Dissertation

Kontextbereitstellung in offenen, ubiquitären Systemen

vorgelegt von

Michael Krause

an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig-Maximilians-Universität München

Tag der Einreichung: 29. September 2006
Tag der mündlichen Prüfung: 7. Dezember 2006

1. Berichterstatterin: Prof. Dr. Claudia Linnhoff-Popien
Ludwig-Maximilians-Universität München

2. Berichterstatter: Prof. Dr. Stefan Fischer
Universität zu Lübeck

ALWAYS BE WARY OF ANY HELPFUL ITEM
THAT WEIGHS LESS THAN ITS OPERATING MANUAL.

Terry Pratchett

Danke an alle, die mich unterstützt haben. Besonders Dir, Andreas, ohne Dich hätte ich niemals durchgehalten. Meinen früheren Kollegen vom „Gang“ in der Oettingenstraße alles Gute für die Prüfungen, die die Promotionsordnung und das Leben an der Uni noch für sie vorsehen. Speziellen Dank auch meinen Kollegen von BearingPoint, FS Technology, die mich in der Stressphase meiner Promotionszeit aushalten mussten und hoch und heilig versprochen haben, mich auch als Promovierten zu ertragen.

Zusammenfassung *– Die Vision des „Ubiquotous Computing“ verspricht schon lange eine Welt, in der jeder Dienst zu jeder Zeit an jedem Ort verfügbar ist. Darüber hinaus soll die Alltagswelt mit Rechnern durchsetzt sein, ohne dass die Benutzer diese als solche bewusst wahrnehmen. Durch Kooperation und Informationsaustausch sollen die Benutzer unaufdringlich bei ihren Aufgaben unterstützt werden, genau abgestimmt auf ihre jeweilige Situation. Dafür bedarf es kontextsensitiver Dienste.*

Kontextsensitive Dienste sind nicht neu: Das Licht im Auto wird automatisch angeschaltet, sobald es draußen dunkler wird. Hierzu sind Sensoren und Aktuatoren fest verknüpft. Um der Vision von ubiquitären Computersystemen näher zu kommen, ist es wichtig, dass Kontextinformationen auch in spontanen, dynamischen Konfigurationen bereitgestellt, gefunden, ausgetauscht und verstanden werden können. Dies ist die Ausgangssituation dieser Arbeit: Kontextbereitstellung in offenen, ubiquitären Systemen.

Dazu werden mehrere Beiträge geliefert: Eine Modellierung für Kontextinformationen, eine darauf aufbauende, dynamische Beschreibung für Kontextinformationsdienste und die Einführung von Kontextkonstruktionsbäumen, mit denen auf nicht-verfügbare Kontextinformationen geschlossen werden kann, oder mit denen diese wenigstens abgeschätzt werden können.

Abstract *– The Vision of „Ubiquitous Computing“ is promising a world where any service can be accessed at any time at any place. Computers will be everywhere without the users recognizing them as computing entities. With cooperation and exchanging information these ubiquitous computers can support the users with their tasks, optimized for the current situation. Therefore context-sensitive services are necessary.*

Context-sensitive services are not a new invention. For example: Modern cars turn their lights on when their sensors report a darkening environment. But here sensors and actuators are hard-linked. To approach the world of „Ubiquitous Computing“ it is important to provide, find, exchange and understand context information even in spontaneous and dynamic configurations. This is the starting point of this thesis: Provisioning of context in open, ubiquitous systems.

This thesis provides a new modelling of context ontologies and context information. It also develops dynamic services descriptions for context information services. Finally it introduces context construction trees which can help to deduce pieces of context information or near approximations if they are not available directly.

Inhaltsverzeichnis

1 Einführung

Wer Visionen hat, sollte zum Arzt gehen.

(Helmut Schmidt)

Zum Beispiel Musik. Wer vor der Erfindung der Schallplatte Musik hören wollte, ohne selbst zu musizieren, musste an einen Ort gehen, an dem andere Musik dargeboten haben. In der Regel war man dort Teil eines größeren Publikums und sowohl Ort als auch das Programm waren fest vorgegeben. Mit Erfindung der Tonaufnahme änderte sich das dann: Nun war die Rezeption auch zur Privatsache geworden; wer es sich leisten konnte, konnte Musik genauso gut zu Hause hören und zudem selbst bestimmen, wann er welche Aufnahme abspielte. Ein erster Schritt hin zur Personalisierung.

Weitere Schritte hat es seitdem immer wieder gegeben. Die Wiedergabegeräte wurden kleiner und tragbar, so dass auch noch der Ort in die Entscheidung des Hörers fiel. Weitere Freiheiten wurden, dass sich der Benutzer die Aufnahmen frei zusammenstellen konnte und nicht mehr an Wiedergabereihenfolge oder Umfang der verkauften Aufnahme gebunden war. Selbst Ort und Zeit des Einkaufs kann der Hörer inzwischen frei wählen, indem er sich Musik aus dem Internet herunterlädt, und selbst in den Inhalt der Aufnahme kann er mittlerweile eingreifen, wenn er zum Beispiel Sprache und Untertitel einer Musical-DVD auswählt.

Getrieben werden diese zunehmenden Möglichkeiten zur Personalisierung, die alle Lebensbereiche umfassen und sich nicht auf die Musik beschränken, vom technologischen Fortschritt und dem individuellen Wunsch nach größerer Selbstbestimmung. Letzterer ist auch immer gepaart mit dem Wunsch, Kosten und Aufwand zu sparen. Seine Grenzen findet dieser Wunsch schnell, wenn der Aufwand für die Personalisierung für den Einzelnen groß wird, weil sie Zeit und Geld kostet oder die Handhabung komplexer Strukturen verlangt. Außerdem darf die Personalisierung nicht die gewünschte soziale Zugehörigkeit zu einer Gruppe sprengen oder einem Laien Arbeit und Entscheidungen aufbürden, für die professionelle Kenntnisse notwendig sind. Experimente mit interaktiven Kriminalfilmen oder individuell zusammenstellbaren Online-Zeitungen zeigen dies deutlich.

Die Personalisierung – oder in Begriffen der Informatik: die Individualisierung der Diensterbringung – geht immer mehr darüber hinaus, lediglich persönliche Vorlieben zu berücksichtigen. Immer stärker wird die aktuelle Situation berücksichtigt, in der sich der Benutzer befindet, dazu gehören vor allem sein Aufenthaltsort und die in dieser Situation zur Verfügung stehenden technischen Ressourcen.

Möglich wurde diese Entwicklung durch den Fortschritt bei häufig so genannten „mobilen Technologien“ (obwohl nicht die Technologien mobil sind, sondern die Nutzer). Funk-

standards wie Bluetooth und W-Lan zur Vernetzung von Geräten oder zum Zugriff auf das Internet, satellitengestützte Ortungsverfahren wie GPS und die Mobiltelefonie, die gleich eine Vielzahl von Möglichkeiten (neben der Telefonie) bietet: Ortung des Benutzers, ortsunabhängiger Zugriff auf das Internet und über die Telefonrechnung ein Abrechnungsverfahren, das relativ sicher und auch für Klein- und Kleinsbeträge geeignet ist. Außerdem gibt es die Funkgestützten Identifikatoren (RFID), mit denen ein Kennzeichnen und berührungsloses Auslesen dieser Kenninformationen möglich wird.

Ein weiterer technologischer Treiber ist der Fortschritt bei der Geräteentwicklung, die immer leistungsfähigere Rechner mit immer weniger Gewicht und Platzbedarf hervorbringt und somit immer mehr tragbare Geräte ermöglicht.

Mark Weiser ist einer der Wissenschaftler, die untersucht haben, welche Möglichkeiten zur Nutzung sich aus der Geräteentwicklung ergeben. Er hat 1991 in einem berühmten Artikel eine Vision beschrieben, die die Entwicklung der Geräte, der Vernetzung und der Personalisierung auf die Spitze treibt. Er hat ihr auch den Namen gegeben: „Ubiquitous Computing“. Weiser sagt voraus, dass in jedem Raum hunderte von Rechnern integriert sein werden, was zu einer allgegenwärtigen Rechenumgebung führen wird. Allerdings werden die einzelnen Rechner nicht mehr als solche wahrgenommen, sondern hinter den Gebrauchsgegenständen verschwinden. Dazu gehört, dass nicht mehr Maus und Tastatur die Haupteingabe- und Schnittstellen sein werden, sondern Stimme, Bewegung, Fingerzeige. Spezielle Kleidung etwa wird in der Lage sein, die Bewegungen des Benutzers zu erkennen, prophezeit Weiser.

Dass solche intuitiven Methoden innerhalb von 20 Jahren zur vorherrschenden Eingabeart für Rechner würden, ist nun, nachdem 15 Jahre bereits vergangen sind, noch nicht abzusehen. Aber es gibt zum Beispiel Spielkonsolen, die bereits mit der Kameraerfassung von Gesten experimentieren. Ob Mark Weisers Vision des ubiquitären Rechnens Wirklichkeit wird, lässt sich nicht sagen. Prophezeiungen, die den technologischen Fortschritt betreffen, haben schon oft geirrt: Den Erfolg der SMS haben sie genauso wenig vorausgesehen, wie den Erfolg des Internets oder des iPods. Dafür wurden schon vor mehreren Jahrzehnten fliegende Autos und Rucksackhelikopter versprochen als Erweiterung des öffentlichen Nahverkehrs, was bekanntermaßen nicht wirklich wahr wurde.

Vorboten ubiquitärer Systeme finden sich längst, wenn man sie als solche verstehen will: Es gibt integrierte Home-Entertainment-Systeme, Navigationsgeräte, Unified-Messaging-Systeme, multifunktionale Mobiltelefone. Will man den von Weiser beschriebenen Wecker, der erkennt, dass man wach ist, bevor es an der Zeit dazu ist, der leise fragt, ob man einen Kaffee möchte, und auf ein gegähntes „Ja“ den entsprechenden Auftrag an die Kaffeemaschine weitergibt, so müssen viele Kontextinformationen ausgetauscht werden: Sensoren, die Informationen über das Hin- und Herrollen im Bett an den Wecker leiten, das Erkennen der Kaffeemaschine in der Küche, vielleicht Informationen darüber, wie viel Zeit bis zum ersten Termin bleibt, oder ob der Kaffee ausfallen muss. All diese Informationen müssen von den beteiligten Komponenten auch verstanden werden.

Eine Standardisierung dieses Informationsaustausches ist deshalb notwendig. Hier setzt diese Arbeit an und liefert unter anderem diese Beiträge:

- Eine Kontextmodellierung, mit der Kontextinformationen geeignet beschrieben werden können.
- Ein Beschreibungsmodell für Kontextinformationsdienste, die als Anbieter von Kontextinformationen auftreten.
- Eine Verknüpfung von Strategien, um Kontextinformationen zu schätzen oder abzuleiten, die nicht direkt zur Verfügung stehen.

Im Fokus sind dabei kontextsensitive Systeme, die nicht fest konfiguriert sind und sich nicht nur innerhalb der selben Domäne befinden. Die Möglichkeiten dafür zu schaffen, dass in solchen offenen, ubiquitären Systemen Kontextinformationen bereit gestellt werden können, das ist das Ziel dieser Arbeit.
. . . und Kaffee am Morgen zu bekommen, ohne vorzeitig geweckt zu werden.

Diese Arbeit gliedert sich wie folgt. Zunächst werden die grundlegenden Begriffe „Kontext“, „ubiquitäre“ und „offene“ Systeme diskutiert, definiert (Kapitel 2) und in ein generisches Szenario zur Kontextbereitstellung eingebettet (Kapitel 3). Dann werden Techniken zur Wissensrepräsentation diskutiert (Kapitel 4) und eine eigene Modellierung für Kontextinformationen entwickelt (Kapitel 5). Aufbauend auf dieser Modellierung werden Kontextanfragen erläutert und eine Beschreibung für Kontextinformationsdienste vorgestellt (Kapitel 6). Schließlich wird diskutiert, wie Kontextinformationen mit Äquivalenzen, Regeln und Bayesschen Netzen gewonnen werden können und es wird ein Algorithmus zur Verknüpfung dieser Strategien beschrieben (Kapitel 7).

2 Grundlagen

Entia non sunt multiplicanda praeter necessitatem.

(Entitäten dürfen nicht über das Notwendige hinaus vermehrt werden.)

(Johannes Clauberg)

In diesem Kapitel werden zunächst die Grundlagen der Arbeit erläutert. Das bedeutet eine Klärung der zentralen Begriffe. Das ist zunächst der Kontext im Zusammenhang mit kontextsensitiven Systemen, dann sind das ubiquitäre Systeme und offene Systeme.

2.1 Der Kontextbegriff

Das Wort „Kontext“ geht auf das lateinische Verb „contexere“ zurück, das soviel bedeutet wie „zusammenflechten“ oder „eng verknüpfen“. Im Partizip Perfekt „contextus“ bezeichnet es entsprechend etwas, das eng verflochten und verknüpft ist. Es ist in vielen Wissenschaften und im Alltag gebräuchlich, um all die Objekte, Eigenschaften, Aussagen oder Ereignisse zu bezeichnen, die in engem Zusammenhang stehen mit dem jeweiligen Betrachtungsgegenstand und ohne deren Kenntnis man nur ein eingeschränktes oder falsches Wissen über eben jenen Betrachtungsgegenstand erlangen kann.

In diesem Kapitel wird die Verwendung und die Bedeutung des Begriffs „Kontext“ in der Informatik erläutert, wenn damit Szenarien gemeint sind[1], für die sich im Englischen der Begriff *context-aware computing* gebildet hat. Im Deutschen spricht man von kontextsensitiven Anwendungen und Diensten.

2.1.1 Diskussion des Kontextbegriffs in der Literatur

Was ist Kontext? Eine der ersten, die den Begriff *context-aware computing* geprägt haben, waren Bill Schilit und Marvin Theimer in [ST94]. Sie schreiben, dass bestimmte Informationen es Software ermöglichen, sich entsprechen anzupassen. Diese Informationen betreffen den aktuellen Ort („*location of use*“) und beschreiben, welche Personen und Objekte in der Nähe sind und wie sich diese Objekte mit der Zeit verändern: „*We use the term context-aware computing to describe software exhibiting these general capabilities*“.

[1] Dieser Bereich ist natürlich nicht der einzige in der Informatik, der einen Fachbegriff „Kontext“ kennt, man denke nur an kontextsensitive Grammatiken oder den Prozesskontext.

Schilit selbst hat später drei Kategorien von Kontextinformationen unterschieden, die er „*environmental dynamics*“ nannte:

- „die Unterschiede, ob man alleine ist oder in der Gesellschaft anderer,“
- „die sich verändernde soziale Situation,“
- „und die sich verändernden Bedingungen der unbelebten Umgebung.“

Er führt aus, dass die Bandbreite der Informationen über die Welt, die als Kontext relevant sein können, sehr groß ist, und liefert detaillierte Beispiele:

Kategorie	*Beispiele*
Physische Objekte	*Menschen, Computer, Bildschirme*
Zustand	*verfügbar, in Gebrauch*
Räumliche Beziehungen	*bei, in der Nähe von*
Umgebungsbedingungen	*schwach beleuchtet, laute Umgebung*
Ways & Means	*Instruktionen, Netzwerkadresse*
Anpassung	*keine Anrufe durchstellen*

Tabelle 2.1: Informationskategorien nach Schilit.

Andere Definitionen von Kontext waren zwar in der Sache enger oder weiter gefasst, blieben aber qualitativ gleichwertig: Sie versuchten, Kontext durch Beispiellisten und feste Klassifikationen zu fassen. So nahmen Peter Brown, John Bovey und Xian Chen [BBC97] auch Tageszeit, Jahreszeit, Temperatur und Ähnliches mit auf in die Kontextdefinition. Und Albrecht Schmidt, Michael Beigl und Hans-W. Gellersen [SBG99] empfahlen, Kontext in sechs Kategorien einzuteilen: „*User*“, „*Social Environment*“, „*Task*“, „*Conditions*“, „*Infrastructure*“, „*Location*“.

All diesen Ansätzen gegenüber führen Anind Dey, Gregory Abowd und Daniel Salber zu Recht aus, dass auf Beispielen fußende Definitionen zu speziell und zu wenig formal sind. Mit ihnen lässt sich zum Beispiel nicht entscheiden, ob potenziell neue Typen von Kontextinformationen von der Definition umfasst werden oder nicht. Die Autoren fordern eine „*operational definition*“ und liefern sie gleich mit.

> Kontext: jede Information, die benutzt werden kann, den Zustand einer Entität zu beschreiben (zum Beispiel einer Person, eines Ortes oder eines Objekts), der als relevant betrachtet wird für die Interaktion zwischen einem Benutzer und einer Anwendung, wobei auch der Benutzer oder die Anwendung selbst diese Entität sein können. Kontext ist typischerweise ein Ort, die Identität oder der Zustand einer einzelnen Person, einer Gruppe oder von Rechnern und anderen Objekten. [DSA01]

Allerdings ist auch diese Definition mangelhaft. Ganz abgesehen davon, dass sie doch auch wieder mit Beispielen arbeitet, sagt sie nichts darüber aus, welche Bedingungen erfüllt sein müssen, damit eine Information die Situation von Entitäten beschreiben kann. Offen bleibt ebenfalls die Frage, wann etwas relevant ist für die Interaktion zwischen Nutzer und Applikation, genauso wie die Frage, welche Arten der Interaktion davon betroffen sind. Nichtsdestotrotz ist sie bis heute die am weitesten verbreitete, und meistzitierte Kontextdefinition.

