57              predictions if there are values missing in the history and
58              the prediction finds given information that are many time
59              steps before the present
60            </xsd:documentation></xsd:annotation>
61          </xsd:attribute>
62          <xsd:attribute name="recentHistoryFrequency"
63                         type="xsd:positiveInteger" use="required">
64            <xsd:annotation><xsd:documentation>
65               values of all variables are measured and stored in the recent
66               history recentHistoryFrequency times in a time step,
67               recentHistoryFrequency has to be a factor of timeStepLength,
68               i.e. timeStepLength \% recentHistoryFrequency = 0,
69               recentHistoryFrequency determines for predictions about a
70               given point of time how precise the point of time can be
71               chosen, e.g. timeStepLength=1200000 (20 minutes) and
72               recentHistoryFrequency=2 means that predictions can be made
73               about a point of time that differs 5 minutes from the target
74               point of time in the worst case
75            </xsd:documentation></xsd:annotation>
76          </xsd:attribute>
77        </xsd:complexType>
78      </xsd:element>
79
80      <xsd:element name="nominalTypeSet">
81        <xsd:complexType>
82          <xsd:sequence>
83            <xsd:element name="nominalType" minOccurs="0"
84                         maxOccurs="unbounded">
85              <xsd:annotation><xsd:documentation>
86                a type of variables with nominal scale that specifies the
87                allowed variable values
88              </xsd:documentation></xsd:annotation>
89              <xsd:complexType>
90                <xsd:sequence>
91                  <xsd:element name="allowedValue" minOccurs="1"
92                               maxOccurs="unbounded">
93                    <xsd:complexType>
94                      <xsd:attribute name="value" type="xsd:string"
95                                     use="required" />
96                    </xsd:complexType>
97                  </xsd:element>
98                </xsd:sequence>
99                <xsd:attribute name="name" type="xsd:string" use="required" />
100             </xsd:complexType>
101           </xsd:element>
```

```
102           </xsd:sequence>
103         </xsd:complexType>
104       </xsd:element>
105
106       <xsd:element name="measureAdapterSet">
107         <xsd:complexType>
108           <xsd:sequence>
109             <xsd:element name="measureAdapter" minOccurs="0"
110                          maxOccurs="unbounded">
111               <xsd:annotation><xsd:documentation>
112                   adapter for measuring a specific value (e.g. location, time
113                   or position) which does not have to be physical, e.g. it
114                   can also measure which programs are running
115               </xsd:documentation></xsd:annotation>
116               <xsd:complexType>
117                 <xsd:sequence>
118                   <xsd:element
119                          ref="measureAdapters:measureAdapterSpecificSettings">
120                     <xsd:annotation><xsd:documentation>
121                       settings that the specific measure adapter requires and
122                       that are defined by it (optionally with an xml-schema)
123                     </xsd:documentation></xsd:annotation>
124                   </xsd:element>
125                 </xsd:sequence>
126                 <xsd:attribute name="name" type="xsd:string" use="required">
127                   <xsd:annotation><xsd:documentation>
128                     unique name of the measure adapter
129                   </xsd:documentation></xsd:annotation>
130                 </xsd:attribute>
131                 <xsd:attribute name="factoryClass" type="xsd:string"
132                                use="required">
133                   <xsd:annotation><xsd:documentation>
134                     specifies how the measure adapter can be created and
135                     which implementation to use, has to be the
136                     fully-qualified class name, i.e. with package
137                   </xsd:documentation></xsd:annotation>
138                 </xsd:attribute>
139                 <xsd:attribute name="timeTolerance"
140                                type="xsd:nonNegativeInteger"
141                                use="required">
142                   <xsd:annotation><xsd:documentation>
143                     maximum number of milliseconds tolerated deviation for a
144                     measurement requested at a specific point of time, a
145                     delay can occur when multiple threads (e.g. for learning)
146                     want measurements at the same time or if the measurement
147                     takes some time for example, if the tolerance is exceeded,
148                     the measured value is dropped and the measurement fails
149                   </xsd:documentation></xsd:annotation>
150                 </xsd:attribute>
151               </xsd:complexType>
152             </xsd:element>
153           </xsd:sequence>
154         </xsd:complexType>
155       </xsd:element>
```

```
156
157     <xsd:element name="methodPreknowledgeSet">
158       <xsd:complexType>
159         <xsd:sequence>
160           <xsd:element name="methodPreknowledge" minOccurs="0"
161                        maxOccurs="unbounded">
162           <xsd:annotation><xsd:documentation>
163               previous knowledge for prediction of one variable as
164               dependent variable by a method, in most cases one entry
165               here is used by only one variable
166             </xsd:documentation></xsd:annotation>
167             <xsd:complexType>
168               <xsd:sequence>
169                 <xsd:element ref="methods:methodSpecificSettings">
170                   <xsd:annotation><xsd:documentation>
171                     settings that the specific method requires and that
172                     are defined by it (optionally with an xml-schema)
173                   </xsd:documentation></xsd:annotation>
174                 </xsd:element>
175               </xsd:sequence>
176               <xsd:attribute name="name" type="xsd:string" use="required">
177                 <xsd:annotation><xsd:documentation>
178                   unique name of the previous knowledge
179                 </xsd:documentation></xsd:annotation>
180               </xsd:attribute>
181               <xsd:attribute name="factoryClass" type="xsd:string"
182                              use="required">
183                 <xsd:annotation><xsd:documentation>
184                   specifies how method previous knowledge can be created
185                   and to which method it belongs, has to be the fully-
186                   qualified class name, i.e. with package
187                 </xsd:documentation></xsd:annotation>
188               </xsd:attribute>
189               <xsd:attribute name="nominalTypeDependentVariable"
190                              type="xsd:string" use="optional">
191                 <xsd:annotation><xsd:documentation>
192                   name of the nominal type of the dependent variable in case
193                   of a method for a dependent variable with nominal scale
194                 </xsd:documentation></xsd:annotation>
195               </xsd:attribute>
196             </xsd:complexType>
197           </xsd:element>
198         </xsd:sequence>
199       </xsd:complexType>
200     </xsd:element>
201
202     <xsd:element name="variableSet">
203       <xsd:complexType>
204         <xsd:sequence>
205           <xsd:element name="variable" minOccurs="0" maxOccurs="unbounded">
206             <xsd:complexType>
207               <xsd:sequence>
208                 <xsd:element name="parentEdge" minOccurs="0"
209                              maxOccurs="unbounded">
```

```
                    <xsd:annotation><xsd:documentation>
                        a parentEdge for each edge from a parent of this
                        variable as independent variable, cycles of variables
                        with time offset 0 on every edge are not allowed!,
                        there is no cycle detection that produces warnings
                    </xsd:documentation></xsd:annotation>
                    <xsd:complexType>
                      <xsd:attribute name="variable" type="xsd:string"
                                     use="required">
                          <xsd:annotation><xsd:documentation>
                            the name of the parent-variable
                          </xsd:documentation></xsd:annotation>
                      </xsd:attribute>
                      <xsd:attribute name="timeOffset"
                                     type="xsd:nonNegativeInteger"
                                     use="required">
                        <xsd:annotation><xsd:documentation>
                            the time offset in number of time steps,
                            e.g. variable Y depends on the value of
                            variable X 3 time steps earlier
                          </xsd:documentation></xsd:annotation>
                      </xsd:attribute>
                    </xsd:complexType>
                  </xsd:element>
                </xsd:sequence>
                <xsd:attribute name="name" type="xsd:string" use="required">
                  <xsd:annotation><xsd:documentation>
                    unique name of the variable
                  </xsd:documentation></xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="scale" type="scaleType" use="required">
                  <xsd:annotation><xsd:documentation>
                    the scale of the variable
                  </xsd:documentation></xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="nominalType" type="xsd:string"
                               use="optional">
                  <xsd:annotation><xsd:documentation>
                    name of the nominal type in case of a variable with
                    nominal scale
                  </xsd:documentation></xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="methodPreknowledgeWithIndep"
                               type="xsd:string" use="optional">
                  <xsd:annotation><xsd:documentation>
                    name of previous knowledge for prediction of variable as
                    dependent variable by a method using the values of the
                    independent variables, should be omitted if and only if
                    the variable has no parents
                  </xsd:documentation></xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="methodPreknowledgeNoIndep"
                               type="xsd:string" use="required">
                  <xsd:annotation><xsd:documentation>
```