Dabei gibt es andere Definitionen, die versuchen, die Mängel der Dey-Definition auszugleichen. Etwa die von Terry Winograd, der in etlichen Artikeln auf den von Dey et al. (z.B. [Win01]) geantwortet hat. Er betont, dass Kontext ein „*operational term*“ ist: „*Features of the World become context through their use.*“. Heinz-Gerd Hegering, Axel Küpper, Claudia Linnhoff-Popien und Helmut Reiser liefern eine Definition (wenn auch nur indirekt), die die Entscheidung darüber, ob Informationen Kontextinformationen sind zumindest erleichtert.

> „*A service then becomes a context-aware service if its behaviour or the content it processed is adapted to the context of one or several entities in a transparent way*“ [HKLPR03].

Im Umkehrschluss bedeutet das nämlich, dass Informationen über Entitäten dann zu Kontext werden, wenn sie benutzt werden, um das Verhalten eines Dienstes darauf anzupassen oder den Inhalt, den dieser Dienst verarbeitet. Ab welchem Grad der Adaption des Prozessverhaltens wird die verantwortliche Information zur Kontextinformation?

2.1.2 Kontextdefinition dieser Arbeit

Ist jede Information, die ein Prozess vom Benutzer oder von anderen Prozessen erhält, eine Kontextinformation? Mit etwas großzügiger Interpretation ließe sich jede der erwähnten Kontextdefinitionen als Argument dafür benutzen, dies mit Ja zu beantworten. Tatsächlich aber würde niemand etwa den Inhalt eines Mailordners, den ein Imap-Server einliest, um sie einem Mail-Client weiterzuleiten, intuitiv als Kontextinformationen verstehen. Die folgende Definition für den Kontextbegriff dieser Arbeit ist deshalb um eine deutlichere Abgrenzung bemüht.

Definition (Kontextinformation)
Eine Information heißt dann eine Kontextinformation für die Ausführung eines Dienstes,

1. *wenn der Dienst die Information selbst anfordert,*
   a) *um die Interaktion mit dem (menschlichen oder elektronischen) Nutzer zu reduzieren oder*
   b) *um eine Information zu erhalten, über die der Nutzer selbst nicht verfügt,*

2. *wenn der Dienst die Information als Beschreibung eines Teils des Zustands einer Entität interpretiert,*

3. *und wenn die Information einen stärkeren Einfluss auf den Ablauf des Dienstes hat, als es ihre bloße Ausgabe verlangen würde.*

Die drei Bedingungen dieser Definition sollen erläutert werden:

Erläuterung zu 1 Diese Definition übernimmt die Auffassung von Terry Winograd, dass eine Kontextinformation sich von anderen Informationen nicht durch das unterscheidet, was sie aussagt, sondern vor allem durch ihre Verwendung. Eine Information wird erst im Zusammenspiel mit einem kontextsensitiven Dienst zur Kontextinformation. Umgekehrt ergibt sich daraus auch dies:

Definition (Kontextsensitiver Dienst)
Ein Dienst heißt kontextsensitiver Dienst, wenn er eine Kontextinformation oder mehrere Kontextinformationen anfragen kann.

Die obige Kontextinformations-Definition umfasst auch den Zweck von kontextsensitiven Diensten: Systemen die Fähigkeit zu größerer Autonomie zu geben, da sie sich notwendige Informationen selbst besorgen können, die ihre Ausführung beeinflussen. Das kann ein Roboter sein, der den ihn umgebenden Raum selbst optisch und taktil erfasst, um sich eigenständig durch ihn hindurch zu bewegen ohne ferngesteuert werden zu müssen. Das kann aber auch nur eine Eingabemaske sein, die bereits Werte enthält, die im aktuellen Kontext sinnvoll erscheinen, so dass der Benutzer, falls er die Voreinstellungen übernimmt, weniger Arbeit mit der Eingabe hat. Besonders wertvoll werden kontextsensitive Dienste, wenn sie Kontextinformationen von Sensoren automatisch erfragen, die der Benutzer gar nicht oder nicht in der notwendigen Genauigkeit kennt – etwa seine aktuelle Position auf der Autobahn, die für ein Navigationssystem notwendig ist.

Der Entwickler eines kontextsensitiven Dienstes programmiert ein Dienstverhalten, das an (mindestens) einer Stelle abhängig von einer Kontextinformationen ist. Er programmiert die Anforderung der betreffenden Information entweder statisch oder legt Routinen fest, die die Bestandteile der Anforderung dynamisch zur Laufzeit bestimmen.

Adaptive Systeme zeichnen sich per se durch eine hohe Dynamik aus. Gemäß der eben aufgezeigten Definition eines kontextsensitiven Dienstes genügt es, dass ein Dienst eine Kontextinformation anfordern kann; es kann durchaus vorkommen, dass eine Dienstausführung je nach Situation keine Kontextinformationen anfordern muss.

Das Verb „anfordern“, ist dabei übrigens nicht eingeschränkt im Sinne eines Pull-Verfahrens zu verstehen. Auch eine Anmeldung bei einem anderen Dienst, der dafür sorgt, dass Ereignis-abhängig Kontextinformationen an den Dienst geschickt werden (Push-Verfahren), ist in diesem Sinne eine „Anforderung“.

Erläuterung zu 2 Die zweite Bedingung der Kontextinformations-Definition, beinhaltet zweierlei. Zum einen, dass der Dienst die Semantik der Kontextinformation erfassen kann, zum anderen ergeben sich daraus Anforderungen an die Modellierung einer Kontextinformation, die dergestalt sein muss, dass eben jene Semantik damit ausgedrückt werden kann.

Erläuterung zu 3 Schließlich wird in der Definition noch unterschieden zwischen Inhaltsinformationen (*content*) und Kontextinformationen. Inhaltsinformationen haben keinen Einfluss auf den Ablauf des Dienstes, Kontextinformationen dagegen schon.

Schließlich bleibt noch der Zusammenhang zwischen Kontextinformation und Kontext zu definieren. Dabei wird unterschieden:

Definition (Kontext einer Dienstausführung)
Die Menge aller Kontextinformationen, die ein Dienst für seine Ausführung anfordert, heißt Kontext dieser Dienstausführung.

Definition (Kontext einer Entität)
Die Menge aller Kontextinformationen zu einer bestimmten Entität heißt Kontext dieser Entität.

Während sich der Kontext einer Dienstausführung bestimmen lässt, ist der gesamte Kontext einer Entität eine eher theoretische Menge, von der auch immer nur eine Teilmenge verfügbar ist.

2.1.3 Qualitätsmerkmale von Kontext

Nach der Definition des Begriffs der eigentlichen Kontextinformation werden in diesem Abschnitt nun Metainformationen diskutiert. Dazu gehören vor allem die Qualitätsmerkmale von Kontextinformationen, denen die Kontextforschung immer größere Bedeutung beimisst (siehe auch [Alb05, KH05]). Bereits 1999 haben Albrecht Schmidt und andere [SAT+99] eine Kontextinformation mit einem allgemeinen Indikator dafür verknüpft, wie wahrscheinlich es ist, dass die Kontextinformation die Wirklichkeit widerspiegelt. Auch Anind Dey und Gregory Abowd [DA99] sagen voraus, dass für künftige, realistische kontextsensitive Dienste die Herausforderung darin bestehen wird, mit nicht perfekten Kontextinformationen umgehen zu können.

Betrachtet man die wichtigsten Arbeiten, die sich direkt oder mittelbar mit Metainformationen zu Kontextinformationen befasst haben [Dey00, Dey01, GS01, CCKM01, CDS04, EHL01, GWPZ04, SLPF03a, SLPF03b, HNSG05, KH05], so erkennt man, dass das Verständnis, was zu diesen Metainformationen zählt, zwar ähnlich ist, aber nie deckungsgleich. Es folgt deshalb eine Liste (ohne Anspruch auf Vollständigkeit), die die jeweiligen Metainformations-Kategorien aus den genannten Ansätzen vereinheitlichend zusammenfasst und bewertet:

Wahrscheinlichkeit der Gültigkeit Zeigt an, mit welcher Wahrscheinlichkeit eine vorliegende Information den angegebenen Zustand der realen Welt tatsächlich beschreibt – natürlich zu dem angegebenen Zeitpunkt und innerhalb der angegebenen Präzisionsgrenzen. Diese Metainformation hängt ab vom (logischen oder physischen) Messverfahren und allgemeinen oder temporären Messbedingungen, berücksichtigt aber nicht die Möglichkeit der absichtlichen Manipulation von Daten. Der Wert dieser Metainformation wird vom Sensor beziehungsweise dem dazu gehörenden Kontextinformationsdienst selbst angegeben und der Kontextinformation hinzugefügt.

Vertrauenswürdigkeit Wird oft mit der „Wahrscheinlichkeit der Gültigkeit“ vermischt. Bezieht sich aber darauf, ob die Kontextinformation tatsächlich durch das Messen des angegebenen Zustands der realen Welt erstellt worden ist. Die Daten könnten von Dritten manipuliert worden sein, wenn der Kontextinformationsdienst und der von ihm gekapselte Sensor nicht sicher sind. Oder die Kontextinformation könnte betrügerisch aktueller oder präziser ausgezeichnet sein, als sie tatsächlich ist. Diese Gefahr ist in offenen Systemen natürlich besonders groß. Der Grad der Vertrauenswürdigkeit kann ein Empfänger einer Kontextinformation selbstverständlich nicht von der Quelle selbst in Erfahrung bringen. Aber sie kann etwa von vertrauenswürdigen Dritten stammen, oder aus der bisherigen eigenen Erfahrung mit den jeweiligen Quellen geschlossen werden. Wichtig dafür ist allerdings die folgende Metainformations-Kategorie:

Historie Diese sagt etwas über den Ursprung einer Kontextinformation aus. Entweder enthält sie die Angabe des Kontextinformationsdienstes, dessen Sensor sie entstammt, oder sie enthält die Kontextinformationen, aus denen sie abgeleitet worden ist. Diese Metainformation ist einerseits wichtig, um auf die Vertrauenswürdigkeit einer Kontextinformation zu schließen. Andererseits kann sie dazu dienen, zu verifizieren, dass unterschiedliche Kontextinformationen über den gleichen Sachverhalt tatsächlich verschiedenen Quellen entstammen und sie sich somit gegenseitig bestätigen. Diese Metainformation wird vom Sensor angegeben oder dem Inferenzsystem, das neue Kontextinformationen aus bestehenden ableitet, und der Kontextinformation hinzugefügt.

Zeitpunkt der Gültigkeit Wird vom Kontextinformationsdienst angegeben und beschreibt, wie der Name schon sagt, zu welchem Zeitpunkt die Kontextinformation gilt. Diese Metainformation wird der Kontextinformation beigefügt.

Präzision Beschreibt die Abweichung (minimale, maximale, durchschnittliche…) des gemessenen Wertes vom tatsächlichen Wert in der realen Welt. Die Präzision einer Messung kann per definitionem nicht aus der Messung allein geschlossen werden, sondern nur aus mehreren Messungen. Der aktuelle Wert wird vom Kontextinformationsdienst angegeben und der Kontextinformation hinzugefügt.

Wiederholbarkeit Dieser übliche Name dieser Metainformationskategorie (vom englischen *„repeatability“*) ist unglücklich gewählt, da es weniger darum geht, ob die Messung unter den gleichen Umständen wieder die gleichen Ergebnisse liefern würde. Im Kern will man mit diesem Parameter aussagen, wie veränderlich eine bestimmte Kontextinformation ist. Als Beispiel: Wird als Kontextinformation der Wohnort einer Person gesucht, so ist die Information in der Regel über mehrere Jahre statisch. Wird aber der Aufenthaltsort einer Person gesucht, die gerade mit 200 km/h auf der Autobahn fährt, so ist diese Information hochveränderlich. Der Wert dieser Metainformations-Kategorie kann nicht aus einer einzelnen Messung geschlossen werden, sondern muss entweder aus den Werten vergangener Messungen extrapoliert oder aus anderen Kontextinformationen (z.B: „Person fährt in Auto auf Autobahn“) geschlossen werden. Quelle dieses Wertes, der der Kontextinformation beigefügt wird, ist also entweder der Kontextinformationsdienst oder ein Inferenzsystem.

Auflösung Gibt die Auflösung der Datenwerte an, die der zu Grunde liegende Kontextinformationsdienst liefern kann. Ein Beispiel: Eine Entfernungsangabe „1500 Meter“ besitzt eine andere Aussagekraft, wenn die Auflösung „500 Meter“ statt „1 Meter“ beträgt. Diese Angabe ist verwandt mit der Präzision, unterscheidet sich aber dennoch von ihr. Der Wert der Auflösung wird vom Kontextinformationsdienst angegeben und der erstellten Kontextinformation beigefügt.

Abdeckung Diese Angabe grenzt ein, welche Werte die Kontextinformationen eines Kontextinformationsdienstes überhaupt annehmen können. Ein Lokalisierungsdienst zum Beispiel, der Entitäten nur in Deutschland lokalisieren kann, grenzt die möglichen Ortsinformationen auf diejenigen ein, die in Deutschland liegen. Die Abdeckung wird vom Kontextinformationsdienst angegeben. Sie dient aber nur der Auswahl eines geeigneten Kontextinformationsdienstes und hat keinen Einfluss auf die Aussagekraft einer Kontextinformation, weswegen sie der Kontextinformation nicht beigefügt wird.

Frequenz Diese sagt aus, in welchen Intervallen der Sensor Messungen vornimmt. Der Wert stammt vom Kontextinformationsdienst, wird aber nicht der Kontextinformation beigefügt, da er keinen Einfluss auf ihre Aussagekraft hat, sondern lediglich der Auswahl der Kontextinformationsdienste und der Verwaltung von Kontextinformationen dient.

Zugriffsrechte Diese Kategorie von Metainformationen dient allein der Verwaltung von Kontextinformationen und hat keinen Einfluss auf die Aussagekraft einer Kontextinformation. Wer die Zugriffsrechte auf eine bestimmte Kontextinformation festlegt, wo diese gespeichert werden und wie sie durchgesetzt werden, ist noch offen, solange es keine generelles Konzept zur Einhaltung der Privatsphäre gibt. Fragen der Sicherheit und der Privatsphäre klammert diese Arbeit noch dezidiert aus.

Preis Auch der Preis einer Kontextinformation ist eine Metainformation, die nichts mit der Aussagekraft der Information zu tun hat. Er dient allein der Auswahl und der Verwaltung von Kontextinformationen.

Aktualität Immer wieder wird die Metainformations-Kategorie „*freshness*“ aufgeführt, die sich am besten als „Aktualität“ übersetzen lässt. Sie besagt, ob eine Kontextinformation noch aktuell genug ist, um verwendet werden zu können. Damit ergibt sich dieser Wert direkt aus dem Alter („Zeitpunkt der Gültigkeit“), der Veränderlichkeit („Wiederholbarkeit“) und den Anforderungen des jeweiligen kontextsensitiven Dienstes. Diese Metainformations-Kategorie ist somit anwendungsabhängig.

Der Begriff „*Quality of Context*“ ist 2003 von Thomas Buchholz, Axel Küpper und Michael Schiffers [BKS03] wie folgt definiert worden:

> Zu *Quality of Context (QoC)* zählt jede Information, die die Qualität einer Information beschreibt, die als Kontextinformation benutzt wird. Also bezieht sich QoC auf die Information und nicht auf den Prozess oder die Geräte, die diese Information eventuell bereit stellen.

Diese strikte Ausnahme vom Prozess der Kontextinformations-Bereitstellung ist ganz offensichtlich davon motiviert, sich vom Begriff des *Quality of Service* (QoS) abzugrenzen. Was im Prinzip richtig ist, sich in dieser Absolutheit jedoch nicht halten lässt. Wie schon ausgeführt, ist eben auch die Quelle einer Kontextinformation durchaus ausschlaggebend dafür, welche Vertrauenswürdigkeit man der Information beimessen möchte.

In [KH05] sind Metainformationen deshalb unterteilt in solche, die Aussagekraft auf einer Kontextinformation haben und andere, die dies nicht haben, wie Abdeckung, Frequenz, Zugriffsrechte und Preis. Die Metainformationen, die diesen Einfluss haben, können zudem unterteilt werden, in Metainformationen, die der Information inhärent sind, und solche, die anwendungsabhängig sind, wie die Vertrauenswürdigkeit oder die Aktualität. Diese Arbeit übernimmt diese Unterteilung.

Nach der Definition des Kontextbegriffs innerhalb dieser Arbeit unter besonderer Berücksichtigung der Qualitätseigenschaften von Kontextinformationen, wird im Folgenden das Einsatzgebiet dieser Kontextinformationen näher betrachtet: Ubiquitäre Systeme.

2.2 Ubiquitäre Systeme

Der Begriff des *Ubiquitous Computing* ist von Mark Weiser 1988 geprägt worden, um eine Zukunft zu beschreiben, in der der PC ersetzt wird durch Computer, die in alltägliche Gebrauchsgegenstände integriert sind [Xer99]. Als Initialpunkt der Forschung zu ubiquitären Systemen gilt der Artikel von Weiser, der 1991 in „The Scientific American“ unter dem Titel „The Computer of the Twenty-First Century“ erschienen ist [Wei91]. Darin beschreibt er die Vision einer Welt, in der Rechner nicht mehr eigenständige Geräte sind, mit denen

sich die Menschen explizit auseinandersetzen müssen. In eigenen Worten: Während man sich heutzutage meist noch an den Schreibtisch setzen muss, um auf den Monitor des PC zu blicken, wenn man Ausgaben wie E-Mails oder WWW-Seiten lesen möchte, könnte dies in Zukunft stattdessen auf dem Fernseher, auf dem Display des Auto-Bordcomputers oder durch Projektion auf beliebige helle Flächen in der Umgebung möglich sein.

Neben *Ubiquitous Computing*, (etwa: der „allgegenwärtige" Computer), haben sich auch die Begriffe *Pervasive Computing* (etwa: der alles durchdringende Computer), *Ambient Intelligence* und (seltener) der *Disappearing Computer* im Sprachgebrauch eingebürgert. Die Nuancen in der Bedeutung, die diese Begriffe bei ihrer Einführung unterscheiden sollten, sind durch einen weitgehend synonymen Gebrauch gänzlich verwischt worden.

Weisers Annahme, dass seine Vision bereits zwanzig Jahre später Wirklichkeit sein könnte, war jedoch viel zu optimistisch. Zu vielfältig sind die Voraussetzungen, die zuvor noch geschaffen werden müssen. Im Folgenden soll dazu ein Überblick gegeben werden, der außerdem hilft, diese Arbeit selbst einzuordnen.

Hardware Wenn Computer in Alltagsgegenständen verschwinden, aber in jeder Umgebung verfügbar sein sollen, dann ist eine sehr große Anzahl von ihnen notwendig. Das wiederum verlangt eine geringe Größe und einen geringen Preis. Zu lösen ist auch das Energieproblem, denn nicht jedes Gerät kann über ein Stromnetzkabel versorgt werden und die Leistungsfähigkeit von Akkus ist noch sehr beschränkt. Eine große Zahl an Computern bedeutet außerdem einen enormen Wartungsaufwand, den es ebenfalls gilt, in Grenzen zu halten. Schließlich müssen einige der Geräte trotz ihrer geringen Größe erheblichen Rechenaufwand leisten können, da sie zum Beispiel Inferenzaufgaben lösen sollen. Alles zusammengenommen beschreibt Hardware auf einer Entwicklungsstufe, die zwar angestrebt ist, aber eben bis heute noch nicht erreicht worden ist. Wie sehr sich die Entwicklung aber in die Richtung bewegt, die Weiser angezeigt hat, zeigt, dass immer mehr Geräte mit Rechnern ausgestattet werden, die vormals keinen besaßen. Oft werden sie dann zudem mit dem Etikett „intelligent" versehen: Fernseher, Telefone, Beamer, Anzeigen im Straßenverkehr und so weiter.

Kommunikation Allein das Vorhandensein von vielen Rechnern in der jeweiligen Umgebung ist nicht genug. Die Vision des *Ubiquitous Computing* verlangt ein hohes Maß an Kooperation und Informationsaustausch, um die beschriebene Unterstützung des Menschen gewährleisten zu können. Die Geräte müssen also in der Lage sein, sich über Kommunikationsnetze zu verständigen. Dazu ist ein entsprechender Ausbau der Kommunikationsnetze notwendig, ebenso wie der Netzzugang von mobilen Geräten. Mit dem Mix aus einem Internetzugang über stationäre und mobile Zugriffspunkte (letztere via Mobilfunknetze oder Satelliten) und diversen Funktechnologien mit kürzeren Reichweiten (Bluetooth, W-Lan, Infrarot, RFID) wird dies grundsätzlich heute bereits erreicht. Bedarf besteht allerdings noch an der Möglichkeit des nahtlosen Übergangs zwischen diesen Übertragungstechnologien, sowie einer ausreichen-

den Bandbreite zu einem Preis, der so günstig ist, dass die meist ständige und teils erhebliche Kommunikation in Kauf genommen werden kann.

Standards Um die für ubiquitäre Systeme notwendige Interoperabilität zu erreichen, bedarf es eines gemeinsamen Verständnisses über Informationen, Dienste und Funktionalität, wozu wiederum entsprechende Standards notwendig sind. Standards zur Beschreibung von Informationen, Diensten und Funktionalität, Standards zum Propagieren, zur Installation, zur Suche und zur Inanspruchnahme von ubiquitären Diensten, aber auch Standards für das Management von ubiquitären Diensten, beispielsweise für das Accounting. Hier ist zum einen noch nicht klar, welche der bereits existierenden Standards sich als die geeignetsten für ubiquitäre Systeme durchsetzen werden. Wobei zu beachten ist, dass wirtschaftliche Interessen und Konkurrenzdruck einer Standardbildung zur Ermöglichung von Interoperabilität oft entgegenstehen. Beispiele dafür sind der so genannte Browserkrieg, der Wettbewerb zwischen Beta2000 und VHS oder die Konkurrenz um ein neues DVD-Format.

Mensch-Maschine-Schnittstelle Einer der Hauptvorteile der Ubiquitous-Computing-Vision ist die Unterstützung des Benutzers durch eine kooperative und automatische Reaktion der allgegenwärtigen Rechner auf den jeweiligen Kontext. Umso wichtiger wird dies, da Geräte, Dienste und ihr Zusammenspiel stetig komplexer werden. Entscheidend dafür ist, dass die Schnittstelle des Benutzers zu den ihn umgebenden Geräten trotz der jeweils wechselnden Konfiguration immer intuitiv bleibt und dass deren Aktionen für ihn stets durchschaubar bleiben, auch wenn er den jeweils gemessenen Kontext im Einzelnen nicht so exakt kennt, wie die Geräte dies tun. Andernfalls hätte es den gegenteiligen Effekt: Für den Benutzer wird die rechnerdurchsetzte Umwelt unkontrollierbar und damit wertlos. Hier ist noch einiges an Forschungsarbeit zu leisten.

Privatsphäre Der notwendige Austausch von Informationen zwischen Geräten in wechselnden Konfigurationen steht dem ebenso notwendigen Schutz der Privatsphäre gegenüber. Notwendig ist eine Möglichkeit, zu formulieren, wer welche Informationen erhalten darf. Ebenso muss die Durchsetzung dieser Regeln gesichert werden. Nach Ansicht des Autors ist es eine *conditio sine qua non*, dass ubiquitäre Systeme in der Lage sein müssen, die Privatsphäre und Datenschutzinteressen der Nutzer zu schützen. Hierfür gibt es bislang noch keine wirksamen und effizienten Gesamtstrategien.

Sicherheit Die freie und wechselnde Kooperation zwischen verschiedenen Geräten birgt die Gefahr, dass sich Angreifer sehr leicht in diese Kooperation einschleusen können und entweder die Privatheit von Informationen gefährden oder das System stören. Sicherheit in ubiquitären Systemen ist ebenso grundlegend notwendig wie schwierig durchzusetzen.

Weitere soziale Voraussetzungen Ubiquitäre Systeme setzen ganz neue Maßstäbe in der automatisierten Zusammenarbeit, für die es auch nicht-technischer, gesellschaftlicher Voraussetzungen bedarf. Etwa juristische (wie zum Beispiel Haftungsfragen), betriebswirtschaftliche (neuartige Geschäftsmodelle) oder soziale (Fragen der Akzeptanz). Anhand der Fülle technischer Voraussetzungen sei es erlaubt, hier erleichtert anzumerken, dass diese weiteren Fragen nicht von der Informatik geklärt werden können.

Ubiquitäre Systeme im Sinne dieser Arbeit sind auch offene Systeme, wie im folgenden Abschnitt erläutert wird.

2.3 Offene Systeme

In der Physik kennt man offene Systeme als Gegensatz zu isolierten Systemen. In offenen Systemen ist ein Stoff- und Energieaustausch mit der Umwelt möglich, in isolierten nicht. Wobei echte isolierte Systeme eher von theoretischer Bedeutung sind, zum Beispiel bei der Entwicklung von Modellen und Theorien.

In der Informatik dagegen kommt es dagegen nicht auf den Austausch von Stoff oder Wärme an, sondern auf den Austausch von Informationen. Der OSI-Standard der *International Standardisation Organisation* definiert zum Beispiel den akzeptierten internationalen Standard, mit dem Rechner in „offenen Systemen“ untereinander kommunizieren können. OSI wiederum steht für *Open Systems Interconnection*.

Andrew Tanenbaum fasst den Begriff weiter und definiert offene Systeme grundsätzlich als „Systeme, die für die Kommunikation mit anderen Systemen offen sind“ [Tan00]. Und ausgerechnet die Online-Enzyklopädie Wikipedia, deren Inhalte man in wissenschaftlichem Zusammenhang nur mit spitzen Fingern anfassen sollte, weil sie eben auch „offen“ ist, ausgerechnet diese liefert eine sehr treffende Beschreibung: „In der Netzwerktechnik ist ein Offenes System eine Betriebsumgebung, die Interoperabilität und Portabilität durch Offene Schnittstellen und Spezifikationen sichert“ (de.wikipedia.org/wiki/Offenes_System). Wie im vorangegangenen Kapitel bereits erläutert, gilt dies auch für die zu schaffenden Ubiquitären Systeme: Geräte und Dienste auf diesen Geräten müssen kommunizieren können, unabhängig von ihrer Plattform (Portabilität) und unabhängig davon, ob sie in der gleichen Domäne oder sich völlig fremd sind (Interoperabilität). Notwendig dazu sind gegenseitig bekannte Schnittstellen und Protokolle für die Kommunikation, die ausgetauschten Informationen und die Dienstzugriffe.

Vor dem Hintergrund der Idee des Ubiquitous Computing wie es Mark Weise beschrieben hat, können ubiquitäre Systeme nur offene Systeme sein. Dennoch gibt es Ansätze für geschlossene Systeme mit ubiquitären Merkmalen, zum Beispiel abgeschlossen innerhalb eines Haushalts oder einer Firma. Es ist die Überzeugung des Autors, dass solche Systeme ein notwendiger Zwischenschritt sind, um einige der im vorigen Abschnitt beschriebenen Herausforderungen erst einmal im Kleinen zu bewältigen. Ihre eigentlichen Vorteile durch Netzeffekte können ubiquitäre Systeme jedoch nur in offenen Systemen entfalten,

wofür unter anderem Methoden wie diese notwendig sind, die in dieser Arbeit beschrieben werden. Um dies herauszustellen, weist auch der Titel darauf hin, dass sie für „offene, ubiquitäre Systeme“ gedacht sind.

2.4 Zusammenfassung

In diesem Kapitel ist der Kontextbegriff für diese Arbeit definiert worden, und es sind die Eigenschaften von ubiquitären und offenen Systemen erläutert worden. Dabei besteht folgender Zusammenhang: Ubiquitäre Systeme innerhalb dieser Arbeit sind auch offene Systeme und benutzen Kontextinformationen, um Abläufe dynamisch der jeweiligen Situation anzupassen.

3 Verteilung von Kontextinformationen in offenen Systemen

Nec scire fas est omnia.

(Es ist unmöglich, alles zu wissen.)

(Horaz)

Nachdem im vorangegangenen Kapitel die drei für diese Arbeit grundlegenden Systemcharakteristika erläutert worden sind („kontextsensitiv“, „ubiquitär“ und „offen“), werden in diesem Kapitel verschiedene Beispielszenarien gezeigt, ein generisches Szenario erläutert und bestehende System-Ansätze zur Verteilung von Kontextinformationen betrachtet.

3.1 Beispielszenarien kontextsensitiver Dienste

Welche Art von Diensten ermöglicht ein System, dessen Dienste auf Kontextinformationen dynamisch reagiert, welche über Domänengrenzen spontan angefordert und ausgetauscht werden können? Es folgen einige Beispiele.

E-Mail im Hotel: Bill betritt zum ersten Mal sein Hotelzimmer. Der E-Mail-Client auf seinem PDA erkennt, dass ein W-Lan verfügbar ist, das kostenfrei genutzt werden darf. Das Gerät prüft deshalb sofort, ob neue E-Mails vorliegen. Eine der neuen Nachrichten – eine Videonachricht – ist von Bills Vorgesetztem und als sehr dringend eingestuft. Da mit dem Wand-Display des Zimmers ein besseres Videoausgabegerät als Bills PDA verfügbar ist, wird die Nachricht dort abgespielt. Mit den Kontrolltasten auf seinem PDA kann Bill die Wiedergabe fern bedienen – also kurz anhalten, die Lautstärke verändern oder sie wiederholen lassen.

Alice will fernsehen: Der Kommunikationsserver in der Wohnung von Alice und Chuck erkennt einen eingehenden Bildtelefonanruf für Chuck. Er erkennt anhand der Ortsdaten, die er von Alice und Chuck überwacht, und anhand der Daten, die er von der Überwachung der hausinternen Geräte erhält, dass Alice gerade fernsieht. Der Bildanruf kann also nicht auf dem Fernseher dargestellt werden. Er sucht nach weiteren Ausgabegeräte und findet den PDA mit Headset, den Chuck bei sich trägt. Also übermittelt er den Anruf dorthin, damit Alice weiterhin ungestört fernsehen kann.

Soft Handover: Beim Spaziergang durch die Stadt telefoniert Theo mit seinem Handy über Voice-over-IP und UMTS. Während der Empfang schlechter wird, sucht das Mobiltelefon andere Verbindungsmöglichkeiten, findet ein teureres GSM-Netz und initiiert einen Soft Handover, damit Theo ungestört weitersprechen kann. Ein Piepton informiert ihn dabei, dass jetzt höhere Gebühren anfallen. Als sich Theo seiner Wohnung nähert, erkennt das Telefon das heimische und damit kostenfreie W-Lan und initiiert abermals einen Soft Handover zu einer Voice-over-IP-Verbindung, diesmal eben über W-Lan.

Restaurant-Finder: Thomas ist in einer fremden Stadt auf der Suche nach einem Restaurant. Der Restaurant-Finder-Dienst, den er über seinen PDA aufruft, lokalisiert ihn, ruft sein Profil ab, um seine Restaurant-Vorlieben zu erfahren, besorgt sich Wetterdaten und Daten über die Auslastung der in Frage kommenden Restaurants in der Nähe. Dann schlägt er ihm fünf passende Restaurants und Biergärten vor, die er zu Fuß oder mit der Straßenbahn in weniger als 15 Minuten erreichen kann.

Museumstour: Bei seiner Tour durch ein Museum erhält Axel Informationen über die Ausstellungsstücke, vor denen er sich gerade befindet über das Headset seines PDA in seiner Sprache. Vitalian, der kein Headset dabei hat, liest die Informationen in seiner Sprache auf dem PDA-Display ab.

Lastverteilung: Ein Online-Shop wird durch einen sich selbst organisierenden Cluster von Rechnern realisiert. Jeweils einer davon fungiert als Dispatcher, der von den anderen regelmäßig Informationen über deren Auslastung erhält und entsprechend die bei ihm eingehenden Http-Requests weiter verteilt. Das Ziel ist die möglichst gleichmäßige Verteilung der Last. Jeder der Rechner testet in Intervallen, ob der Dispatcher noch ordnungsgemäß funktioniert. Ist dies nicht mehr der Fall, werden die übrigen Rechner informiert, kooperativ ein neuer Rechner bestimmt, der die Dispatcher-Aufgabe übernimmt, und alle eingehenden Http-Requests in Zukunft über diesen geleitet.

Staubsaugroboter: Rüssel2D2 ist ein autonomer Staubsaugroboter. Seine eigenen Sensoren helfen ihm, Objekten auszuweichen, während er durch die Zimmer fährt und Teppiche saugt. Aufenthaltsdaten aus dem Wohnungssystem teilen ihm mit, wann niemand zu Hause ist, und er saugen kann: Einerseits selbst ungestört, andererseits ohne andere zu stören.

Intelligente Verkehrsleitung: FritzFritz ist ein Navigationssystem, das nicht nur Staudaten sammelt, über Karteninformationen verfügt und die aktuelle Position kennt. Aufgrund der Information, welche anderen Verkehrsteilnehmer sich auf seiner aktuellen Route bewegen, welches Ziel sie haben, und wie sie sich aktuell bewegen, kann er die eigene

Route und Geschwindigkeit selbsttätig anpassen. Dadurch wird der Durchsatz aller Fahrzeuge einer Strecke optimiert und es werden Staus verhindert, die durch unkoordiniertes Bremsen und Beschleunigen entstehen.