```
264                   name of previous knowledge for prediction of variable
265                   as dependent variable by a method without using the
266                   (for some predictions possibly unknown) values of the
267                   independent variables
268                 </xsd:documentation></xsd:annotation>
269               </xsd:attribute>
270               <xsd:attribute name="measureAdapter" type="xsd:string"
271                              use="optional">
272                 <xsd:annotation><xsd:documentation>
273                   name of measure adapter for measuring values of this
274                   variable, may be omitted if no measuring possible
275                 </xsd:documentation></xsd:annotation>
276               </xsd:attribute>
277               <xsd:attribute name="defaultLearningDistance"
278                              type="xsd:positiveInteger" use="required">
279                 <xsd:annotation><xsd:documentation>
280                   milliseconds between the beginning of measuring and
281                   learning a dataset consisting of a value combination
282                   of the dependent variable and the independent variables,
283                   the learning distance has to be bigger then the biggest
284                   time offset of the parents
285                 </xsd:documentation></xsd:annotation>
286               </xsd:attribute>
287               <xsd:attribute name="learningMethodPreknowlWithIndep"
288                              type="xsd:boolean" use="required">
289                 <xsd:annotation><xsd:documentation>
290                   indicates if automatic online-learning is switched on
291                   for methodPreknowledgeWithIndep
292                 </xsd:documentation></xsd:annotation>
293               </xsd:attribute>
294               <xsd:attribute name="learningMethodPreknowlNoIndep"
295                              type="xsd:boolean" use="required">
296                 <xsd:annotation><xsd:documentation>
297                   indicates if automatic online-learning is switched on
298                   for methodPreknowledgeNoIndep
299                 </xsd:documentation></xsd:annotation>
300               </xsd:attribute>
301             </xsd:complexType>
302           </xsd:element>
303         </xsd:sequence>
304       </xsd:complexType>
305     </xsd:element>
306   </xsd:sequence>
307 </xsd:complexType>
308
309 <!-- keys and references -->
310 <xsd:key name="measureAdapterNameAsKey">
311   <xsd:selector xpath="measureAdapterSet/measureAdapter"/>
312   <xsd:field xpath="@name"/>
313 </xsd:key>
314 <xsd:key name="nominalTypeNameAsKey">
315   <xsd:selector xpath="nominalTypeSet/nominalType"/>
316   <xsd:field xpath="@name"/>
317 </xsd:key>
```

```
318   <xsd:key name="methodPreknowledgeNameAsKey">
319     <xsd:selector xpath="methodPreknowledgeSet/methodPreknowledge"/>
320     <xsd:field xpath="@name"/>
321   </xsd:key>
322   <xsd:key name="variableNameAsKey">
323     <xsd:selector xpath="variableSet/variable"/>
324     <xsd:field xpath="@name"/>
325   </xsd:key>
326   <xsd:keyref name="nominalTypeReferenceFromMethodPreknowledge"
327               refer="nominalTypeNameAsKey">
328     <xsd:selector xpath="methodPreknowledgeSet/methodPreknowledge"/>
329     <xsd:field xpath="@nominalTypeDependentVariable"/>
330   </xsd:keyref>
331   <xsd:keyref name="nominalTypeReferenceFromVariable"
332               refer="nominalTypeNameAsKey">
333     <xsd:selector xpath="variableSet/variable"/>
334     <xsd:field xpath="@nominalType"/>
335   </xsd:keyref>
336   <xsd:keyref name="methodPreknowledgeWithIndepReferenceFromVariable"
337               refer="methodPreknowledgeNameAsKey">
338     <xsd:selector xpath="variableSet/variable"/>
339     <xsd:field xpath="@methodPreknowledgeWithIndep"/>
340   </xsd:keyref>
341   <xsd:keyref name="methodPreknowledgeNoIndepReferenceFromVariable"
342               refer="methodPreknowledgeNameAsKey">
343     <xsd:selector xpath="variableSet/variable"/>
344     <xsd:field xpath="@methodPreknowledgeNoIndep"/>
345   </xsd:keyref>
346   <xsd:keyref name="measureAdapterReferenceFromVariable"
347               refer="measureAdapterNameAsKey">
348     <xsd:selector xpath="variableSet/variable"/>
349     <xsd:field xpath="@measureAdapter"/>
350   </xsd:keyref>
351   <xsd:keyref name="parentReferenceFromVariable"
352               refer="variableNameAsKey">
353     <xsd:selector xpath="variableSet/variable/parentEdge"/>
354     <xsd:field xpath="@variable"/>
355   </xsd:keyref>
356
357 </xsd:element>
358
359
360 <xsd:simpleType name="scaleType">
361   <xsd:restriction base="xsd:string">
362     <xsd:enumeration value="nominal" />
363     <xsd:enumeration value="ratio" />
364   </xsd:restriction>
365 </xsd:simpleType>
366
367
368 <xsd:complexType name="componentSpecificSettingsType" final="extension">
369   <xsd:annotation><xsd:documentation>
370     a type for data only belonging to components that can be added to the
371     prediction system like plugins (e.g. methods and measure adapters)
```

```
372   </xsd:documentation></xsd:annotation>
373   <xsd:sequence>
374     <!-- the property1, ..., property9 - elements are a workaround because
375          XSD seems not to allow elements with the same name and different
376          types in a sequence, the property1, ..., property9 - elements can
377          be used instead of multiple property - elements in a sequence if
378          the intended properties have a different fixed name or different
379          value-types or different subelements, the property - element can
380          be used if a list with unbounded number of entries of the same
381          kind is needed -->
382     <xsd:element name="property
383                  type="componentSpecificStructuredSettingsType"
384                  minOccurs="0" maxOccurs="unbounded" />
385     <xsd:element name="property1"
386                  type="componentSpecificStructuredSettingsType"
387                  minOccurs="0" maxOccurs="unbounded" />
388     <xsd:element name="property2"
389                  type="componentSpecificStructuredSettingsType"
390                  minOccurs="0" maxOccurs="unbounded" />
391     <xsd:element name="property3"
392                  type="componentSpecificStructuredSettingsType"
393                  minOccurs="0" maxOccurs="unbounded" />
394     <xsd:element name="property4"
395                  type="componentSpecificStructuredSettingsType"
396                  minOccurs="0" maxOccurs="unbounded" />
397     <xsd:element name="property5"
398                  type="componentSpecificStructuredSettingsType"
399                  minOccurs="0" maxOccurs="unbounded" />
400     <xsd:element name="property6"
401                  type="componentSpecificStructuredSettingsType"
402                  minOccurs="0" maxOccurs="unbounded" />
403     <xsd:element name="property7"
404                  type="componentSpecificStructuredSettingsType"
405                  minOccurs="0" maxOccurs="unbounded" />
406     <xsd:element name="property8"
407                  type="componentSpecificStructuredSettingsType"
408                  minOccurs="0" maxOccurs="unbounded" />
409     <xsd:element name="property9"
410                  type="componentSpecificStructuredSettingsType"
411                  minOccurs="0" maxOccurs="unbounded" />
412   </xsd:sequence>
413   <!-- unstructured data can be represented as attributes -->
414   <xsd:anyAttribute />
415 </xsd:complexType>
416 <xsd:complexType name="componentSpecificStructuredSettingsType"
417                  final="extension">
418   <xsd:sequence>
419     <xsd:element name="property"
420                  type="componentSpecificStructuredSettingsType"
421                  minOccurs="0" maxOccurs="unbounded" />
422     <xsd:element name="property1"
423                  type="componentSpecificStructuredSettingsType"
424                  minOccurs="0" maxOccurs="unbounded" />
425     <xsd:element name="property2"
```

```
                type="componentSpecificStructuredSettingsType"
                minOccurs="0" maxOccurs="unbounded" />
    <xsd:element name="property3"
                type="componentSpecificStructuredSettingsType"
                minOccurs="0" maxOccurs="unbounded" />
    <xsd:element name="property4"
                type="componentSpecificStructuredSettingsType"
                minOccurs="0" maxOccurs="unbounded" />
    <xsd:element name="property5"
                type="componentSpecificStructuredSettingsType"
                minOccurs="0" maxOccurs="unbounded" />
    <xsd:element name="property6"
                type="componentSpecificStructuredSettingsType"
                minOccurs="0" maxOccurs="unbounded" />
    <xsd:element name="property7"
                type="componentSpecificStructuredSettingsType"
                minOccurs="0" maxOccurs="unbounded" />
    <xsd:element name="property8"
                type="componentSpecificStructuredSettingsType"
                minOccurs="0" maxOccurs="unbounded" />
    <xsd:element name="property9"
                type="componentSpecificStructuredSettingsType"
                minOccurs="0" maxOccurs="unbounded" />
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required" />
  <xsd:attribute name="value" type="xsd:anySimpleType" use="optional" />
</xsd:complexType>


</xsd:schema>
```

Listing B.1: XML-Hauptschema für Prognosemodelle

C

XML-Schema für Wahrscheinlichkeitstabellen