3.2 Die Kontextwertschöpfungskette als Vorarbeit

Hegering et al. [HKLPR03] haben eine Wertschöpfungskette für Kontextinformationen vorgestellt und am Beispiel eines Medizinischen Ratgeber- und Notfallsystems beschrieben (siehe Abb. 3.1). Der Kontext einer Entität wird in einer Messphase von verschiedenen Quellen erfragt, etwa von physikalischen Sensoren, über Lokalisierungsmethoden wie das *Global Positioning System (GPS)* oder aus Datenbanken mit medizinischen Profilen. Das erhaltene Rohmaterial wird als *low-level context information* bezeichnet, das in einem oder mehreren Verfeinerungsschritten zu *high-level context* wird, der sich durch einen höheren Abstraktionsgrad auszeichnet. In der Aggregationsphase werden alle Informationen, die sich auf eine Entität beziehen (im Beispiel der Patient „X“) und relevant für einen bestimmten kontextsensitiven Dienst sind, zusammengefasst („aggregiert“). Die Anpassung des Dienstverhaltens an den erfahrenen Kontext (im Beispiel wird ein Alarm ausgelöst) ist der letzte Schritt (”´kontextualisieren“).

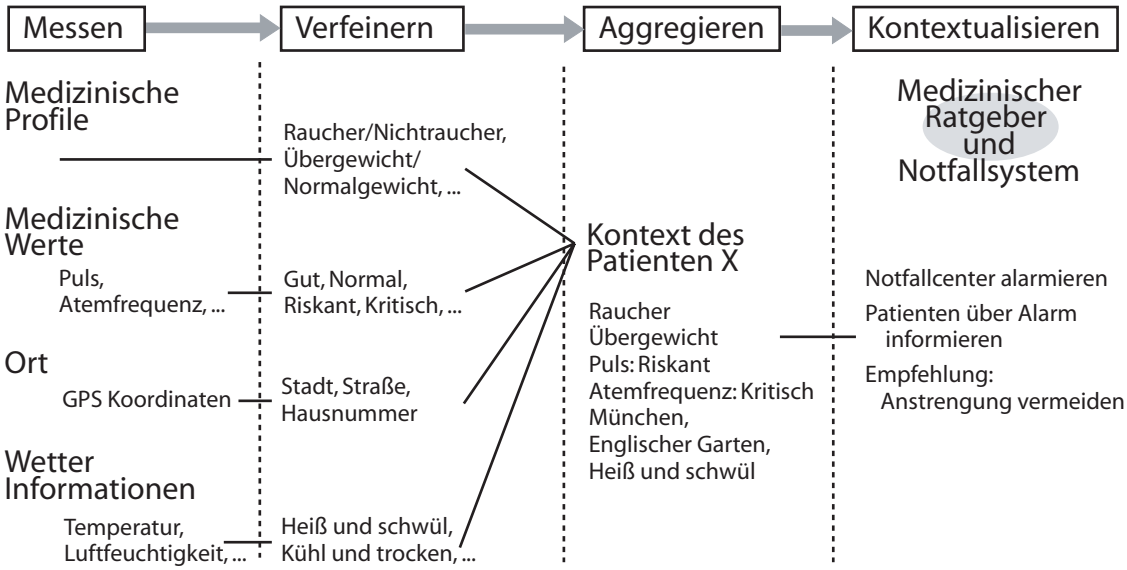


Abbildung 3.1: Die Kontextwertschöpfungskette nach Hegering et al. [HKLPR03].

Jeder der vier Schritte kann Unterschritte enthalten und auch mehrmals hintereinander ausgeführt werden. Dabei durchläuft jede Kontextinformation ihre eigene Kette dieser vier Schritte, unabhängig von den anderen Kontextinformationen. Die Steuerung der verschiedenen Bearbeitungsketten hängt von verschiedenen Umständen ab, etwa den Anforderun-

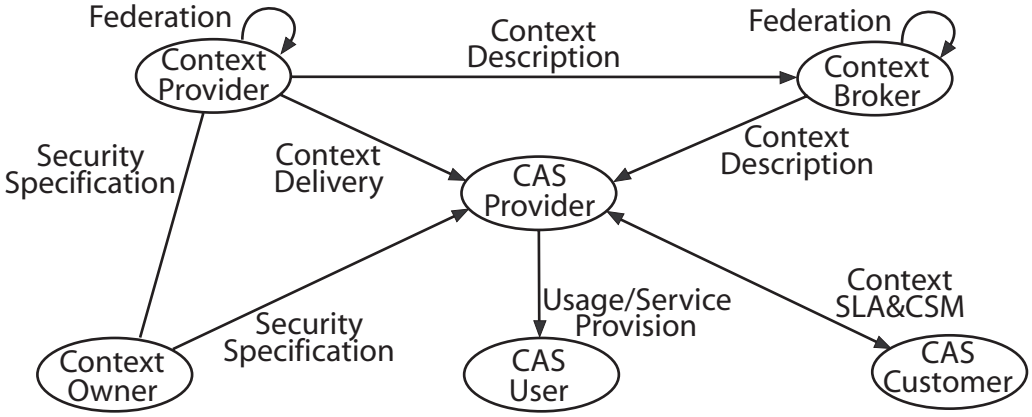


Abbildung 3.2: Rollenmodell nach Hegering et al. [HKLPR03].

gen des kontextsensitiven Dienstes, die Menge der verarbeiteten Kontextinformationen, der Verfügbarkeit von Kontextquellen und den Abhängigkeiten der Informationen untereinander.

Abgesehen von trivialen Diensten können kontextsensitive Systeme nur organisationsübergreifend realisiert werden, so die Schlussfolgerung (was wiederum offene Systeme bedingt). Um die daran beteiligten Akteure einzuordnen anhand ihrer Rolle, die sie in so einer Föderation spielen, haben Hegering et al. ein Rollenmodell (siehe Abb. 3.2) für kontextsensitive Dienste erstellt.

Die zentrale Rolle nimmt dabei der *CAS Provider* ein, der Anbieter von kontextsensitiven Diensten. Er entwickelt und betreibt kontextsensitive Dienste, die er dem Kunden, dem *CAS Customer* anbietet und verkauft. Dieser trifft Dienstvereinbarungen im Auftrag des Benutzers, des *CAS User*. Der Anbieter von kontextsensitiven Diensten erhält die Kontextinformationen von einem Kontextanbieter (*Context Provider*), der in der Regel mehrere Kontextquellen betreibt.

Da es unwahrscheinlich ist, dass alle benötigten Kontextinformationen von dem selben Kontextanbieter zu erhalten sind, wird ein Vermittler dazwischen geschaltet, der Kontextbroker (*Context Broker*). Dieser ist in diesem Rollenmodell nur dazu da, die Suche nach einem geeigneten Kontextanbieter zu unterstützen, indem er ein Verzeichnis der Beschreibungen der Kontextinformationen vorhält, die die jeweiligen Anbieter liefern können. Die tatsächliche Anforderung und Auslieferung von Kontextinformationen läuft dann ausschließlich zwischen Kontextanbieter und dem Anbieter kontextsensitiver Dienste ab. Dabei gibt es noch eine weitere Rolle zu berücksichtigen, die des Kontextbesitzers (zum Beispiel die Person, dessen Ort ermittelt werden soll). Die Auslieferung der Kontextinformation kommt nur zustande, wenn die Vereinbarungen des Kontextbesitzers mit dem Kontextanbieter (*Security Specifications*) dies erlauben.

Das im Weiteren hier vorgestellte Konzept baut auf diesen Vorarbeiten auf, unterscheidet sich jedoch in mindestens einem entscheidenden Punkt: Die Rolle des Vermittlers wird

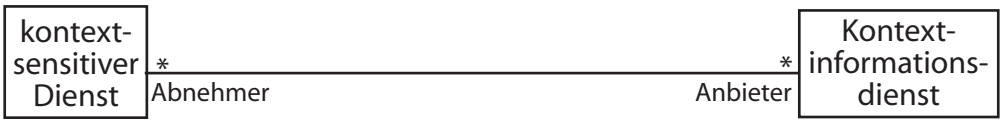


Abbildung 3.3: Beteiligte bei der Kontextbereitstellung ohne...

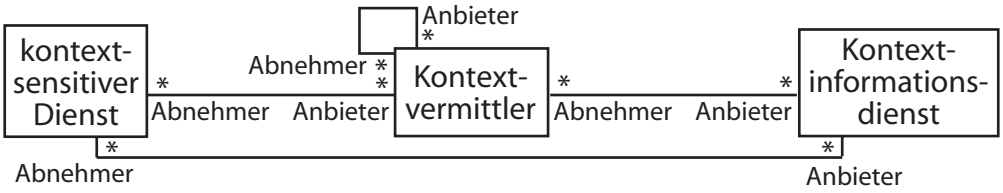


Abbildung 3.4: ... und mit Vermittler.

aufgewertet. Er erledigt nicht nur Hilfsfunktionen eines Verzeichnisdienstes, sondern wird zwischen Kontextanbieter und kontextsensitiven Dienst geschaltet, so dass er dem kontextsensitiven Dienst gegenüber stellvertretend die Anbieterrolle übernimmt (siehe Abb. 3.3 und Abb. 3.4). Dieser Unterschied liegt darin begründet, dass der Vermittler zusätzliche Aufgaben hat, zum Beispiel in Bezug auf Privacy oder Accounting, und die Informationen bei Bedarf auch veredelt: Etwa durch Aggregation von Informationen oder durch die Transformation in ein anderes Repräsentationsformat. Dabei kann sich jeder Vermittler wiederum an einen weiteren Vermittler wenden, so dass eine Reihung von Vermittlern für den den Vermittlungsprozess entsteht. Dabei wird unterschieden:

Definition (Kontextinformationsdienst)
Dienste, die nicht nur stellvertretend als Vermittler agieren, sondern direkt von logischen und physischen Kontextquellen gesammelte Daten umwandeln und so originär als Kontextinformationen bereit stellen, heißen Kontextinformationsdienste. Physische Kontextquellen sind dabei Sensoren, die Zustände der realen Welt abtasten (wie Bewegungsmelder oder Temperatursensoren). Logische Kontextquellen fragen Informationen ab, die bereits elektronisch verfügbar sind (etwa von Datenbanken, Inferenzsystemen oder Profilservern).

3.3 Generisches Szenario

Allen vorhin genannten Beispielen ist gemeinsam, dass Kontextinformationen bereit gestellt werden müssen. Dies soll losgelöst von diesen Beispielszenarien nun abstrakt in einem generischen Szenario für die Anforderung und die Bereitstellung von Kontextinformationen detailliert betrachtet werden. Es folgen die dafür notwendigen Prozessschritte.

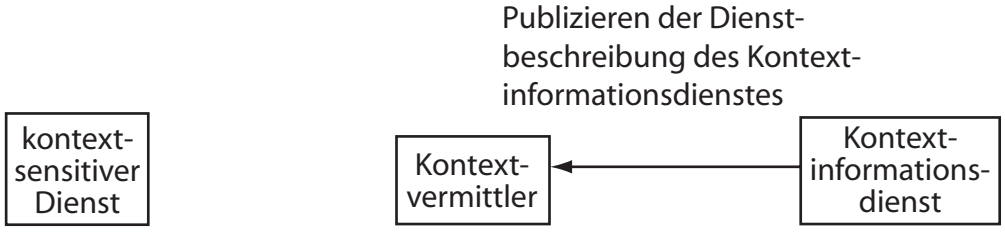


Abbildung 3.5: Szenario: Veröffentlichung des Kontextangebots.

3.3.1 Prozessschritte

Publikation des Kontextinformationsdienstes

Bevor es zur Kooperation zwischen Kontextinformationsdiensten und kontextsensitiven Diensten kommen kann, muss der Kontextinformationsdienst zunächst der Welt bekannt machen, welche Kontextinformationen er anbieten kann (siehe Abb. 3.5). Um den leichtgewichtigen kontextsensitiven Dienst von der Aufgabe der Dienstsuche zu entlasten, geht diese Dienstbeschreibung an die Kontextvermittler. Diese Beschreibung muss im Aufbau korrelieren mit den Kontextanfragen von kontextsensitiven Diensten.

Erstellen und Versenden der Kontextanforderung

Der kontextsensitive Dienst erstellt die Anforderung einer Kontextinformation. Die folgenden Bestandteile sind dabei notwendig.

- Die Anforderung enthält die eindeutigen Angabe, auf **welche Entität** sich die Kontextinformation beziehen soll.
- Sie enthält ferner die Spezifikation, welche Art der Aussage über diese Entität die Kontextinformation treffen soll, also zu welcher **Kontextinformationsklasse** die Information gehören soll. Dazu gehören optional auch Bedingungen über die Qualitätseigenschaften der Kontextinformation (zum Beispiel auf wie viele Meter genau eine Ortsangabe sein soll).
- Außerdem ist eine Angabe notwendig, für welchen **Zeitpunkt** die angeforderte Aussage gültig sein soll. Im Regelfall werden aktuelle Informationen angefordert.
- Schließlich wird spezifiziert, **wann** die Kontextinformation zugesandt werden soll. Im einfachen Falle („sofort“) bedeutet das, dass die Informationsbeschaffung *pull*-basiert ist. Andernfalls werden zeitliche oder kausale Ereignisse definiert, die die (einmalige oder wiederholte) Zusendung der Kontextinformation auslösen. Dann ist die Informationsbeschaffung *push*-basiert.

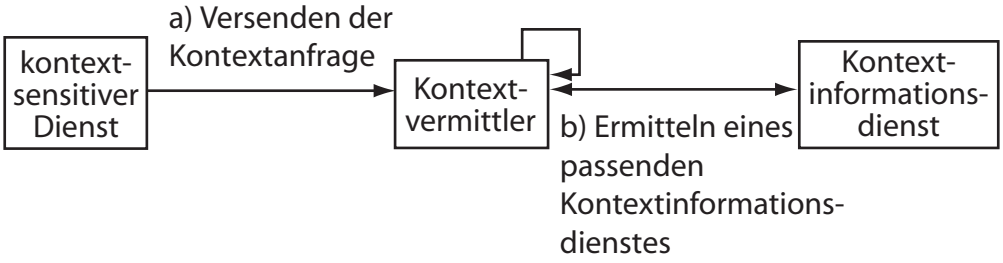


Abbildung 3.6: Szenario: Suche nach einer Kontextinformation.

Versenden der Anfrage

Schließlich sendet der kontextsensitive Dienst die Anfrage nach einer Kontextinformation an einen geeigneten Kontextanbieter. Anbieter von Kontextinformationen sind entweder Kontextinformationsdienste, die bestimmte Kontextinformationen an einer Dienstschnittstelle bereit stellen. Oder es sind Kontextvermittler (siehe Abb. 3.6), wenn der kontextsensitive Dienst keinen direkten Kontextinformationsdienst für die gerade benötigte Kontextinformation kennt. In veränderlichen Umgebungen mit mobilen Diensten und Nutzern ist vom zweiten Fall als Regelfall auszugehen (siehe Abb. 3.6).

Der Kontextvermittler

Ein Kontextvermittler übernimmt eine typische Stellvertreterrolle ähnlich einem Proxyserver. Dabei agiert er gegenüber dem kontextsensitiven Dienst selbst als Anbieter von Kontextinformationen und verschattet alle Prozesse, die er benötigt, um selbst die Kontextinformationen von anderen Anbietern zu erhalten. Deshalb werden für die Bereitstellung von Kontextinformationen die Rollen „Anbieter“ und „Abnehmer“ definiert (siehe Abb. 3.3), die unabhängig davon sind, von welchen Komponenten sie ausgefüllt werden. Ein Kontextvermittler ist zwar auch Abnehmer, aber dennoch kein kontextsensitiver Dienst, da er die Kontextinformationen lediglich weiterleitet, ohne eine eigene Adaption zu erfahren.

Dabei gibt es verschiedene Optionen, wie ein Kontextvermittler mit einer Anfrage nach einer Kontextinformation umgehen kann.

- Zunächst gibt es die Möglichkeit der reinen Vermittlung. Das heißt, der Vermittler startet nach Eingang einer Kontextanfrage eine eigene Anfrage bei einem anderen Kontextanbieter. Erhält er die Kontextinformation, reicht er sie dann an seinen Abnehmer durch.

  Bei dieser Variante besteht die Leistung des Vermittlers entweder darin, den geeigneten Kontextanbieter zu finden. Oder der Vermittler wird als Wächter eingesetzt, um zu verhindern, dass auf die Kontextanbieter direkt zugegriffen wird. Dies kann der Fall sein, um zentral prüfen zu können, ob ein Abnehmer eine Kontextinformation überhaupt erhalten darf, oder ob damit der Datenschutz des Kontextbesitzers

verletzt würde. Oder der Vermittler übernimmt die Abrechnungsfunktionalität stellvertretend für den Anbieter. Es sei angemerkt, dass in keinem der Fälle es zwingend ist, dass Kontextanbieter und Kontextvermittler zu der selben Domäne gehören.

- Daneben liegt es nahe, den Vermittler mit einem Cache auszurüsten, in dem er Informationen speichert, die er bereits vermittelt hat. Befindet sich eine angefragte Kontextinformation in seinem Cache, so braucht der Vermittler den Anbieter nicht erneut kontaktieren.

- Der Vermittler kann diesen Cache nicht nur mit den Informationen füllen, die bei tatsächlichen Kontextanfragen anfallen, sondern bereits im Voraus Kontextinformationen bei Anbietern besorgen, von denen er erwartet, dass Anfragen nach ihnen bei ihm eintreffen werden. Das ist vor allem der Fall, wenn ein Kontextvermittler sich auf einen räumlich oder thematisch eng gefassten Bereich spezialisiert, zu dem er Kontextinformationen anbietet.

- Im weiter gehenden Fall verfügt der Vermittler über Regeln, wie er eine angefragte Kontextinformation aus einer oder mehreren anderen Kontextinformationen erstellen kann. Für die Beschaffung jeder weiteren dafür notwendigen Kontextinformation steht dem Vermittler rekursiv das bisher beschriebene Instrumentarium zur Verfügung: Er kann in seinem Cache suchen, einen Anbieter befragen, oder mit Hilfe von Regeln die Information aus anderen Informationen schließen.

  Die meisten Autoren sprechen hierbei davon, dass aus niederwertigen („low-level“) höherwertige („high-level“) Kontextinformationen werden. Es sei angemerkt, dass es keine Messgröße dafür gibt, wie hoch- oder niederwertig eine Kontextinformation ist. Wenn aus der Information darüber, dass sich Person A in der Oettingenstraße 67 in München befindet, geschlossen wird, diese Person befindet sich in Europa, so ist die Ortsangabe „Europa“ aus Sicht des anfordernden Dienstes die höherwertige. Und das obwohl die Ausgangsinformation doch um ein Vielfaches genauer war.

Suche nach einem geeigneten Kontextinformationsdienst

Doch egal, ob der Kontextvermittler die Kontextinformation in seinem Cache direkt verfügbar hat, oder sie aus verfügbaren Kontextinformationen ableitet – bevor er über eine Kontextinformation verfügt, muss er sie sich selbst zunächst von einem Kontextinformationsdienst besorgen. In diesem Fall agiert er wie ein kontextsensitiver Dienst: Er erstellt die Anfrage und sendet sie an einen Kontextanbieter, der wiederum ein Kontextvermittler sein kann, oder direkt ein Kontextinformationsdienst. Anders als der kontextsensitive Dienst (der etwa über ein Verzeichnis der wichtigsten Kontextvermittler verfügen kann) muss er in der Lage sein, diese Kontextanbieter zu suchen und zu finden. Dazu erstellt er eine Dienstsuchanfrage, in der er den gewünschten Dienst beschreibt. Diese Beschreibung

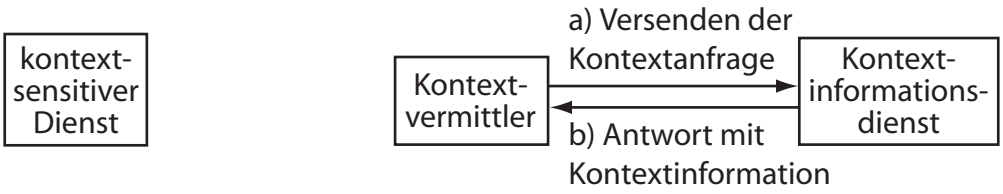


Abbildung 3.7: Szenario: Bereitstellung einer Kontextinformation.

muss ein Dienstvermittlungssystemen mit den Beschreibungen von Kontextinformationsdiensten abgleichen, von denen es Kenntnis hat, und eine geeignete Auswahl treffen (siehe Abb. 3.6).

- Diese Dienstbeschreibung eines Kontextinformationsdienstes enthält unter anderem die **Menge der Kontextinformationsklassen**, zu denen Kontextinformationen bereit gestellt werden können.
- Ferner beschreibt sie die **Menge der Entitäten**, über die Kontextinformationen bereit gestellt werden können.
- Außerdem trifft sie Aussagen über zusicherbare oder wahrscheinliche **Qualitätskriterien** der Kontextinformationen (siehe auch Abschnitt 2.1.3).

Abrufen der Kontextinformation

Hat der Kontextvermittler geeignete Kontextinformationsdienste gefunden, erstellt er (wie zuvor der kontextsensitive Dienst) eine entsprechende Kontextanfrage, versendet sie und erhält in der Regel dann die gewünschte Kontextinformation, die er nach eventueller Prüfung direkt an den kontextsensitiven Dienst weiterleitet (siehe Abb. 3.7) und gegebenenfalls zur mehrmaligen Verwendung selbst in einem Cache speichert.

Kontextinformation steht nicht direkt bereit

Kann ein Kontextvermittler eine Kontextinformation nicht direkt besorgen, weil er sie nicht bereits verfügbar hat und außerdem keinen Kontextinformationsdienst findet, der sie ihm bereit stellt, oder diese Bereitstellung schlägt fehl, dann kann der Kontextvermittler versuchen, auf andere Weise diese (oder eine ähnliche) Kontextinformation zu erstellen (siehe Abb. 3.8). Diese kann er aus anderen Kontextinformationen ableiten, über die er verfügt, oder die von Kontextinformationsdiensten bereit gestellt werden können.

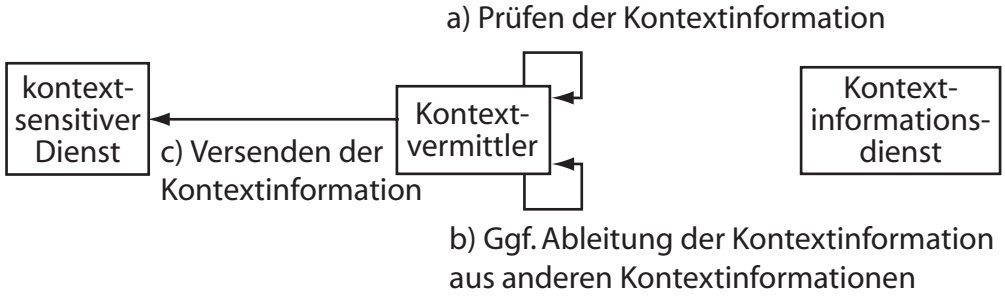


Abbildung 3.8: Szenario: Wichtige Vermittlerrolle.

3.3.2 Einordnung dieser Arbeit

Diese Arbeit schafft Voraussetzungen, die zur Umsetzung des eben beschriebenen Szenarios zwingend notwendig sind:

- Damit entschieden werden kann, ob eine angebotene Kontextinformation die Kontextinformation ist, die der Abnehmer verlangt, bedarf es eines gemeinsamen Verständnisses über die Information und damit auch einer gemeinsamen **Modellierung von Kontextinformationen**. Diese ist bei allen Prozessschritten zwischen zwei beteiligten Komponenten notwendig, um Interoperabilität zu gewährleisten: Bei der Publikation der Beschreibung eines Kontextinformationsdienstes, bei der Erstellung und dem Versand einer Kontextanfrage, bei der Suche nach geeigneten Kontextinformationsdiensten, beim Austausch einer Kontextinformation und bei der Ableitung einer Kontextinformation aus anderen. In Kapitel 5 wird mit CMPlus eine geeignete Modellierung vorgestellt.

- Damit die passenden Kontextinformationsdienste von Abnehmern gefunden werden, müssen sie in der Lage sein, zu veröffentlichen, welche Kontextinformationen sie anbieten. Mit dem Dienstprofil CISP, das in Kapitel 6 erläutert wird, wird es möglich, entsprechende **Dienstbeschreibungen von Kontextinformationsdiensten** zu erstellen.

- Schließlich adressiert diese Arbeit auch die diesem *best-effort*-Szenario inne wohnende Gefahr, dass eine benötigte Kontextinformation nicht direkt verfügbar ist. Dafür werden in Kapitel 7 **Ausweichstrategien zur Bereitstellung von Kontextinformationen** entwickelt und diskutiert.

Im eben vorgestellten Prozess wird noch nicht die Verteilung der Kontextvermittler und die damit verbundene Möglichkeiten zur Föderation dieser Komponenten betrachtet. Dieses Problem klammert diese Arbeit bewusst noch aus, allerdings eingedenk der Tatsache, dass dies der nächste notwendige Schritt zur Schaffung von Kontextsensitivität in offenen, ubiquitären Systemen sein muss.

Ebenfalls ausgeklammert werden Fragen des Schutzes der Privatsphäre und des Bezahlens für Kontextinformationen. Auch dies geschieht allein, weil es den Umfang dieser Arbeit sprengen würde, und nicht, weil diese Fragen ohne Belang wären. Der Autor ist der Ansicht, dass auch dieses weitere zentrale Probleme sind.

3.4 Verwandte Arbeiten zur Bereitstellung und Verteilung von Kontextinformationen

Bei der folgenden Diskussion verwandter Arbeiten zur Verteilung von Kontextinformationen ist zu beachten, dass die jeweiligen Autoren unter Umständen mehr oder weniger abweichende Definitionen der zentralen Begriffe von Kontext, Kontextinformation usw. haben.

Pionierarbeiten kontextabhängiger Systeme

Zunächst soll eine Auswahl von Arbeiten vorgestellt werden, die zu den ersten einflussreichen Arbeiten auf dem Gebiet des *Ubiquitous Computing* gehören. Sie waren meist darauf angelegt, in größeren aber überschaubaren Umgebungen, wie Firmengelände oder Gebäuden eine Experimentalumgebung für ortsabhängige Dienste zu erstellen. Sie waren allesamt kaum oder gar nicht darauf aus, Kontextinformationen über Domänengrenzen hinweg auszutauschen, sondern waren meist auf spezielle Anwendungen oder enge Anwendungsfelder zugeschnitten. Alle würden somit Probleme aufwerfen, wollte man die Systeme auf ein globales, offenes Umfeld übertragen.

„Context Toolkit“ Das „*Context Toolkit*“ ist von Anind Dey und Gregory Abowd [DA99, DA00] am Georgia Institute of Technology in Atlanta entwickelt worden. Es soll zur Entwicklung und Installation von kontextsensitiven Diensten eingesetzt werden und bietet dazu neben einer graphischen Benutzerschnittstelle drei Klassen von Komponenten: *context widgets*, die die Sensorik verschatten und nach außen fertige Kontextinformationen liefern, *context servers* die dafür verantwortlich sind, den kompletten „Kontext einer Entität“ (man beachte den anderen, eingeschränkten Kontextbegriff) zu aggregieren und *context interpreters*, die Kontextinformationen verarbeiten, transformieren und neue Informationen ableiten.

„Active Badges“ Die ersten „*active badges*“ sind von 1989 bis 1992 an den Olivetti Research Labs (inzwischen AT&T) entwickelt worden. Kleine Anstecker für Personen und Geräte sandten alle zehn Sekunden ein individuelles Infrarot-Signal aus. Über entsprechend vernetzte Sensoren in den Büroräumen der Testumgebung konnte somit der jeweilige Aufenthaltsort verfolgt werden [WHFG92].

„Xerox ParcTab“ Die Entwickler am Xerox Palo Alto Research Center nutzten für ihre eigene Experimentalumgebung ebenfalls Infrarotübertragung zu den tragbaren Endgeräten der Benutzer. Nur dass deren „Tabs“ (heute würde man sie „PDAs“ nennen) rechenfähige Geräte waren und somit Plattform für verschiedene einfache *Ubiquitous-Computing*-Anwendungen sein konnten [WSA+95].

„Cooltown“ Auf Infrarot-Signale setzten auch die „Cooltown“-Entwickler von HP. Ihre Idee war es, für alle möglichen Orte, Objekte und Geräte Repräsentationen im Internet zu schaffen. Infrarotsender, die an den Objekten angebracht waren, sandten entweder direkt die entsprechende URL für das Objekt, oder einen Identifier, der auf die URL abgebildet werden konnte. Über W-Lan-Zugriff auf das Internet ließen sich dann sofort zum Beispiel Informationen über die jeweiligen Objekte aufrufen. Im Gegensatz zur netzbasierten Lokalisierung (vgl. [Küp05]) der *Badges* und *Tabs* findet hier die Ortung (Im Sinne von: „In der Nähe welcher Entitäten befinde ich mich gerade?“) im mobilen Endgerät der Benutzer statt [KBM+02].

„Neural Network House“ Im Neural Network House in Colorado haben Michael Mozer et al. [Moz98, MDA+95] eine intelligente Umgebung geschaffen, in der Beleuchtung und Klima sowohl automatisch geregelt, als auch von den Benutzern gesteuert werden. Das Projekt – eines von vielen „Smart-House“-Projekten – war eines der ersten, das auch Maschinenlernen setzte. Es versuchte, aus dem Wissen über das Eingreifen der Nutzer in der Vergangenheit und den jeweils gemessenen Auswirkungen zu lernen und daraus Entscheidungen für zukünftige Regelungsvorgänge abzuleiten.

Ziel war es, Beleuchtung- und Klima so zu regeln, dass zum einen die Bewohner die Umgebung als angenehm empfinden, gleichzeitig aber der Energieverbrauch so niedrig wie möglich gehalten wird. Kontextinformationen kamen aus einfachen Klimasensoren in den Zimmern sowie Bewegungsmeldern, aus denen geschlossen wurde, ob sich in einem bestimmten Raum jemand befindet oder nicht.

Die Architektur der „Technology for Enabling Awareness“

„Techonolgy for Enabling Awareness“ (TEA) ist ein gemeinsames Forschungsprojekt der Universität Karlsruhe, des Forschungslabors „Starlab“ (Brüssel), und den beiden Firmen Nokia (Finnland) und Omega (Italien). Erklärtes Ziel war es unter anderem, leicht zu integrierende Komponenten zu entwickeln für Mobiltelefone, PDAs und mobile Rechner im Allgemeinen, die diesen Geräten einen nützlichen Grad an Kontextsensitivität ermöglichen. „Sensitivität“ bedeutet für TEA in diesem Zusammenhang die dauernde Verfügbarkeit des gerade aktuellen lokalen Nutzungskontextes.

Kristof Van Laerhoven und Kofi Aidoo [LA01] haben einen Überblick über die TEA-Architektur gegeben (siehe Abb. 3.9). Die Forscher haben mit kleinen und billigen Sensoren experimentiert, etwa zur Messung von Licht, Temperatur, Beschleunigung, Lautstärke

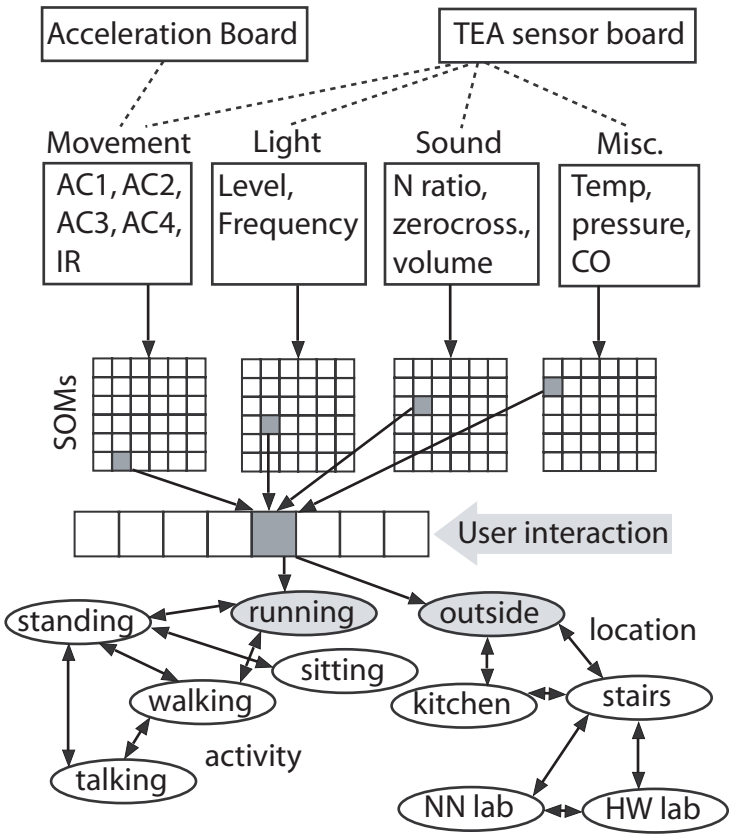


Abbildung 3.9: Die TEA-Architektur nach [LA01] mit selbstorganisierenden Karten.

oder Luftdruck. Diese sind auf ein Sensorbrett gebaut worden. Das System verknüpft nun die Ausgaben dieser Sensoren und bildet sie ab auf Beschreibungen des Kontexts wie: „Gehen im Keller“.