```
<xsd:element name="probabilityTableSettings"
             substitutionGroup="methodSpecificSettings">
    <xsd:complexType><xsd:complexContent>
    <xsd:restriction base="modelFixed:componentSpecificSettingsType">
    <xsd:sequence>
    <xsd:element name="property" minOccurs="1" maxOccurs="1">
        <xsd:annotation><xsd:documentation>
          nominal types of the independent variables, have to be in the same
          order as the independent variables as parents in the definition of
          the dependent variable
        </xsd:documentation></xsd:annotation>
        <xsd:complexType><xsd:complexContent>
        <xsd:restriction
                 base="modelFixed:componentSpecificStructuredSettingsType">
        <xsd:sequence>
        <xsd:element name="property" minOccurs="0" maxOccurs="unbounded">
            <xsd:complexType><xsd:complexContent>
            <xsd:restriction
                   base="modelFixed:componentSpecificStructuredSettingsType">
            <xsd:attribute name="name" type="xsd:string" use="required"
                           fixed="nominalType" />
            <xsd:attribute name="value" type="xsd:string" use="required">
              <xsd:annotation><xsd:documentation>
                name of the nominal type
              </xsd:documentation></xsd:annotation>
            </xsd:attribute>
            </xsd:restriction>
            </xsd:complexContent></xsd:complexType>
        </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string" use="required"
                       fixed="indepVarNominalTypes" />
        </xsd:restriction>
        </xsd:complexContent></xsd:complexType>
    </xsd:element>
    </xsd:sequence>
    </xsd:restriction>
    </xsd:complexContent></xsd:complexType>
</xsd:element>
```

Listing C.1: XML-Schema für die Wahrscheinlichkeitstabellen-Methode in Prognosemodellen als Beispiel für ein methodenspezifisches Schema

Prognosemodell zur Prognose der Dienstverfügbarkeit

```
<?xml version="1.0" encoding="UTF-8"?>
<mdl:predictionModel xmlns:mdl="http://contextprediction.vsis.org/
                                predictionModel"
                     xmlns:mdlma="http://contextprediction.vsis.org/
                                  predictionModel/measureAdapters"
                     xmlns:mdlme="http://contextprediction.vsis.org/
                                  predictionModel/methods">


  <generalSettings recentHistoryFrequency="4" recentHistoryLength="6"
                   timeStepLength="900000"/>


  <nominalTypeSet>

    <nominalType name="timeOfDay">
      <allowedValue value="0-3"/>
      <allowedValue value="3-6"/>
      <allowedValue value="6-7"/>
      <allowedValue value="7-8"/>
      <allowedValue value="8-9"/>
      <allowedValue value="9-10"/>
      <allowedValue value="10-11"/>
      <allowedValue value="11-12"/>
      <allowedValue value="12-13"/>
      <allowedValue value="13-14"/>
      <allowedValue value="14-15"/>
      <allowedValue value="15-16"/>
      <allowedValue value="16-17"/>
      <allowedValue value="17-18"/>
      <allowedValue value="18-19"/>
      <allowedValue value="19-20"/>
      <allowedValue value="20-21"/>
      <allowedValue value="21-22"/>
      <allowedValue value="22-23"/>
      <allowedValue value="23-24"/>
    </nominalType>

    <nominalType name="place">
      <allowedValue value="Informatikum"/>
      <allowedValue value="HH Innenstadt"/>
      <allowedValue value="Zu Hause"/>
      <allowedValue value="Route: Zu Hause - Köhlbrandbrücke"/>
      <allowedValue value="Route: Köhlbrandbrücke - HH Innenstadt"/>
      <allowedValue value="Route: Köhlbrandbrücke - Informatikum"/>
```

```
      <allowedValue value="Supermarkt"/>
      <allowedValue value="Sportclub"/>
      <allowedValue value="unknown"/>
    </nominalType>

    <nominalType name="boolean">
      <allowedValue value="false"/>
      <allowedValue value="true"/>
    </nominalType>

  </nominalTypeSet>



  <measureAdapterSet>

    <!-- time in milliseconds since 1970 -->
    <measureAdapter name="timeMillis"
                    factoryClass="org.vsis.contextprediction.dataacquisition.
                          measureadapters.TimeMeasureAdapterFactory"
                    timeTolerance="5000">
      <mdlma:measureAdapterSpecificSettings/>
    </measureAdapter>

    <!-- x-component of position as coordinates on earth -->
    <measureAdapter name="positionCoordX"
                    factoryClass="org.vsis.contextprediction.dataacquisition.
                         measureadapters.PositionMeasureAdapterFactory"
                    timeTolerance="180000">
      <mdlma:positionMeasureSpecificSettings coordinatesPart="x" />
    </measureAdapter>

    <!-- y-component of position as coordinates on earth -->
    <measureAdapter name="positionCoordY"
                    factoryClass="org.vsis.contextprediction.dataacquisition.
                         measureadapters.PositionMeasureAdapterFactory"
                    timeTolerance="180000">
      <mdlma:positionMeasureSpecificSettings coordinatesPart="y" />
    </measureAdapter>

    <!-- discrete place (e.g. at home) -->
    <measureAdapter name="place"
                    factoryClass="org.demac.impl.j2me.context.registry.
                         prediction.measureadapters.PlaceMeasureAdapterFactory"
                    timeTolerance="180000">
      <mdlma:measureAdapterSpecificSettings/>
    </measureAdapter>

    <!-- size of network = number of devices in network -->
    <measureAdapter name="netSize"
                    factoryClass="org.demac.impl.j2me.context.registry.
                       prediction.measureadapters.NetSizeMeasureAdapterFactory"
                    timeTolerance="10000">
      <mdlma:measureAdapterSpecificSettings/>
```

```
100     </measureAdapter>
101
102   </measureAdapterSet>
103
104
105
106   <methodPreknowledgeSet>
107
108     <!-- method previous knowledge for time -->
109
110     <methodPreknowledge name="timeMillisIncrement"
111                         factoryClass="org.vsis.contextprediction.preknowledge.
112                             methods.specialized.TimeIncrementPreknowlFactory">
113       <mdlme:methodSpecificSettings />
114     </methodPreknowledge>
115
116     <methodPreknowledge name="timeMillisToHours"
117                         factoryClass="org.vsis.contextprediction.preknowledge.
118                             methods.specialized.TimePeriodsPreknowlFactory">
119       <mdlme:timePeriodsSettings targetTimeUnit="hourOfDay"/>
120     </methodPreknowledge>
121
122     <methodPreknowledge name="timeHoursRatioToNomi"
123                         factoryClass="org.vsis.contextprediction.preknowledge.
124                             methods.specialized.BinningPreknowlFactory"
125                         nominalTypeDependentVariable="timeOfDay">
126       <mdlme:binningSettings>
127         <property name="intervalToBinMappings">
128           <property1 name="-infinityUntilNext">
129             <property1 name="bin" value="0-3"/>
130           </property1>
131           <property2 name="lowerBoundUntilNext">
132             <property1 name="lowerBound" value="3"/>
133             <property2 name="bin" value="3-6"/>
134           </property2>
135           <property2 name="lowerBoundUntilNext">
136             <property1 name="lowerBound" value="6"/>
137             <property2 name="bin" value="6-7"/>
138           </property2>
139           <property2 name="lowerBoundUntilNext">
140             <property1 name="lowerBound" value="7"/>
141             <property2 name="bin" value="7-8"/>
142           </property2>
143           <property2 name="lowerBoundUntilNext">
144             <property1 name="lowerBound" value="8"/>
145             <property2 name="bin" value="8-9"/>
146           </property2>
147           <property2 name="lowerBoundUntilNext">
148             <property1 name="lowerBound" value="9"/>
149             <property2 name="bin" value="9-10"/>
150           </property2>
151           <property2 name="lowerBoundUntilNext">
152             <property1 name="lowerBound" value="10"/>
153             <property2 name="bin" value="10-11"/>
```

```
154           </property2>
155           <property2 name="lowerBoundUntilNext">
156             <property1 name="lowerBound" value="11"/>
157             <property2 name="bin" value="11-12"/>
158           </property2>
159           <property2 name="lowerBoundUntilNext">
160             <property1 name="lowerBound" value="12"/>
161             <property2 name="bin" value="12-13"/>
162           </property2>
163           <property2 name="lowerBoundUntilNext">
164             <property1 name="lowerBound" value="13"/>
165             <property2 name="bin" value="13-14"/>
166           </property2>
167           <property2 name="lowerBoundUntilNext">
168             <property1 name="lowerBound" value="14"/>
169             <property2 name="bin" value="14-15"/>
170           </property2>
171           <property2 name="lowerBoundUntilNext">
172             <property1 name="lowerBound" value="15"/>
173             <property2 name="bin" value="15-16"/>
174           </property2>
175           <property2 name="lowerBoundUntilNext">
176             <property1 name="lowerBound" value="16"/>
177             <property2 name="bin" value="16-17"/>
178           </property2>
179           <property2 name="lowerBoundUntilNext">
180             <property1 name="lowerBound" value="17"/>
181             <property2 name="bin" value="17-18"/>
182           </property2>
183           <property2 name="lowerBoundUntilNext">
184             <property1 name="lowerBound" value="18"/>
185             <property2 name="bin" value="18-19"/>
186           </property2>
187           <property2 name="lowerBoundUntilNext">
188             <property1 name="lowerBound" value="19"/>
189             <property2 name="bin" value="19-20"/>
190           </property2>
191           <property2 name="lowerBoundUntilNext">
192             <property1 name="lowerBound" value="20"/>
193             <property2 name="bin" value="20-21"/>
194           </property2>
195           <property2 name="lowerBoundUntilNext">
196             <property1 name="lowerBound" value="21"/>
197             <property2 name="bin" value="21-22"/>
198           </property2>
199           <property2 name="lowerBoundUntilNext">
200             <property1 name="lowerBound" value="22"/>
201             <property2 name="bin" value="22-23"/>
202           </property2>
203           <property3 name="lowerBoundUntilInfinity">
204             <property1 name="lowerBound" value="23"/>
205             <property2 name="bin" value="23-24"/>
206           </property3>
207         </property>
```