Dazu benutzt das System selbstorganisierende Karten („SOMs“ in Abb. 3.9 für *self-organizing maps*), eine Spezialform der künstlichen neuronalen Netze, um typische Sensordaten-Kombinationen zu Situationen zu Clustern, die der Benutzer dann mit Namen versieht (siehe Abb. 3.10). Die nächste Schicht überwacht die möglichen Zustandsübergänge mit Hilfe eines Probabilistischen Endlichen Automaten. Das Modell kennt Wahrscheinlichkeiten für die Übergänge der Kontextzustände. Ist eine Veränderung nicht wahrscheinlich, wird der nächste Zustand erst erreicht, nachdem die Veränderung mehrmals angezeigt worden ist.

Ist keine Lernphase zur Erkennung und Beschreibung der verschiedenen Kontextsituationen notwendig, weil diese bereits bekannt sind, so beschreibt Albrecht Schmidt [Sch02], dass das einfache Verfahren „*Nearest Neighbour Matching*“ ausreicht, um für den jeweiligen Eingabevektor den „nächsten“ Referenzvektor für eine Situation zu errechnen.

Diese Form, sich neuronaler Netz zur Ableitung höherwertigen Kontextwissens aus nie-

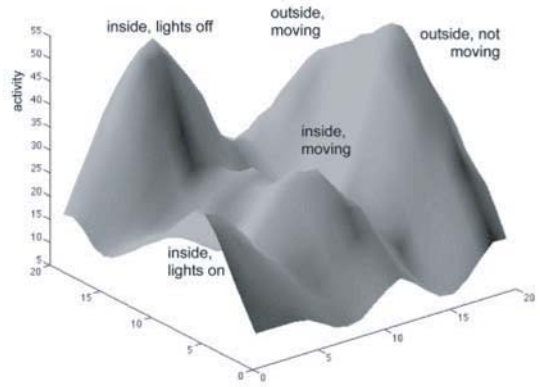


Abbildung 3.10: Die Cluster der verschiedenen TEA-Kontextzustände [Sch02].

derwertigeren Sensorinformationen zu bedienen, ist ein sehr spannender und viel versprechender Ansatz. Außerdem ist bereits im Fokus, dass die verwandte Hardware günstig und möglichst klein sein muss. Leider widerspricht die feste Konfiguration von – noch dazu auf einem Brett montierten – Sensoren der Vorstellung vom dynamischen Zusammenspiel von Sensoren und Diensten. Ein TEA-System könnte aber in Systemen, wie sie diese Arbeit untersucht, als einzelner Anbieter von Kontextinformationen fungieren, der an einem festen Ort installiert oder einer festen Entität zugeordnet ist.

Die „Contextual-Information-Service"-Architektur

AURA ist ein Forschungsverbund an der Carnegie Mellon Universität in Pittsburgh, der von folgender Prämisse ausgeht: Die inzwischen wertvollste Ressource eines Computersystems ist die Aufmerksamkeit des Benutzers, die im Gegensatz zu anderen Ressourcen überhaupt nicht gemäß dem Mooreschen Gesetz wächst. Forschungsziel, auf das hin mehrere Projekte arbeiten wie etwa das Coda Dateisystem [Sat90] ist die Verwirklichung eines Konzeptes namens *Personal Information Aura*. Diese soll den Benutzer intuitiv umgeben mit Rechen- und Informationsdiensten, die verfügbar sind, egal an welchen Ort sich der Benutzer begibt.

Glenn Judd und Peter Steenkiste stellen einen Dienst vor, den sie *Contextual Information Service* [JS03] nennen, der Anwendungen mit Kontextinformationen versorgt, als wäre er eine Datenbank. Dazu haben sie eine Anfrageschnittstelle entwickelt, die sie *Contextual Service Interface* (CSInt) nennen. Komplexe Anfragen von *clients* werden vom *CIS Query Synthesizer* zerlegt in eine oder mehrere einfache Anfragen, die dann an die Anbieter von Kontextinformationen weitergeleitet werden. Die Ergebnisse werden von dem *Synthesizer* wieder aufgesammelt, zusammengesetzt und an den entsprechenden *client* weitergeleitet.

Daneben wurde zur direkten Anfrage an relationale Datenbanken ein CSInt-SQL-Wrapper entwickelt. Judd und Steenkiste merken an, dass Datenbanken nur für relative statische

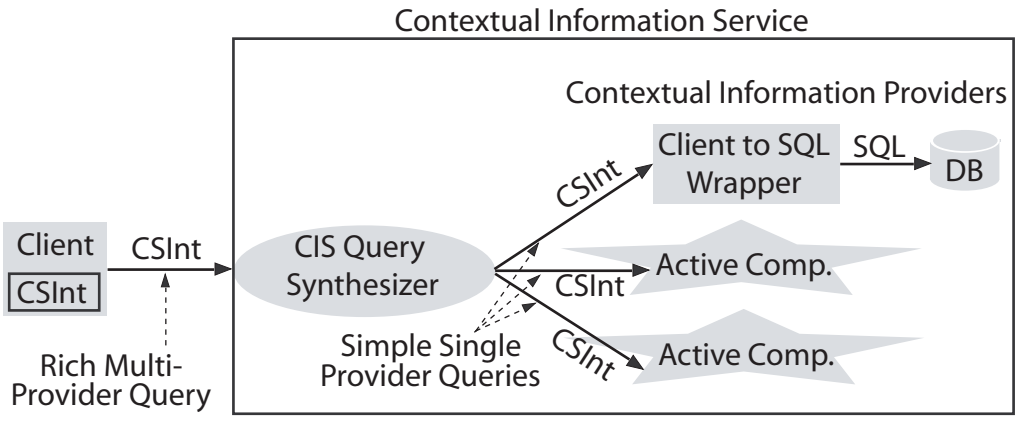


Abbildung 3.11: Die *Contextual-Information-Service*-Architektur.

Informationen geeignet seien. Die Autoren geben an, auf jeder Ebene (*Client*, *Synthesizer* und *Provider*) das Cachen der Informationen zu unterstützen.

Diese Arbeit adressiert das Problem, dass kontextsensitive Dienste überfordert würden, bürde man jedem einzelnen von ihnen die Aufgabe auf, ihre Anfragen nach hochwertigen Kontextinformationen selbst in Anfragen nach niederwertigen Kontextinformationen umzuwandeln. Die *Synthesizer*-Komponente ähnelt dem Kontextvermittler dieser Arbeit, ist allerdings nicht in der Lage, die Zerlegung der Anfrage zu ändern, wenn die Unteranfragen erfolglos bleiben. Auch die dynamische Suche nach Anbietern und ein generisches Informationsformat fehlen diesem Ansatz.

Die „ContextWare“ Architektur

„Ambient Networks“ ist ein von der Europäischen Union geförderter Forschungsverbund (sechstes Rahmenprogramm), der Netzlösungen für *Mobile and Wireless Systems Beyond 3G* entwickeln möchte. Im von Ericsson geführten Konsortium aus 45 Partnern sind Hersteller, Händler, Forschungslabore und akademische Einrichtungen aus Europa und darüber hinaus vertreten. Für eines der Teilprojekte, die Entwicklung von kontextsensitiven Netzen, haben Annika Jonsson und andere die *ContextWare* Architektur als Ziel ihres Vorhabens vorgestellt [JGB+05].

Diese Architektur beinhaltet zwei kontextspezifische *Functional Areas (FA)* (siehe Abb. 3.12), die eine davon, die *Context Coordination FA (ConCord FA)*, bietet die Schnittstelle zu *Contextware Clients* und zu anderen *FA* innerhalb des *Ambient Network*, während die andere, die *Context Management FA (CM FA)*, die grundlegenden internen Operationen implementiert, die in dem System zur Kontextbereitstellung notwendig sind. Dabei wird für die *ConCord FA* ein neues Konzept der *Context Level Agreements (CLA)* eingeführt, die überwachen, welche Art Kontextinformationen zwischen verschiedenen Partnern ausgetauscht werden dürfen. Die *CLA* können zwischen funktionalen Einheiten innerhalb ei-

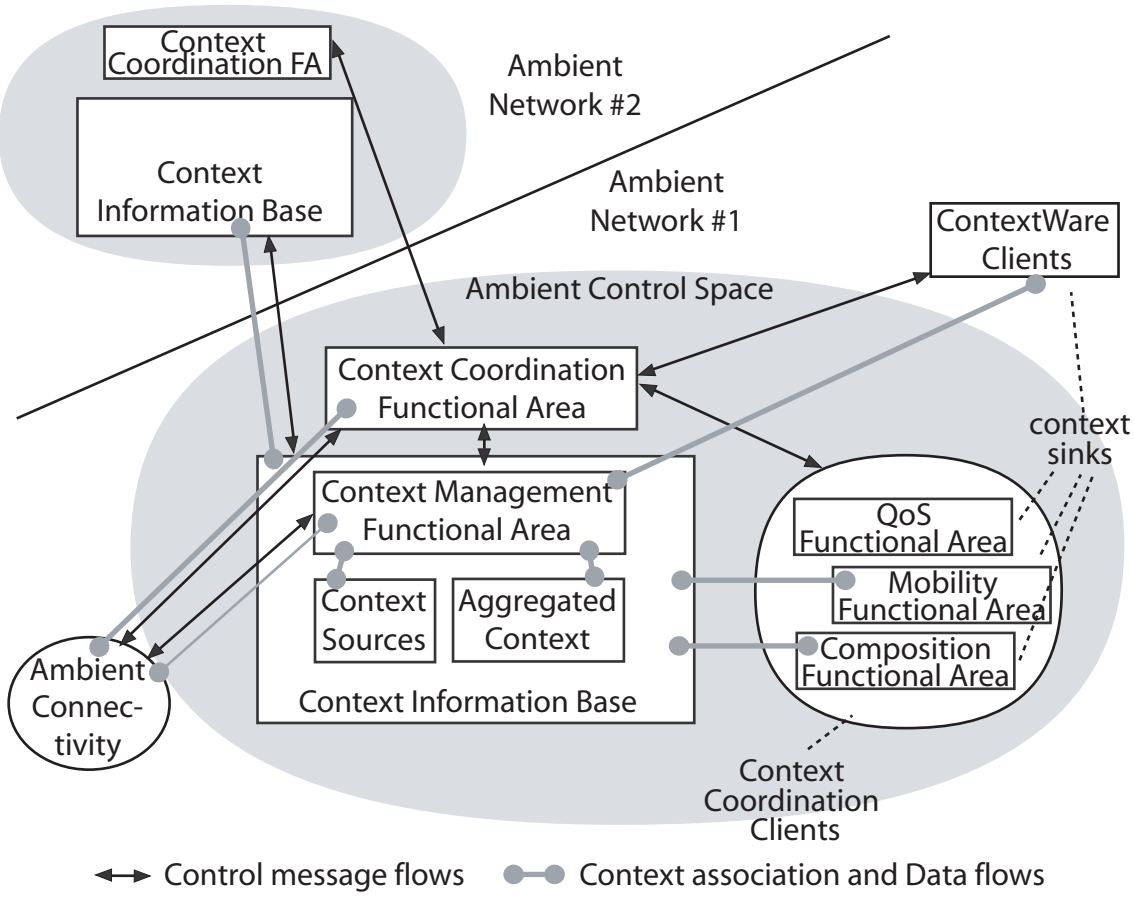


Abbildung 3.12: Architekturskizze für ContextWare.

nes *Ambient Networks* oder zwischen verschiedenen *Ambient Networks* ausgehandelt werden. *ContextWare* adressiert vor allem die Verteilung von Kontextinformationen in großflächigen Netzen und damit zwischen verschiedenen *Context Information Bases*. Nachdem diese Arbeit den Aspekt der Verteilung der Kontextvermittler und wie diese föderiert werden können nicht berücksichtigt, kann die *Contextware*-Architektur eine sinnvolle Erweiterung werden, mit der die Föderation von Kontextvermittlern ermöglicht wird. Die Ergebnisse bleiben abzuwarten.

Ninja Paths

Ganz ähnlich ist das Konzept der *Ninja Paths*, die eines der Grundelemente innerhalb der dienstorientierten *Ninja Architecture* [GWvB+01] darstellen. Das *Ninja Projekt* war ein Forschungsprojekt der Universität Berkeley in Kalifornien und hatte zum Ziel, Infrastrukturkomponenten für das Internet zu schaffen, um Dienste ubiquitär verfügbar, ausfallsicher und skalierbar zu machen. Hauptziel von *Ninja Paths* ist es, die Komposition

von Diensten zu unterstützen. Dabei besteht ein Pfad aus Operatoren, die Berechnungen auf Daten anstellen, und Konnektoren, die Protokoll-Übersetzungen zwischen Operatoren erledigen. Eine Komponente zur Automatischen Pfaderstellung erhält vom Benutzer die Spezifikation der Endpunkte des benötigten Pfades, eine teilweise sortierte Liste von Operationen, die auf dem Pfad erledigt werden müssen und Kostenlimits im Sinne von Latenzzeit, Rechnerleistung oder Speicheranforderungen. Aus diesen Eingaben wird dann automatisch zunächst ein logischer Pfad erstellt, dann auf physische Komponenten abgebildet und schließlich instanziiert.

Das Konzept, das stark an Agentensysteme erinnert, verteilt zwar auch Informationen, die Operationen durchlaufen, um umgewandelt zu werden, ist aber für die Verteilung und Komposition von Kontextinformationen im Sinne dieser Arbeit nicht tauglich. Schließlich kann der Weg der Informationen nicht ausgehend von der Quelle bestimmt werden, auch die Art und Weise der Verarbeitung von Kontextinformationen ist von der anderen Seite, der Abnehmerseite initiiert und bestimmt.

Das Projekt „Global Smart Spaces“

Global Smart Spaces (Gloss) ist ein Projekt von vier Partnern (Universität Strathclyde (Glasgow, Schottland), Trinity College Dublin (Irland), Universität St. Andrews (Schottland) und Universität Joseph Fourier (Grenoble, Frankreich)) innerhalb der *Disappearing Computer*-Initiative der Europäischen Union. Diese ist Teil der *Future-and-Emerging-Technologies*-Aktion (FET) innerhalb des *Information-Society-Technologies*-Programms (IST) der EU. Innerhalb von *Gloss* stellen Kirby und andere [KDM+03] eine *Active Architecture for Pervasive Contextual Services* vor. Herzstück ist ein *global contextual matching service* (siehe Abb. 3.13), der beschrieben wird als eine Entität, die ausgelöst durch den Empfang von *events* verschiedener Quellen einen neuen Datenstrom ausgehender *events* erschafft. Typischerweise sind die ausgehenden *events* semantisch aussagekräftiger als die eingehenden und beschreiben die Beziehung zwischen *input-events* und Fakten, die für einen kontextsensitiven Dienst relevant sind. Neben dem eingehenden Datenstrom arbeitet der *matching service* auch auf einer globalen Wissensdatenbank, die Elemente enthält wie einen Geo-Informations-Server, Webbasierte Systeme, Datenbanken und semistrukturierte Daten.

Zur Realisierung ihres Dienstes haben sich Kirby et al. für die Verwendung von verteilten *XML Pipelines* entschieden, deren Komponenten unabhängig voneinander hintereinander geschalten werden können. Damit bauen sie auf dem Prinzip der *Active Pipes* auf, das von Keller und anderen [KRWP01] eingeführt worden ist. Eine *Active Pipe* beschreibt die Übertragung und Verarbeitung eines Datenstroms als Sequenz von Funktionen, die auf dem Datenstrom ausgeführt werden. Jede dieser Funktionen korrespondiert mit einem Modul Programmcode, das entlang des Pfades zu instanziieren ist. Wobei eine *Active Pipe* nur einen logischen Ende-zu-Ende-Pfad beschreibt, der auf das darunter liegende physische Netz abgebildet werden muss. Dafür stellen Keller et al. eine Abbildungsvorschrift vor auf Basis eines Kürzester-Weg-Algorithmus.

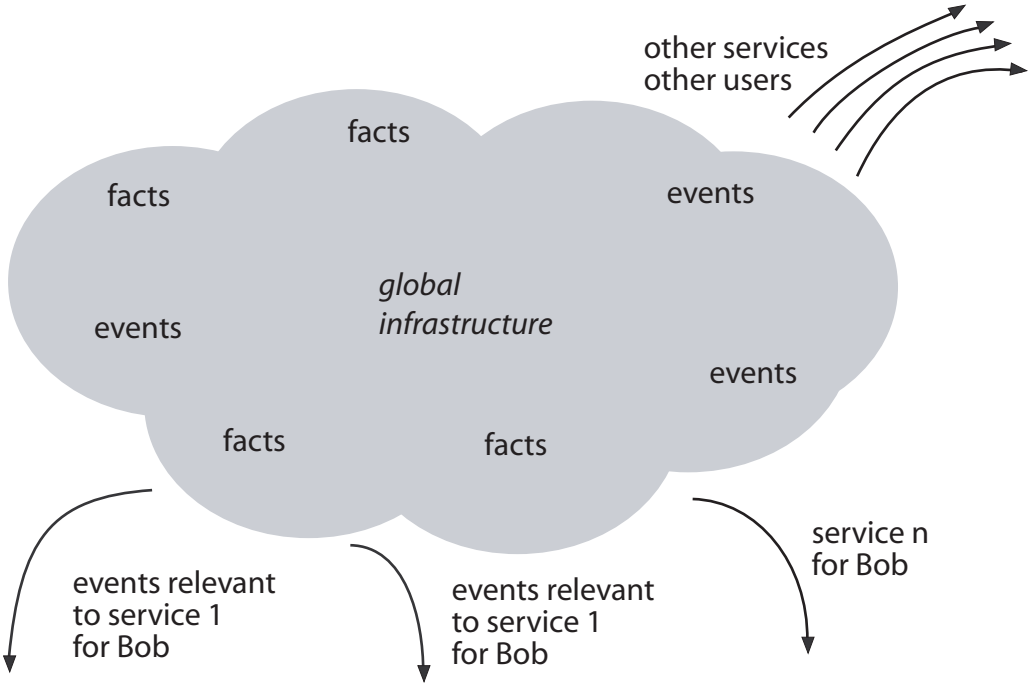


Abbildung 3.13: Die GLOSS-Infrastruktur nach [KDM+03].

Was für die *Ninja Paths* gilt, trifft auch für die Datenkanäle von Gloss zu. Es fehlt ihnen an Mächtigkeit der Datenmodellierung und die Vorstellung von Datenströmen von Kontextinformationen entspricht nicht dem Ausgangspunkt dieser Arbeit, nämlich Kontextinformationen in hochdynamischen Umgebungen stetig auf neuen Wegen zu besorgen, beziehungsweise dafür bereit zu stellen.

Die „Nexus Platform“

An der Universität Stuttgart beschäftigt sich ein Sonderforschungsbereich der Deutschen Forschungsgesellschaft mit der Entwicklung von Weltmodellen für kontextsensitive Systeme, insbesondere hinsichtlich der Fragen der Kommunikation wie auch des Informationsmanagements. Ebenso behandelt werden Verfahren zur Modellpräsentation sowie die Integration von Sensordaten. Nexus hat zum Ziel, alle möglichen Arten von kontextsensitiven Diensten zu unterstützen, indem sie ein gemeinsames, globales Kontextmodell erstellen, das von allen genutzt werden kann. Dabei haben Frank Dürr und andere [DHN+04] die *Nexus Platform* für kontextsensitive Dienste (siehe Abb. 3.14) vorgestellt, die lokale Kontextmodelle zu einer Föderation zusammenfasst.

Ein *context server* speichert ein lokales Kontextmodell. Er implementiert ein Interface, das einfache ortsbezogene Anfragen erlaubt und die Resultate in einem spezifizierten XML-Format zurück liefert. Der *context server* registriert sich in einem *Area Service Re-*

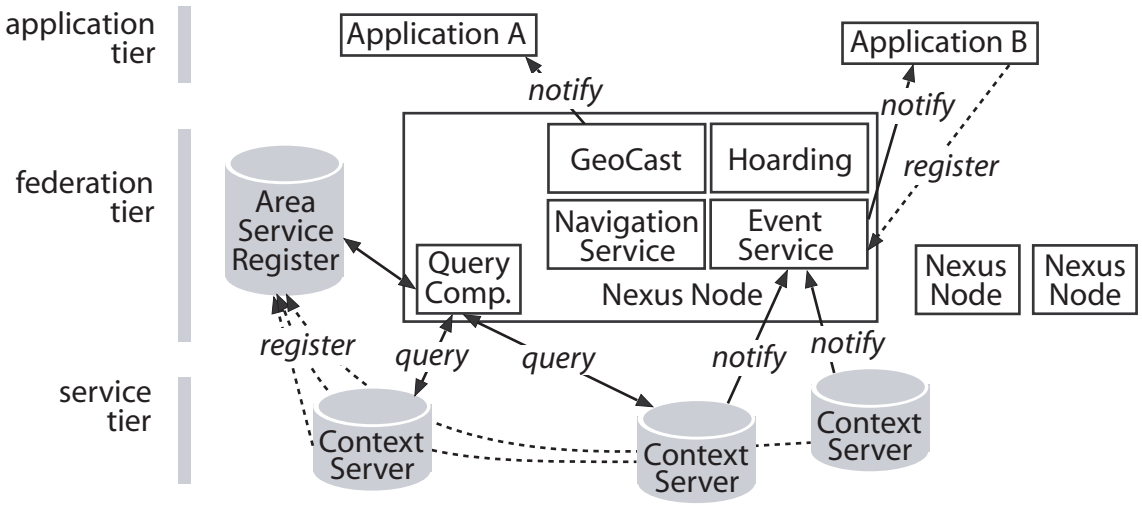


Abbildung 3.14: Die *Nexus Platform* für kontextsensitive Dienste.

gister mit Angaben darüber, welches Dienstgebiet er abdeckt und welche Objekttypen. Ein Knoten in der *federation tier* vermittelt zwischen Anwendungen und *context servers*. Er analysiert die Anfragen, bestimmt mit Hilfe des *Service Area Registers* die dafür zuständigen *context servers* und leitet die Anfrage an sie weiter. Die eingehenden Resultat kombiniert er zu einer einheitlichen Sicht und gibt diese zurück an die Anwendung.

Abgesehen von der Funktionalität, Anfragen nach Kontextinformationen zu erfüllen, unterstützt jeder *Nexus Node* spezialisierte Zusatzdienste, die eigene Schnittstellen besitzen und das föderierte Kontextmodell benutzen. Der *Event Service* in Abb. 3.14 überwacht räumliche Events, und kombiniert einfache Events zu komplexeren. Er erlaubt damit die Verarbeitung räumlicher Bedingung, wie dieser: „Zwei meiner Freunde treffen sich."

Nexus ist sehr von räumlichen Informationen und lokalen Modellen sowie Kontextquellen abhängig und ist damit nur beschränkt oder mit Zusatzaufwand tauglich für ubiquitäre Systeme, in denen Nutzer wie Geräte hochmobil sind, und sich erst im Augenblick der Dienstnutzung verknüpfen.

Architekturschema Ferschas

Alois Ferscha von der Universität Linz war der erste, der den Begriff *contextware* [Fer02] eingeführt hat. Mit der zuvor erwähnten *ContextWare*-Architektur der *Disappearing-Computer*-Initiative hat er somit nichts zu tun. In Analogie zum Begriff *middleware*, den Ferscha versteht als Software-Technologie, die dazu dient, zwischen Softwarekomponenten zu vermitteln, ist *contextware* für ihn der Kern einer Software-Technologie, die zwischen Diensten und dem Kontext ihrer Ausführung vermittelt: eine Brücke zwischen virtueller und realer Welt. Ferscha liefert auch eine „Architektur kontextsensitiver Anwendungen" [Fer03], erläutert von Volker Christian [Chr04]: Die erste Schicht, die „Kontextsensorik",

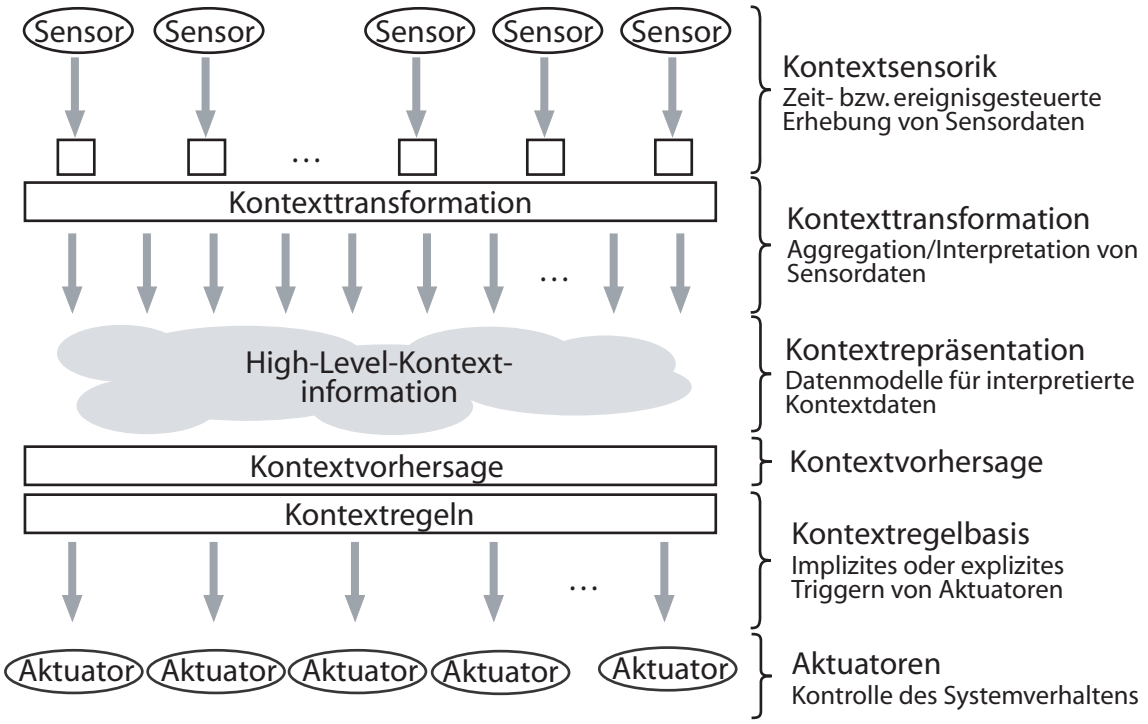


Abbildung 3.15: Ferschas „Architektur kontextsensitiver Anwendungen“ [Fer03].

ist verantwortlich dafür, so viele Informationen wie möglich über die reale Umwelt zu sammeln, in ein gemeinsames Datenformat umzuwandeln und an die nächste Schicht weiterzuleiten. Hier, in der Schicht „Kontexttransformation“ werden die eingehenden Daten gesammelt und in einer Weise aggregiert, dass die Schicht „Kontextrepräsentation“ eine einheitliche Abstraktion der realen Welt daraus generieren kann. Der Kontextinterpreter in dieser Schicht und die Schichten „Kontextvorhersage“ und „Kontextregeln“ benutzen diese Sicht, um anwendungsspezifische Transformationen darauf auszuführen und Steuerbefehle zu generieren, die dann von den Aktuatoren ausgeführt werden.

Dieses Modell berücksichtigt weder, wie Sensoren, Aktuatoren und die dazu gehörenden Regeln in hoch dynamischen Umgebungen zugeordnet werden können, noch sieht es für die Aktuatoren die Möglichkeit vor, selbst nach Kontextinformationen nachzufragen. Die Logik des kontextsensitiven Dienstes liegt somit allein in den Kontextregeln. In dieser Statik eignet es sich nicht für dynamische Umgebungen.

Das Projekt „Oresteia“

Ein weiteres Projekt der *Disappearing-Computer*-Initiative ist „Oresteia“, mit vollem Projektnamen: „*Modular Hybrid Artefacts with adaptive Functionality*“. Ziel des Projektes ist es, Architektur und kooperative Funktionen von Komponenten zu schaffen, die lokale Ent-

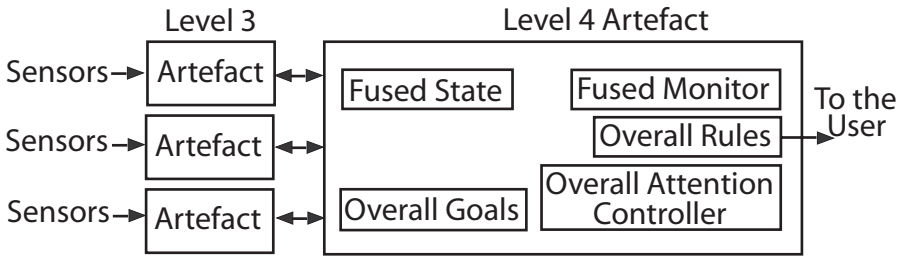


Abbildung 3.16: Level 3 und Level 4 von Oresteia nach [Tay03].

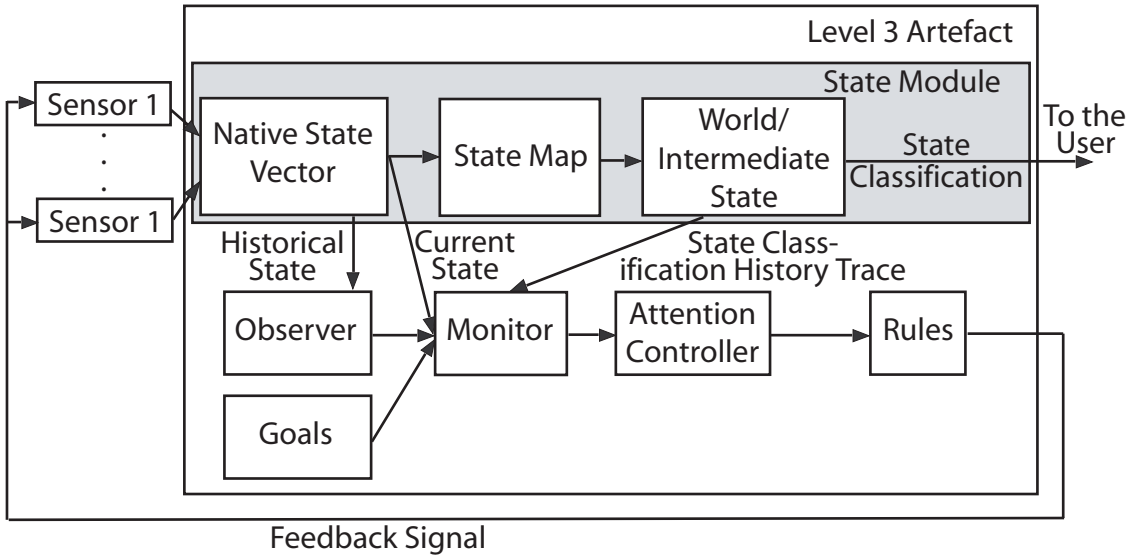


Abbildung 3.17: Zoom in ein Level-3-Artefakt von Oresteia.

scheidungsfähigkeiten besitzen und die Fähigkeit, ihr Verhalten zur Laufzeit anzupassen. Insbesondere sollten Komponenten geschaffen werden zur Analyse des Gesundheitszustandes und zur Gefahrenvermeidung.

John Taylor [Tay03] und Stathis Kasderidis beschreiben die Oresteia Architektur für Softwareagenten. Einem Agenten entspricht ein Level-4-Artefakt, dem die Hauptsteuerung und oberste Entscheidungsinstanz gehört. Es nimmt dazu alle Entscheidungen der darunter liegenden Level-3-Artefakte auf (siehe Abb. 3.16), die lokale Entscheidungsträger sind. Einem Level-3-Artefakt sind die Sensoren (Level 1) und Vorverarbeitungsstufen (Level 2) jeweils einer Messgröße zugeordnet.

Ein Level-3-Artefakt (siehe Abb. 3.17) ist verantwortlich für die Steuerung der Sensoren (bzw. deren Datenrate), die Aufrechterhaltung der Konsistenz der Messgröße (durch Hinzunahme weiterer Sensoren, sobald Verlässlichkeit ein Thema wird) und unerwartete Ereignisse innerhalb einer Messgröße zu behandeln. Für letztere Aufgabe besitzen die Ar-

tefakte Voraussagemodule (*Observers*), die entscheiden, ob ein Verhalten „wichtig“, „neu“ oder zu erwarten ist. Parallel dazu werden die Zustände des Benutzers, der Umgebung und des Agenten im lokalen *State Evaluation System* ausgewertet, um entsprechende Entscheidungen abzuleiten.

Ein Oresteia Agent besteht nach Kasderidis aus vier Untersystemen:

- Ein *Attention Control system*, das die verschiedenen Ereignissen sortiert, die Aufmerksamkeit verlangen,
- ein *State Evaluation system*, das Muster erkennt und damit die Grundlage schafft für Entscheidungen auf höherer Ebene,
- ein *Rules decision-making system*, das verantwortlich dafür ist, wie der Agent auf lokaler oder globaler Ebene reagiert und
- ein *Computational Model*, das die Informationsflüsse innerhalb eines Agenten definiert und deren Konsistenz überwacht.

Auch das Augenmerk dieser Architektur ist gerichtet auf die Logik, aus niederwertigen Sensordaten auf die jeweilige Situation zu schließen, nicht aber darauf, dass die zu Grunde liegenden Daten spontan von verschiedensten, fremden Quellen zusammengetragen werden können, wie es Fokus dieser Arbeit ist.

Das Projekt „Oxygen“

Im *Oxygen*-Forschungsbereich am Massachusetts Institute of Technology (MIT) geht man davon aus, dass in der Zukunft Rechner nicht nur überall und immer zur Verfügung stehen werden (wie die Luft zum Atmen, daher der Projektname), sondern auch auf natürliche, intuitive Art und Weise bedient werden können, durch menschliche Sprache und Gesten. Dafür sind verschiedene Konzepte und Technologien entwickelt worden, einmal mit dem Schwerpunkt Netze, einmal mit dem Schwerpunkt Software, dann auch mit dem Fokus auf die Geräte oder die Benutzer. Entstanden sind dabei zum Beispiel das *Intentional Naming System (INS)* [AWSBL99], das Indoor-Lokalisierungssystem *Cricket* [PCB00] oder die Java-Erweiterung MetaGlue [CPW+99] zur Unterstützung von Softwareagenten in intelligenten Räumen. All diese Techniken sind stark anwendungsbezogen. Ein übergeordnetes, generisches Konzept bilden sie nicht.