```
208       </mdlme:binningSettings>
209     </methodPreknowledge>
210
211     <!-- the following two method previous knowledge definitions should
212          normally never be used for prediction by the prediction system -->
213     <methodPreknowledge name="timeRatioFallback"
214                         factoryClass="org.vsis.contextprediction.preknowledge.
215                             methods.universal.EmpiricalDistribPreknowlFactory">
216       <mdlme:empiricalDistributionSettings maxCapacity="1"/>
217     </methodPreknowledge>
218     <methodPreknowledge name="timeNominalFallback"
219                         factoryClass="org.vsis.contextprediction.preknowledge.
220                             methods.universal.ProbabTablePreknowlFactory"
221                         nominalTypeDependentVariable="timeOfDay">
222       <mdlme:probabilityTableSettings>
223         <property name="indepVarNominalTypes" />
224       </mdlme:probabilityTableSettings>
225     </methodPreknowledge>
226
227
228     <!-- method previous knowledge for position -->
229
230     <methodPreknowledge name="placeNoIndep"
231                         factoryClass="org.vsis.contextprediction.preknowledge.
232                             methods.universal.ProbabTablePreknowlFactory"
233                         nominalTypeDependentVariable="place">
234       <mdlme:probabilityTableSettings>
235         <property name="indepVarNominalTypes" />
236       </mdlme:probabilityTableSettings>
237     </methodPreknowledge>
238
239     <methodPreknowledge name="placeByTime"
240                         factoryClass="org.vsis.contextprediction.preknowledge.
241                             methods.universal.ProbabTablePreknowlFactory"
242                         nominalTypeDependentVariable="place">
243       <mdlme:probabilityTableSettings>
244         <property name="indepVarNominalTypes">
245           <property name="nominalType" value="timeOfDay"/>
246         </property>
247       </mdlme:probabilityTableSettings>
248     </methodPreknowledge>
249
250     <methodPreknowledge name="placeByPlaceBefore"
251                         factoryClass="org.vsis.contextprediction.preknowledge.
252                             methods.universal.ProbabTablePreknowlFactory"
253                         nominalTypeDependentVariable="place">
254       <mdlme:probabilityTableSettings>
255         <property name="indepVarNominalTypes">
256           <property name="nominalType" value="place"/>
257         </property>
258       </mdlme:probabilityTableSettings>
259     </methodPreknowledge>
260
261     <methodPreknowledge name="positionCoordNoIndep"
```

```
                        factoryClass="org.vsis.contextprediction.preknowledge.
                            methods.universal.EmpiricalDistribPreknowlFactory">
      <mdlme:empiricalDistributionSettings maxCapacity="250"/>
    </methodPreknowledge>

    <methodPreknowledge name="positionCoordExtrapolation"
                        factoryClass="org.vsis.contextprediction.preknowledge.
                            methods.specialized.
                            PolynomialExtrapolationPreknowlFactory">
      <mdlme:methodSpecificSettings />
    </methodPreknowledge>

    <methodPreknowledge name="positionCoordToPlace"
                        factoryClass="org.demac.impl.j2me.context.registry.
                           prediction.methods.ModifiedBinning2DPreknowlFactory"
                        nominalTypeDependentVariable="place">
      <mdlme:binning2DSettings defaultBin="unknown">
        <property name="rectangleToBinMappings" />
      </mdlme:binning2DSettings>
    </methodPreknowledge>

    <methodPreknowledge name="placeResultIntegration"
                        factoryClass="org.vsis.contextprediction.preknowledge.
                            methods.specialized.MajorityVotePreknowlFactory"
                        nominalTypeDependentVariable="place">
      <mdlme:majorityVoteSettings numIndepVars="3" badValue="unknown" />
    </methodPreknowledge>


    <!-- method previous knowledge for net size (= number of devices
         in network) -->

    <methodPreknowledge name="netSizeNoIndep"
                        factoryClass="org.vsis.contextprediction.preknowledge.
                            methods.universal.EmpiricalDistribPreknowlFactory">
      <mdlme:empiricalDistributionSettings maxCapacity="250"/>
    </methodPreknowledge>

    <methodPreknowledge name="netSizeExtrapolation"
                        factoryClass="org.vsis.contextprediction.preknowledge.
                            methods.specialized.
                            PolynomialExtrapolationPreknowlFactory">
      <mdlme:methodSpecificSettings />
    </methodPreknowledge>


    <!-- method previous knowledge for service availability -->

    <methodPreknowledge name="serviceAvailableTendencyFallback"
                        factoryClass="org.vsis.contextprediction.preknowledge.
                            methods.universal.EmpiricalDistribPreknowlFactory">
      <mdlme:empiricalDistributionSettings maxCapacity="1"/>
    </methodPreknowledge>

```

```
316     <methodPreknowledge name="serviceAvailabilityFallback"
317                         factoryClass="org.vsis.contextprediction.preknowledge.
318                             methods.universal.ProbabTablePreknowlFactory"
319                         nominalTypeDependentVariable="boolean">
320       <mdlme:probabilityTableSettings>
321         <property name="indepVarNominalTypes" />
322       </mdlme:probabilityTableSettings>
323     </methodPreknowledge>
324
325     <methodPreknowledge name="serviceAvailabilityByTime"
326                         factoryClass="org.vsis.contextprediction.preknowledge.
327                             methods.universal.ProbabTablePreknowlFactory"
328                         nominalTypeDependentVariable="boolean">
329       <mdlme:probabilityTableSettings>
330         <property name="indepVarNominalTypes">
331           <property name="nominalType" value="timeOfDay"/>
332         </property>
333       </mdlme:probabilityTableSettings>
334     </methodPreknowledge>
335
336     <methodPreknowledge name="serviceAvailabilityByPlace"
337                         factoryClass="org.vsis.contextprediction.preknowledge.
338                             methods.universal.ProbabTablePreknowlFactory"
339                         nominalTypeDependentVariable="boolean">
340       <mdlme:probabilityTableSettings>
341         <property name="indepVarNominalTypes">
342           <property name="nominalType" value="place"/>
343         </property>
344       </mdlme:probabilityTableSettings>
345     </methodPreknowledge>
346
347     <methodPreknowledge name="serviceAvailabilityByTimeAndPlace"
348                         factoryClass="org.vsis.contextprediction.preknowledge.
349                             methods.universal.ProbabTablePreknowlFactory"
350                         nominalTypeDependentVariable="boolean">
351       <mdlme:probabilityTableSettings>
352         <property name="indepVarNominalTypes">
353           <property name="nominalType" value="timeOfDay"/>
354           <property name="nominalType" value="place"/>
355         </property>
356       </mdlme:probabilityTableSettings>
357     </methodPreknowledge>
358
359     <methodPreknowledge name="serviceAvailableTendencyByNetSize"
360                         factoryClass="org.vsis.contextprediction.preknowledge.
361                             methods.universal.RegressionPreknowlFactory">
362       <mdlme:methodSpecificSettings/>
363     </methodPreknowledge>
364
365     <methodPreknowledge name="serviceAvailabilityByNetSize"
366                         factoryClass="org.vsis.contextprediction.preknowledge.
367                             methods.specialized.BinningPreknowlFactory"
368                         nominalTypeDependentVariable="boolean">
369       <mdlme:binningSettings>
```

```
370         <property name="intervalToBinMappings">
371           <property1 name="-infinityUntilNext">
372             <property1 name="bin" value="false"/>
373           </property1>
374           <property3 name="lowerBoundUntilInfinity">
375             <property1 name="lowerBound" value="0.5"/>
376             <property2 name="bin" value="true"/>
377           </property3>
378         </property>
379       </mdlme:binningSettings>
380     </methodPreknowledge>
381
382     <methodPreknowledge name="serviceAvailabilityResultIntegrate"
383                         factoryClass="org.vsis.contextprediction.preknowledge.
384                             methods.specialized.MajorityVotePreknowlFactory"
385                         nominalTypeDependentVariable="boolean">
386       <mdlme:majorityVoteSettings numIndepVars="4" />
387     </methodPreknowledge>
388
389   </methodPreknowledgeSet>
390
391
392
393   <variableSet>
394
395     <!-- variables for time -->
396     <variable name="timeMillis" scale="ratio"
397               defaultLearningDistance="1000000"
398               learningMethodPreknowlNoIndep="false"
399               learningMethodPreknowlWithIndep="false"
400               measureAdapter="timeMillis"
401               methodPreknowledgeNoIndep="timeRatioFallback"
402               methodPreknowledgeWithIndep="timeMillisIncrement">
403       <parentEdge timeOffset="1" variable="timeMillis"/>
404     </variable>
405     <variable name="timeOfDayRatio" scale="ratio"
406               defaultLearningDistance="1000000"
407               learningMethodPreknowlNoIndep="false"
408               learningMethodPreknowlWithIndep="false"
409               methodPreknowledgeNoIndep="timeRatioFallback"
410               methodPreknowledgeWithIndep="timeMillisToHours">
411       <parentEdge timeOffset="0" variable="timeMillis"/>
412     </variable>
413     <variable name="timeOfDayNomi" scale="nominal" nominalType="timeOfDay"
414               defaultLearningDistance="1000000"
415               learningMethodPreknowlNoIndep="false"
416               learningMethodPreknowlWithIndep="false"
417               methodPreknowledgeNoIndep="timeNominalFallback"
418               methodPreknowledgeWithIndep="timeHoursRatioToNomi">
419       <parentEdge timeOffset="0" variable="timeOfDayRatio"/>
420     </variable>
421
422     <!-- variables for position -->
423     <variable name="placeByTime" scale="nominal" nominalType="place"
```

```
424             measureAdapter="place"
425             defaultLearningDistance="900000"
426             learningMethodPreknowlNoIndep="false"
427             learningMethodPreknowlWithIndep="true"
428             methodPreknowledgeNoIndep="placeNoIndep"
429             methodPreknowledgeWithIndep="placeByTime">
430       <parentEdge timeOffset="0" variable="timeOfDayNomi"/>
431     </variable>
432     <variable name="placeByPlaceBefore" scale="nominal" nominalType="place"
433             measureAdapter="place"
434             defaultLearningDistance="1000000"
435             learningMethodPreknowlNoIndep="false"
436             learningMethodPreknowlWithIndep="true"
437             methodPreknowledgeNoIndep="placeNoIndep"
438             methodPreknowledgeWithIndep="placeByPlaceBefore">
439       <parentEdge timeOffset="1" variable="placeIntegratedResults"/>
440     </variable>
441     <variable name="positionCoordXExtrapolated" scale="ratio"
442             measureAdapter="positionCoordX"
443             defaultLearningDistance="3000000"
444             learningMethodPreknowlNoIndep="false"
445             learningMethodPreknowlWithIndep="false"
446             methodPreknowledgeNoIndep="positionCoordNoIndep"
447             methodPreknowledgeWithIndep="positionCoordExtrapolation">
448       <parentEdge timeOffset="3" variable="positionCoordXExtrapolated"/>
449       <parentEdge timeOffset="2" variable="positionCoordXExtrapolated"/>
450       <parentEdge timeOffset="1" variable="positionCoordXExtrapolated"/>
451     </variable>
452     <variable name="positionCoordYExtrapolated" scale="ratio"
453             measureAdapter="positionCoordY"
454             defaultLearningDistance="3000000"
455             learningMethodPreknowlNoIndep="false"
456             learningMethodPreknowlWithIndep="false"
457             methodPreknowledgeNoIndep="positionCoordNoIndep"
458             methodPreknowledgeWithIndep="positionCoordExtrapolation">
459       <parentEdge timeOffset="3" variable="positionCoordXExtrapolated"/>
460       <parentEdge timeOffset="2" variable="positionCoordXExtrapolated"/>
461       <parentEdge timeOffset="1" variable="positionCoordXExtrapolated"/>
462     </variable>
463     <variable name="placeExtrapolated" scale="nominal" nominalType="place"
464             measureAdapter="place"
465             defaultLearningDistance="900000"
466             learningMethodPreknowlNoIndep="false"
467             learningMethodPreknowlWithIndep="false"
468             methodPreknowledgeNoIndep="placeNoIndep"
469             methodPreknowledgeWithIndep="positionCoordToPlace">
470       <parentEdge timeOffset="0" variable="positionCoordXExtrapolated"/>
471       <parentEdge timeOffset="0" variable="positionCoordYExtrapolated"/>
472     </variable>
473     <variable name="placeIntegratedResults" scale="nominal" nominalType="place"
474             measureAdapter="place"
475             defaultLearningDistance="900000"
476             learningMethodPreknowlNoIndep="true"
477             learningMethodPreknowlWithIndep="false"
```

```
478             methodPreknowledgeNoIndep="placeNoIndep"
479             methodPreknowledgeWithIndep="placeResultIntegration">
480       <parentEdge timeOffset="0" variable="placeByTime"/>
481       <parentEdge timeOffset="0" variable="placeByPlaceBefore"/>
482       <parentEdge timeOffset="0" variable="placeExtrapolated"/>
483     </variable>
484
485     <!-- variable for net size (= number of devices in network) -->
486    <variable name="netSize" scale="ratio"
487             measureAdapter="netSize"
488             defaultLearningDistance="3000000"
489             learningMethodPreknowlNoIndep="true"
490             learningMethodPreknowlWithIndep="false"
491             methodPreknowledgeNoIndep="netSizeNoIndep"
492             methodPreknowledgeWithIndep="netSizeExtrapolation">
493       <parentEdge timeOffset="3" variable="netSize"/>
494       <parentEdge timeOffset="2" variable="netSize"/>
495       <parentEdge timeOffset="1" variable="netSize"/>
496     </variable>
497
498     <!-- variables for service availability, no automatic learning because
499          the availability of a service can hardly be determined regularly
500          without high effort -->
501     <variable name="serviceAvailabilityByTime" scale="nominal"
502             nominalType="boolean"
503             defaultLearningDistance="900000"
504             learningMethodPreknowlNoIndep="false"
505             learningMethodPreknowlWithIndep="false"
506             methodPreknowledgeNoIndep="serviceAvailabilityFallback"
507             methodPreknowledgeWithIndep="serviceAvailabilityByTime">
508       <parentEdge timeOffset="0" variable="timeOfDayNomi"/>
509     </variable>
510     <variable name="serviceAvailabilityByPlace" scale="nominal"
511             nominalType="boolean"
512             defaultLearningDistance="900000"
513             learningMethodPreknowlNoIndep="false"
514             learningMethodPreknowlWithIndep="false"
515             methodPreknowledgeNoIndep="serviceAvailabilityFallback"
516             methodPreknowledgeWithIndep="serviceAvailabilityByPlace">
517       <parentEdge timeOffset="0" variable="placeIntegratedResults"/>
518     </variable>
519     <variable name="serviceAvailabilityByTimeAndPlace" scale="nominal"
520             nominalType="boolean"
521             defaultLearningDistance="900000"
522             learningMethodPreknowlNoIndep="false"
523             learningMethodPreknowlWithIndep="false"
524             methodPreknowledgeNoIndep="serviceAvailabilityFallback"
525             methodPreknowledgeWithIndep="serviceAvailabilityByTimeAndPlace">
526       <parentEdge timeOffset="0" variable="timeOfDayNomi"/>
527       <parentEdge timeOffset="0" variable="placeIntegratedResults"/>
528     </variable>
529     <variable name="serviceAvailableTendencyByNetSize"
530             scale="ratio"
531             defaultLearningDistance="900000"
```

```
532             learningMethodPreknowlNoIndep="false"
533             learningMethodPreknowlWithIndep="false"
534             methodPreknowledgeNoIndep="serviceAvailableTendencyFallback"
535             methodPreknowledgeWithIndep="serviceAvailableTendencyByNetSize">
536       <parentEdge timeOffset="0" variable="netSize"/>
537     </variable>
538     <variable name="serviceAvailabilityByNetSize" scale="nominal"
539             nominalType="boolean"
540             defaultLearningDistance="900000"
541             learningMethodPreknowlNoIndep="false"
542             learningMethodPreknowlWithIndep="false"
543             methodPreknowledgeNoIndep="serviceAvailabilityFallback"
544             methodPreknowledgeWithIndep="serviceAvailabilityByNetSize">
545       <parentEdge timeOffset="0" variable="serviceAvailableTendencyByNetSize"/>
546     </variable>
547     <variable name="serviceAvailabilityIntegratedResults" scale="nominal"
548             nominalType="boolean"
549             defaultLearningDistance="900000"
550             learningMethodPreknowlNoIndep="false"
551             learningMethodPreknowlWithIndep="false"
552             methodPreknowledgeNoIndep="serviceAvailabilityFallback"
553             methodPreknowledgeWithIndep="serviceAvailabilityResultIntegrate">
554       <parentEdge timeOffset="0" variable="serviceAvailabilityByTime"/>
555       <parentEdge timeOffset="0" variable="serviceAvailabilityByPlace"/>
556       <parentEdge timeOffset="0" variable="serviceAvailabilityByTimeAndPlace"/>
557       <parentEdge timeOffset="0" variable="serviceAvailabilityByNetSize"/>
558     </variable>
559
560   </variableSet>
561
562
563 </mdl:predictionModel>
```

Listing D.1: Prognosemodell zur Prognose der Dienstverfügbarkeit in der DEMAC-Middleware

Literaturverzeichnis

[ABH85] ACHILLES, Manfred ; BENDISCH, Jürgen ; HARTKOPF, Bernd: *Einführung in die Zeitreihen-Analyse mit ARIMA-Modellen*. Sankt Augustin : Gesellschaft für Mathematik und Datenverarbeitung mbH, 1985 (GMD-Studien 107). – ISBN 3–88457–107–9

[And71] ANDERSON, T. W.: *The Statistical Analysis of Time Series*. John Wiley and Sons, 1971. – ISBN 0–471–02900–9

[BF05] BATTESTINI, Agathe ; FLANAGAN, John A.: Modelling and Simulating Context Data in a Mobile Environment. In: *CAPS 2005 (Workshop on Context Awareness for Proactive Systems)*, Helsinki Institute for Information Technology, 2005, 127–136

[BGS99] BEIGL, M. ; GELLERSON, H.-W. ; SCHMIDT, A.: There is more to Context than Location: Environment Sensing Technologies for Adaptive Mobile User Interfaces. In: *Computers & Graphics Journal* 23 (1999), Nr. 6, 893–902. http://www.comp.lancs.ac.uk/~albrecht/pubs/pdf/schmidt_cug_elsevier_12-1999-context-is-more-than-location.pdf

[Bis06] BISHOP, Christopher M.: *Pattern Recognition and Machine Learning*. Springer, 2006 (Information science and statistics). – ISBN 0–387–31073–8

[BK02] BORGELT, Christian ; KRUSE, Rudolf: *Graphical Models - Methods for Data Analysis and Mining*. Chichester : John Wiley & Sons, 2002. – ISBN 0–470–84337–3

[Bra07a] BRAMER, Max: *Principles of Data Mining*. Springer, 2007 (Undergraduate Topics in Computer Science). – ISBN 1–84628–765–0

[Bra07b] BRAUBACH, Lars: *Architekturen und Methoden zur Entwicklung verteilter agentenorientierter Softwaresysteme*, Universität Hamburg - Department Informatik, Diss., 2007

[CDK05] COULOURIS, George ; DOLLIMORE, Jean ; KINDBERG, Tim: *Distributed Systems - Concepts and Design*. 4. Addison-Wesley, 2005. – ISBN 0–321–26354–5

[CK00] CHEN, Guanling ; KOTZ, David: A Survey of Context-Aware Mobile Computing Research / Dartmouth College. Version: 2000. http://www.cs.dartmouth.edu/reports/TR2000-381.pdf. Hanover, NH, USA : Dartmouth College, 2000 (TR2000-381). – Forschungsbericht

[CR99] CUNHA, Uraquitan S. ; RAMALHO, Geber: An Intelligent Hybrid Model for Chord Prediction. In: *Organised Sound* 4 (1999), 115–119. http://www.di.ufpe.br/~glr/Publications/organised-sound99.pdf. – ISSN 1355–7718

[CS95] CHAN, Philip ; STOLFO, Salvatore J.: A Comparative Evaluation of Voting and Meta-learning on Partitioned Data. In: *Proceedings of the Twelfth International Conference on Machine Learning*, Morgan Kaufmann, 1995, S. 90–98

[DA99] DEY, A. K. ; ABOWD, G. D.: Towards a Better Understanding of Context and Context-Awareness / Georgia Institute of Technology. Version: 1999. http://hdl.handle.net/1853/3389. Atlanta, GA, USA : Georgia Institute of Technology, 1999 (GIT-GVU-99-22). – Forschungsbericht. – veröffentlicht auch in *Proceedings of the CHI'00 workshop on 'situated interaction in ubiquitous computing' (CHI'00)*

[Dav03] DAVISON, A. C.: *Statistical Models*. Cambridge University Press, 2003 (Cambridge Series in Statistical and Probabilistic Mathematics). – ISBN 0–521–77339–3

[Dey00] DEY, Anind K.: Enabling the use of context in interactive applications. In: *CHI '00: CHI '00 extended abstracts on Human factors in computing systems*. New York, NY, USA : ACM, 2000. – ISBN 1–58113–248–4, S. 79–80

[Dey01] DEY, Anind K.: Understanding and Using Context. In: *Personal Ubiquitous Comput.* 5 (2001), Nr. 1, S. 4–7. http://dx.doi.org/10.1007/s007790170019. – DOI 10.1007/s007790170019. – ISSN 1617–4909

[DGL97] DEVROYE, Luc ; GYÖRFI, Lászlo ; LUGOSI, Gábor: *A Probabilistic Theory of Pattern Recognition*. 2. Springer, 1997 (Applications of Mathematics 31). – ISBN 0–387–94618–7

[Die00] DIETTERICH, Thomas G.: Ensemble Methods in Machine Learning. In: *MCS '00: Proceedings of the First International Workshop on Multiple Classifier Systems*. London, UK : Springer-Verlag, 2000. – ISBN 3–540–67704–6, 1–15

[Doe96] DOERR, Helen M.: Stella ten years later: A review of the literature. In: *International Journal of Computers for Mathematical Learning* 1 (1996), Nr. 2, 201-224. http://dx.doi.org/10.1007/BF00571080. – DOI 10.1007/BF00571080. – ISSN 1382–3892

[DSAF99] DEY, Anind K. ; SALBER, Daniel ; ABOWD, Gregory D. ; FUTAKAWA, Masayasu: The Conference Assistant: Combining Context-Awareness with Wearable Computing. In: *ISWC '99: Proceedings of the 3rd IEEE International Symposium on Wearable Computers*. Washington, DC, USA : IEEE Computer Society, 1999. – ISBN 0–7695–0428–0, S. 21–28

[Dul08] DULLER, Christine: *Einführung in die nichtparametrische Statistik mit SAS und R - Ein anwendungsorientiertes Lehr- und Arbeitsbuch*. Heidelberg : Physica-Verlag, 2008. http://dx.doi.org/10.1007/978-3-7908-2060-7. http://dx.doi.org/10.1007/978-3-7908-2060-7. – ISBN 978–3–7908–2059–1

[Dum] DUMKE, Reiner R.: *UML Tutorial - Zustandsdiagramm*. http://ivs.cs.uni-magdeburg.de/~dumke/UML/21.htm. – abgerufen am 28.03.2009

[Enc06] *Kapitel* Die bunte Welt der „Ambient Intelligence". In: ENCARNAÇÃO, José: *Umhegt oder abhängig?* Springer-Verlag, 2006. – ISBN 3–540–28143–6, S. 3–34

[ER06] EBERSPÄCHER, Jörg (Hrsg.) ; REDEN, Wolf von (Hrsg.): *Umhegt oder abhängig?* Springer-Verlag, 2006. – ISBN 3–540–28143–6

[Fer07] *Kapitel* Pervasive Computing - connected > aware > smart. In: FERSCHA, Alois: *Die Informatisierung des Alltags*. Springer-Verlag, 2007. – ISBN 978–3–540–71454–5, S. 3–10

[FN00] FENTON, N. E. ; NEIL, M.: Software metrics: Roadmap. In: *Proceedings of the Conference on The Future of Software Engineering*, ACM, 2000, 357–370

[Fri91] FRIEDMAN, Jerome H.: Multivariate Adaptive Regression Splines. In: *The Annals of Statistics* 19 (1991), Nr. 1, 1–67. http://projecteuclid.org/euclid.aos/1176347963. – ISSN 00905364

[GHJV95] GAMMA, Erich ; HELM, Richard ; JOHNSON, Ralph ; VLISSIDES, John: *Design patterns : Elements of Reusable Object-Oriented Software*. 1. Addison-Wesley, 1995. – ISBN 0–201–63361–2

[Hüb03] HÜBNER, Gerhard: *Stochastik - Eine anwendungsorientierte Einführung für Informatiker, Ingenieure und Mathematiker*. 4. Braunschweig : Friedr. Vieweg & Sohn, 2003. – ISBN 3–528–35443–7

[Hil95] *Kapitel* An Overview Of Strategies For Neurosymbolic Integration. In: HILARIO, Mélanie: *Connectionist-symbolic integration*. Lawrence Erlbaum Associates, 1995. – ISBN 0805823492, 13–35

[HMNS03] HANSMANN, Uwe ; MERK, Lothar ; NICKLOUS, Martin S. ; STOBER, Thomas: *Pervasive Computing - The Mobile World*. Springer-Verlag, 2003. – ISBN 3–540–00218–9

[IB94] IMIELINSKI, Tomasz ; BADRINATH, B. R.: Mobile wireless computing: challenges in data management. In: *Communications of the ACM* 37 (1994), Nr. 10, S. 18–28. http://dx.doi.org/10.1145/194313.194317. – DOI 10.1145/194313.194317. – ISSN 0001–0782

[Jen01] JENSEN, F. V.: *Bayesian Networks and Decision Graphs*. New York : Springer, 2001. – ISBN 0–387–95259–4

[Jän04] JÄNICH, Klaus: *Lineare Algebra*. 10. Springer, 2004. – ISBN 3–540–40207–1

[KB08] KASTENS, Uwe ; BÜNING, Hans K.: *Modellierung - Grundlagen und formale Methoden*. 2. München : Carl Hanser Verlag, 2008. – ISBN 978–3–446–41537–9

[KE06] KEMPER, A. ; EICKLER, A.: *Datenbanksysteme*. 6. München : Oldenbourg Wissenschaftsverlag, 2006. – ISBN 3–486–57690–9

[KH07] KRÄMER, Bernd J. ; HALANG, Wolfgang A.: *Contributions to Ubiquitous Computing*. Springer-Verlag, 2007. – ISBN 3–540–44909–4

[Kir98] KIRKWOOD, Craig W.: System Dynamics Methods: A Quick Introduction - Version 1 / College of Business - Arizona State University. Version: 1998. http://www.public.asu.edu/~kirkwood/sysdyn/SDIntro/sdintro.zip. 1998. – Forschungsbericht

[KP98] KOLLER, Daphne ; PFEFFER, Avi: Probabilistic frame-based systems. In: *AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*. Menlo Park, CA, USA : American Association for Artificial Intelligence, 1998. – ISBN 0–262–51098–7, S. 580–587

[Kri99] KRIEBEL, Stefan K. T.: *A Combined Parametric and Nonparametric Approach To Time Series Analysis*. Akad. Verl.-Ges., 1999 (Dissertationen zur Künstlichen Intelligenz (DISKI) 222). – ISBN 3–89601–222–3

[KS97] KANTZ, Holger ; SCHREIBER, Thomas: *Nonlinear time series analysis*. Cambridge University Press, 1997 (Cambridge Nonlinear Science Series 7). – ISBN 0–521–55144–7

[Kun05] KUNZE, C. P.: DEMAC: A Distributed Environment for Mobility-Aware Computing. In: *Adjunct Proceedings of the Third International Conference on Pervasive Computing*, Österreichische Computer Gesellschaft, 2005, 115–121

[Kun08] KUNZE, C. P.: *Kontextbasierte Kooperation: Unterstützung verteilter Prozesse im Mobile Computing*, Universität Hamburg, Fachbereich Informatik, Verteilte Systeme und Informationssysteme, Diss., 2008

[Lan00] LANE, David C.: Diagramming Conventions in System Dynamics. In: *The Journal of the Operational Research Society* 51 (2000), Nr. 2, 241–245. http://www.jstor.org/stable/254265. – ISSN 01605682

[Lar06] LAROSE, Daniel T.: *Data Mining - Methods and Models*. Wiley, 2006. – ISBN 0–471–66656–4

[LKZ+05] *Kapitel* Improved Knowledge Mining with the Multimethod Approach. In: LENIČ, Mitja ; KOKOL, Peter ; ZORMAN, Milan ; POVALEJ, Petra ; STIGLIC, Bruno ; YAMAMOTO, Ryuichi: *Foundations of Data Mining and Knowledge Discovery*. Springer, 2005 (Studies in Computational Intelligence 6). – ISBN 3–540–26257–1, S. 305–318

[LOL+05] LIN, Tsau Y. (Hrsg.) ; OHSUGA, Setsuo (Hrsg.) ; LIAU, Churn-Jung (Hrsg.) ; HU, Xiaohua (Hrsg.) ; TSUMOTO, Shusaka (Hrsg.): *Foundations of Data Mining*

and Knowledge Discovery. Springer, 2005 (Studies in Computational Intelligence 6). – ISBN 3–540–26257–1

[LP06] *Kapitel* Ubiquitous Computing - Machbarkeit und Grenzen. In: LINNHOFF-POPIEN, Claudia: *Umhegt oder abhängig?* Springer-Verlag, 2006. – ISBN 3–540–28143–6, S. 35–48

[LSR03] LAWRY, Jonathan ; SHANAHAN, Jimi ; RALESCU, Anca: *Modelling With Words - Learning, Fusion, and Reasoning within a Formal Linguistic Representation Framework*. Springer, 2003 (Lecture Notes in Artificial Intelligence 2873). – ISBN 3–540–20487–3

[Mat03] *Kapitel* Vom Verschwinden des Computers - Die Vision des Ubiquitous Computing. In: MATTERN, Friedemann: *Vom Verschwinden des Computers*. Springer-Verlag, 2003, 1–41

[Mat07] MATTERN, Friedemann (Hrsg.): *Die Informatisierung des Alltags*. Springer-Verlag, 2007. – ISBN 978–3–540–71454–5

[May] MAYRHOFER, Rene: *Context Database*. http://www.pervasive.jku.at/Research/Context_Database/index.php

[May04] MAYRHOFER, R.: *An Architecture for Context Prediction*, Johannes Kepler Universität Linz, Diss., 2004. http://www.mayrhofer.eu.org/downloads/publications/PhD-ContextPrediction-2004.pdf

[May05] MAYRHOFER, R.: Context prediction based on context histories: Expected benefits, issues and current state-of-the-art. In: PRANTE, T. (Hrsg.) ; MEYERS, B. (Hrsg.) ; FITZPATRICK, G. (Hrsg.) ; HARVEL, L. D. (Hrsg.): *PROCEEDINGS ECHISE 2005: 1st International Workshop on Exploiting Context Histories in Smart Environments*, 2005, 31–36

[MBK98a] *Kapitel* A Review of Machine Learning Methods. In: MICHALSKI, Ryszard S. ; BRATKO, Ivan ; KUBAT, Miroslav: *Machine Learning and Data Mining Methods and Applications*. Wiley, 1998. – ISBN 0–471–97199–5, S. 3–70

[MBK98b] MICHALSKI, Ryszard S. (Hrsg.) ; BRATKO, Ivan (Hrsg.) ; KUBAT, Miroslav (Hrsg.): *Machine Learning and Data Mining Methods and Applications*. Wiley, 1998. – ISBN 0–471–97199–5

[Men07] MENZEL, Wolfgang: Data Base and Information Systems - Part II / Universität Hamburg - Department Informatik - Natural Language Systems Division. Version: 2007. https://nats-www.informatik.uni-hamburg.de/view/DIS07/FolienZurVorlesung. 2007. – Vorlesungsfolien

[Müh06] *Kapitel* Mobile Usability. In: MÜHLBACH, Lothar: *Umhegt oder abhängig?* Springer-Verlag, 2006. – ISBN 3–540–28143–6, S. 171–178

[Min90] MINTON, Steven: Quantitative results concerning the utility of explanation-based learning. In: *Artif. Intell.* 42 (1990), Nr. 2-3, 363–391. http://dx.doi.org/10.1016/0004-3702(90)90059-9. – DOI 10.1016/0004–3702(90)90059–9. – ISSN 0004–3702

[mis08] *GSM World Coverage 2008*. http://www.coveragemaps.com/pubs/GSM_WorldPoster2008A.pdf. Version: 2008/A, 2008. – abgerufen am 08.03.2009

[mis09a] *Mobilfunk-Netzabdeckung beim Betreiber E-Plus*. http://eis03sn1.eplus-online.de/geo/portal/umts. Version: 3 2009. – abgerufen am 29.03.2009

[mis09b] *Mobilfunk-Netzabdeckung beim Betreiber T-Mobile*. http://www.t-mobile.de/funkversorgung/inland. Version: 3 2009. – abgerufen am 29.03.2009

[mis09c] *Mobilfunk-Netzabdeckung beim Betreiber Vodafone*. http://netmap.vodafone.de/cover4internet/index.jsp?appprofile=UMTS-Maps. Version: 3 2009. – abgerufen am 29.03.2009

[NMF05] NURMI, Petteri ; MARTIN, Miquel ; FLANAGAN, John A.: Enabling Proactiveness through Context Prediction. In: FLORÉEN, Patrik (Hrsg.) ; LINDÉN, Greger (Hrsg.) ; NIKLANDER, Tiina (Hrsg.) ; RAATIKAINEN, Kimmo (Hrsg.): *Proceedings of the Workshop on Context Awareness for Proactive Systems (CAPS 2005)* Bd. 2005-1, Helsinki University Press, 2005 (HIIT Publications). – ISBN 952–10–2519–0, 159–168

[Obj09a] OBJECT MANAGEMENT GROUP (Hrsg.): *OMG Unified Modeling Language^{TM}(OMG UML), Infrastructure*. Version 2.2. Needham, MA, USA: Object Management Group, 2 2009. http://www.omg.org/docs/formal/09-02-04.pdf

[Obj09b] OBJECT MANAGEMENT GROUP (Hrsg.): *OMG Unified Modeling Language^{TM}(OMG UML), Superstructure*. Version 2.2. Needham, MA, USA: Object Management Group, 2 2009. http://www.omg.org/docs/formal/09-02-02.pdf

[Pea88] PEARL, Judea: *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, 1988. – ISBN 1558604790

[Pet05] PETZOLD, Jan: *Zustandsprädiktoren zur Kontextvorhersage in ubiquitären Systemen*. Universitätsstr. 22, 86159 Augsburg, Universität Augsburg, Diss., 2005. http://opus.bibliothek.uni-augsburg.de/volltexte/2005/155/pdf/dissertation_petzold.pdf

[Pie06] *Kapitel* Begeisternd und mühelos zu bedienen: Mobile Endgeräte für den Menschen. In: PIETRALLA, Jens-Thomas: *Umhegt oder abhängig?* Springer-Verlag, 2006. – ISBN 3–540–28143–6, S. 115–132

[Pre99] PRECHELT, Lutz: Ausgewählte Kapitel der Softwaretechnik - Skriptum zur Vorlesung / Universität Karlsruhe. Version: 4 1999. http://page.mi.fu-berlin.de/prechelt/swt2/skript.ps.gz. 1999. – Vorlesungsskript

[Rad06] RADI, Harald: *Adding Smartness to Mobile Devices - Recognizing Context by Learning from User Habits*, Johannes Kepler Universität Linz, Diplomarbeit, 2006. http://harald.fluffnstuff.org/publications/2006/thesis.pdf

[Ric99] RICHARDSON, G. P.: Reflections for the Future of System Dynamics. In: *The Journal of the Operational Research Society* 50 (1999), Nr. 4, 440–449. http://www.jstor.org/stable/3010465. – ISSN 01605682

[Rot02] ROTH, Jörg: *Mobile Computing - Grundlagen, Technik, Konzepte*. 1. Heidelberg : dpunkt.verlag, 2002. – ISBN 3–89864–165–1

[RP06] RECHENBERG, Peter (Hrsg.) ; POMBERGER, Gustav (Hrsg.): *Informatik Handbuch*. 4. Carl Hanser Verlag, 2006. – ISBN 3–446–40185–7

[RPM00] ROMAN, Gruia-Catalin ; PICCO, Gian P. ; MURPHY, Amy L.: Software engineering for mobility: a roadmap. In: *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*. New York, NY, USA : ACM, 2000. – ISBN 1–58113–253–0, S. 241–258

[Rud08] RUDOLPH, Andreas: Data Mining Verfahren / Universität der Bundeswehr München. Version: 9 2008. http://www.bwi.unibw.de/home/rudolph/dm.html. 2008. – Vorlesungsskript

[Sat96] SATYANARAYANAN, M.: Fundamental challenges in mobile computing. In: *PODC '96: Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing*. New York, NY, USA : ACM, 1996. – ISBN 0–89791–800–2, S. 1–7

[SAT+99] SCHMIDT, Albrecht ; AIDOO, Kofi A. ; TAKALUOMA, Antti ; TUOMELA, Urpo ; LAERHOVEN, Kristof V. ; VELDE, Walter Van d.: Advanced Interaction in Context. In: *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*. London, UK : Springer-Verlag, 1999. – ISBN 3–540–66550–1, 89–101

[Sat01] SATYANARAYANAN, M.: Pervasive computing: vision and challenges. In: *IEEE Personal Communications* 8 (2001), 8, Nr. 4, 10–17. http://dx.doi.org/10.1109/98.943998. – DOI 10.1109/98.943998. – ISSN 1070–9916

[SAW94] SCHILIT, Bill ; ADAMS, Norman ; WANT, Roy: Context-Aware Computing Applications. In: *Proceedings of the Workshop on Mobile Computing Systems and Applications*, IEEE Computer Society, 1994, S. 85–90

[Sch07] *Kapitel* Eingebettete Interaktion - Symbiose von Mensch und Information. In: SCHMIDT, Albrecht: *Die Informatisierung des Alltags*. Springer-Verlag, 2007. – ISBN 978–3–540–71454–5, S. 77–101

[SGS00] SPIRTES, Peter ; GLYMOUR, Clark ; SCHEINES, Richard: *Causation, Prediction, and Search*. 2. MIT Press, 2000. – ISBN 0–262–19440–6

[SHD06] SIGG, S. ; HASELOFF, S. ; DAVID, K.: A Novel Approach to Context Prediction in UBICOMP Environments. In: *IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications 2006* (2006), S. 1–5. http://dx.doi.org/10.1109/PIMRC.2006.254051. – DOI 10.1109/PIMRC.2006.254051. ISBN 1–4244–0329–4

[SHD07] SIGG, Stephan ; HASELOFF, Sandra ; DAVID, Klaus: Prediction of Context Time Series. In: IKONEN, J. (Hrsg.) ; JUUTILAINEN, M. (Hrsg.) ; PORRAS, J. (Hrsg.): *Proceedings of the 5th Workshop on Applications of Wireless Communications (WAWC'07)*, 2007, 31–45

[Sig08] SIGG, S.: *Development of a novel context prediction algorithm and analysis of context prediction schemes*, Universität Kassel, Diss., 2008. http://www.upress.uni-kassel.de/online/frei/978-3-89958-392-2.volltext.frei.pdf

[SM05] SYMEONIDIS, Andreas L. ; MITKAS, Pericles A.: *Multiagent Systems, Artificial Societies, and Simulated Organizations*. Bd. 14: *Agent Intelligence through Data Mining*. Springer, 2005. – ISBN 0–387–24352–6

[SS07] SCHILL, Alexander ; SPRINGER, Thomas: *Verteilte Systeme*. Springer, 2007. – ISBN 3–540–20568–3

[Ste07] STELAND, Ansgar: *Basiswissen Statistik - Kompaktkurs für Anwender aus Wirtschaft, Informatik und Technik*. Springer, 2007 http://www.springerlink.com/content/978-3-540-74204-3. – ISBN 978–3–540–74204–3

[STW93] SCHILIT, Bill N. ; THEIMER, Marvin M. ; WELCH, Brent B.: Customizing Mobile Applications. In: *Proceedings of the USENIX Symposium on Mobile & Location-Independent Computing*, 1993, 129–138

[SX08] STEFANI, Antonia ; XENOS, Michalis: E-commerce system quality assessment using a model based on ISO 9126 and Belief Networks. In: *Software Quality Journal* 16 (2008), 3, Nr. 1, 107-129. http://dx.doi.org/10.1007/s11219-007-9032-5. – DOI 10.1007/s11219–007–9032–5. – ISSN 1573–1367

[SZ04] SHASHA, Dennis ; ZHU, Yunyue: *High Performance Discovery in Time Series*. Springer, 2004 (Monographs in Computer Science). – ISBN 0–387–00857–8

[TK06] THEODORIDIS, Sergios ; KOUTROUMBAS, Konstantinos: *Pattern Recognition*. 3. Orlando, FL, USA : Academic Press, Inc., 2006. – ISBN 0123695317

[Tsy04] TSYMBAL, Alexey: The problem of concept drift: definitions and related work / Trinity College Dublin - Department of Computer Science. Version: 4 2004. https://www.cs.tcd.ie/publications/tech-reports/reports.04/TCD-CS-2004-15.pdf. Irland : Trinity College Dublin - Department of Computer Science, 4 2004 (TCD-CS-2004-15). – Forschungsbericht

[Tur06] TURJALEI, Mirwais: *Integration von Context-Awareness in eine Middleware für mobile Systeme*, Universität Hamburg, Fachbereich Informatik, Verteilte Systeme und Informationssysteme, Diplomarbeit, 2006. http://vsis-www.informatik.uni-hamburg.de/getDoc.php/thesis/407/TurjaleiDiplomarbeit.pdf

[Val07] VALK, Rüdiger: Formale Grundlagen der Informatik II - Modellierung & Analyse paralleler und verteilter Systeme - WiSe 2007/08 / Universität Hamburg, Department Informatik. Version: 2007. http://www.informatik.uni-hamburg.de/TGI/lehre/vl/WS0708/FGI2/sec/fgi_2_07.pdf. 2007. – Vorlesungsskript

[VGS05] VOVK, Vladimir ; GAMMERMAN, Alexander ; SHAFER, Glenn: *Algorithmic Learning in a Ramdom World*. Springer, 2005. – ISBN 0–387–00152–2

[Wei91] WEISER, Mark: The Computer for the 21st Century. In: *Scientific American* 265 (1991), 9, Nr. 3, 66–75. http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html

[Wit02] WITTIG, F.: *Maschinelles Lernen Bayes'scher Netze für benutzeradaptive Systeme*, Universität Saarbrücken, Diss., 2002. http://deposit.d-nb.de/cgi-bin/dokserv?idn=972323384&dok_var=d1&dok_ext=pdf&filename=972323384.pdf

[Zha04] ZHANG, Harry: The Optimality of Naive Bayes. In: BARR, Valerie (Hrsg.) ; MARKOV, Zdravko (Hrsg.): *Proceedings of the 17th International FLAIRS conference (FLAIRS2004)*, AAAI Press, 2004, 562–567. – Best paper award winner (second place)

Eidesstattliche Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht.

Ich bin mit einer Einstellung in den Bestand der Bibliothek des Departments Informatik und einer Veröffentlichung im Internet einverstanden.

Hamburg, den _____ Unterschrift: _____