Die Strathclyde Context Infrastructure

An der Universität Strathclyde (Glasgow, Schottland) ist eine *Strathclyde Context Infrastructure* (SCI) entwickelt worden, die als Middleware zur Suche, Aggregation und Zustellung von Kontextinformationen dient.

Die SCI ist nach Richard Glassey und anderen [GSR+03] unterteilt in zwei Schichten (siehe Abb. 3.18). Die obere ist ein Overlay von teilweise verknüpften *range*-Knoten.

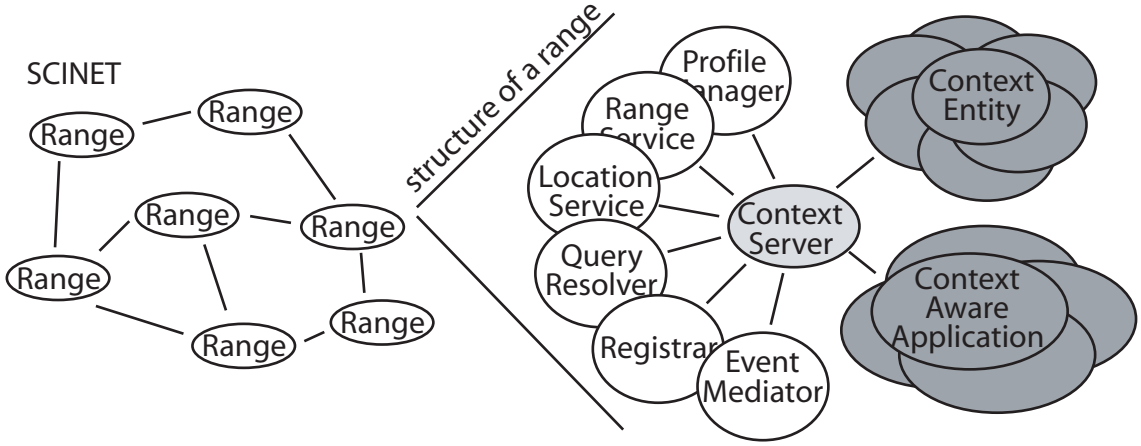


Abbildung 3.18: Die zwei Schichten von SCI.

Die untere betrifft den Inhalt jedes *range*-Knotens. Das Overlay-Netz kümmert sich um Interaktionen zwischen zwei oder mehr *range*-Knoten, um jeweils passende Kontextinformationen zur Verfügung zu stellen.

Ein *range*-Knoten kann beschrieben werden als logischer und als realer Bereich. Er kann durch einen realen Ort definiert werden (etwa der Bereich mehrerer zusammenhängender Räume) oder etwa durch den Bereich, den ein bestimmtes Netz abdeckt (zum Beispiel ein drahtloses Netz). Dadurch wollen Glassey et al. physische wie Software-Komponenten durch ein gemeinsames Modell abdecken.

Jeder *range*-Knoten besitzt einen eigenen, zentralen *Context Server*, von dem angenommen wird, dass er sicher und immer verfügbar ist, und der das Management der Kontextinformationen übernimmt. Über ihn erhalten die kontextsensitiven Dienste (*Context Aware Applications* in dieser Terminologie) Zugriff auf die Kontextinformationen. Eine *Context Entity* ist eine einfache Softwarekomponente, die eine Entität der SCI repräsentiert und eine Abstraktion für nicht rechenfähige Entitäten der SCI darstellt. Sie verwaltet ein Profil, das Metadaten über die Entität enthält und auch die Beschreibung der Dienste, die die Entität eventuell bereit stellt. *Context Entities* registrieren sich, wenn sie in den Bereich eines *range*-Knotens eintreten, und melden sich ab, sobald sie ihn verlassen.

Daneben gibt es *Context Utilities*, die dem *Context Server* spezialisierte Dienste zur Verfügung stellen. Zum Kern dieser Dienstgruppe gehören

- der *Range Service*, der Ankunft und Verlassen von Entitäten innerhalb des *range*-Knotens erkennt,
- der *Query Resolver*, der die Hilfsmittel bereit stellt, um eine eine Kontextanfrage in sinnvolle Bestandteile aufzuteilen und *Context Entities* zuzuordnen,
- der *Location Service*, der Ortsabhängigkeiten löst,

- der *Profile Manager*, der den Zugriff auf die Profile der *Context Entities* und deren Aktualisierung erlaubt,
- der *Event Mediator*, der die Verwaltung der Abonnements der Ereignisse zwischen *Context Entities* und *Context Aware Applications* übernimmt
- und der *Registrar*, der eine korrekte Sicht auf alle Entitäten innerhalb des *range*-Knotens verwaltet.

Auch diese Infrastruktur geht von sicher verfügbaren und zuvor bekannten Kontextquellen aus und erfüllt damit nicht die Anforderungen, die diese Arbeit an ubiquitäre, kontextsensitive Umgebungen stellt.

Das Projekt „WearNET“

Am „Wearable Computer Lab“ ist WearNET entwickelt und implementiert worden, ein verteiltes Multisensor-System für kontextsensitive Kleidung [LJS+02]. Die verschiedenen Kontextschichten, die dessen Architektur vorsieht, zeigt Abb. 3.19. Lukowicz und andere schreiben, dass die Entwicklung eines tragbaren Kontexterkennungssystems ein Trade-off sein muss zwischen der flexiblen Einsatzmöglichkeit auf der einen und der Effizienz für eine eng eingegrenzte Aufgabe auf der anderen. Als Kompromiss haben sie ihre Architektur ausgerichtet auf eine einzelne Anwendung und eine generische Gruppe von Sensoren. In der vermittelnden *Componente Layer* definieren sie vier Informationsarten und spezifizieren die jeweilige Sensorenkombination, die ihnen jeweils passend schien, um die Informationen zu liefern. Diese vier Informationstypen sind

- *Extended Location* (zum Beispiel „im Büro“, „in einer Einkaufsstraße“, „in einem Aufzug“),
- *Environment State* (zum Beispiel „Cocktail Party“, „Beamer-Vorführung“, „gedimmtes Licht“, „offenes Fenster“),
- *User Activity* (zum Beispiel „sitzend“, „gehend“, „trinkend“, „essend“, „Tür öffnend“) und
- *User State* (zum Beispiel „physisch erschöpft“, „im Stress“, „entspannt“).

Die entsprechenden Sensormodule bei WearNET dafür sind das *Navigation Module*, mit einem inertialen Navigationssystem (dient der Erfassung der Position und der Orientierung bewegter Objekte) aus GPS auf der einen und Sensoren für Beschleunigung, Gyroskop und Kompass auf der anderen Seite. Zusätzlich besitzt das Modul einen Prozessor zur Berechnung der Verfolgung der Ortsänderungen.

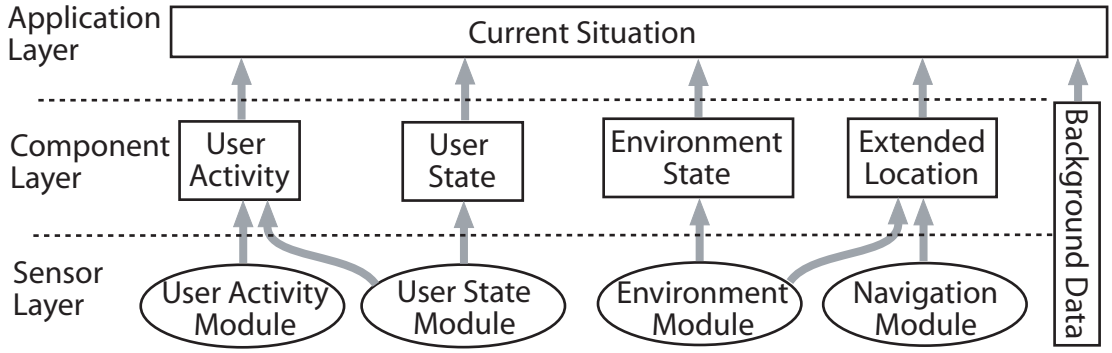


Abbildung 3.19: Die WearNET-Schichten nach [LJS+02]

Die Sensoren im *Environment Module* messen Infrarot-, Ultraviolett- und sichtbares Licht. Außerdem die Temperatur, Luftdruck, Luftfeuchtigkeit, Geräuschpegel und das Magnetfeld. Ein Prozessor übernimmt die Analog-Digital-Wandlung, die Sensorsteuerung und eine einfaches Vorverarbeiten der Daten.

Das *User Activity Module* ist ein mehrstufiges Netz von Bewegungssensoren, deren Hierarchie auf der Anatomie des Körpers beruht. Jede Gliedmaße, der Rumpf und der Kopf haben jeweils eigene Unternetze mit einem Bussystem und einer eigenen Steuereinheit. Die Steuereinheiten wiederum besitzen einen eigenen Bus und unterliegen einer zentralen Hauptsteuereinheit.

Das *User State Module* kombiniert einen GSR-Sensor (steht für „Galvanic Skin Rezeption“) zur Messung des Hautwiderstandes mit Sensoren zur Messung des Pulses und des Sauerstoffgehaltes des Blutes. Auch hier sorgt ein einfacher Prozessor für die Analog-Digital-Wandlung, und einfache Vorverarbeitung.

Mit der Einschränkung der Sensortypen und des Verwendungszwecks ist auch WearNET nur als proprietäre Lösung im Sinne einer nicht-generischen Lösung zu betrachten, und damit nicht tauglich für generische Verwendungszwecke und den uneingeschränkten Einsatz in offenen Systemen.

University Queensland: „Software Infrastructure“

Karen Henricksen und Jadwiga Indulska [HI04] von der Universität Queensland haben das Konzept einer Software- Infrastruktur vorgeschlagen, die zum einen der Umsetzung zweier Programmmechanismen namens „Branching“ und „Triggering“ dient, als auch das Management von Kontextinformationen unterstützt. Teilweise haben sie diese Infrastruktur als Proof-of-Concept auch implementiert.

Die Infrastruktur umfasst mehrere lose verbundene Schichten. Die *Context gathering layer* verarbeitet reine Sensordaten und interpretiert oder aggregiert sie nach Bedarf. Das Ergebnis sind Informationen höherer Abstraktion. Die *Context reception layer* leitet zum einen Anfragen der darüber liegenden *Context management layer* an die entsprechenden

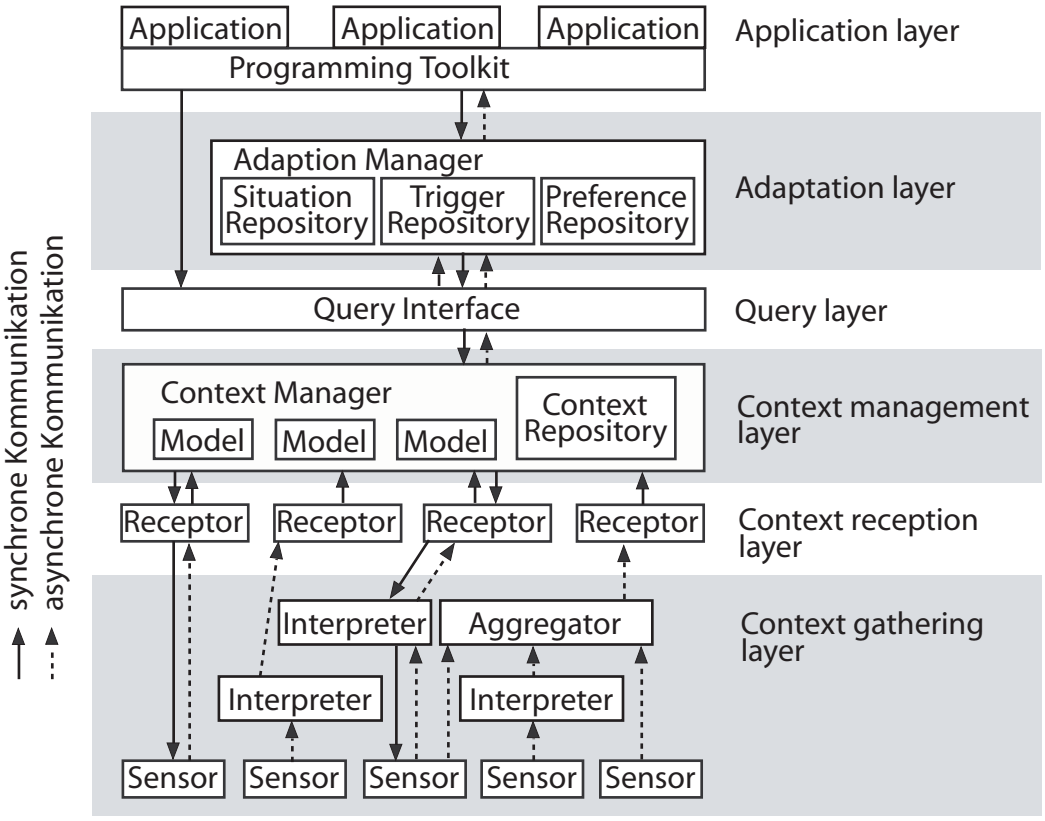


Abbildung 3.20: Infrastruktur zur Unterstützung von Kontextinformationen nach [HI04].

Komponenten der darunter liegenden Schicht weiter. Zum anderen übersetzt es die Daten der *Context gathering layer* in die Repräsentation, mit der die darüber liegende Schicht arbeitet.

Die *Context management layer* verwaltet Kontextmodelle und -instanzen und sollte nach Ansicht von Henricksen und Indulska verteilt sein. Typischerweise solle jede Applikation ihr eigenes Modell besitzen, das zwischen verwandten Anwendungen allerdings geteilt werden könne. Die *Query Layer* bietet eine Anfrageschnittstelle an die *Context management layer*, die sowohl synchrone Anfragen als auch asynchrone Benachrichtigungen unterstützt.

Die *Context adaption layer* verwaltet gemeinsame Repositories mit Definitionen von Situationen, Präferenzen und Triggern. Alle Anwendungen, die zu einem bestimmten Nutzer gehören, oder alle Anwendungen, die auf einem einzelnen Gerät ausgeführt werden, bilden typischerweise jeweils eine Gruppe, die sich ein Repository teilt.

Die *Application layer* bietet schließlich ein *toolkit* (einen Bausatz) für die grundlegenden Funktionen der bereits erwähnten Programmiermechanismen: *branching* bietet generische Möglichkeiten, abhängig von Kontextzuständen Aktionen auszuwählen und aus-

zuführen, ohne „if“ oder „case“-Anweisungen auf Ebene des Programmcodes verwenden zu müssen. *triggering* dagegen unterstützt ähnlich dem ECA-Modell (*„event-condition-action*“) in aktiven Datenbanken die Möglichkeit, bei Zustandsübergängen des Kontexts bestimmte Aktionen auszulösen.

Mit dem komplexen *object role model*, das Henricksen und andere zur Modellierung von Kontextinformationen entwickelt haben, beschäftigt sich Abschnitt 5.4 ausführlicher.

Ähnlich wie Ferschas Schichtenmodell ist auch das von Henricksen und Indulska bestimmt durch die notwendigen funktionalen Schritte zwischen der Erzeugung der Kontextinformation (hier in der *context gathering layer*) und der Verwendung im kontextsensitiven Dienst (hier in der *„Application layer*“), wie sie auch schon die Kontextwertschöpfungskette von Hegering et al. vorsieht. Der große (noch fehlende) nächste Schritt ist die Ausgestaltung der Schichten oder Schritte und der Schnittstellenprotokolle der vertikalen und horizontalen (bei Verteilung von Komponenten derselben Ebene) Schnitte.

Die Architektur: „one.World“

Robert Grimm von der Universität New York und andere [GDL+04] haben drei „einzigartige Anforderungen“ des *Pervasive Computing* aufgestellt. Demnach müssten pervasive Systeme

1. „Änderungen des Kontexts offenlegen, statt eine Verteilung zu verschatten“,
2. „spontane Vernetzung unterstützen“ und
3. „den gegenseitigen Austausch von Informationen zwischen Anwendungen als Regelfall anerkennen“.

Die von ihnen vorgestellte Architektur *one.World* (siehe Abb. 3.21) soll allen drei Anforderungen gerecht werden. Die *Foundation* und *System Services* sind Teil des Kernels während *Libraries, system utilities* und *Applications* im Bereich des Nutzers laufen. Die *Virtual Machine*, also etwa die Java Virtual Machine oder die Common Language Runtime von Microsoft ist der erste von vier grundlegenden *Foundation services*. Ein zweiter, *Tuples* definiert ein gemeinsames Datenmodell für alle Anwendungen, um den Datenaustausch zu erleichtern. Tupel sind selbst beschreibende Einträge mit Name-Wert-Paaren, die verschachtelt weitere Einträge enthalten können. Kontextuelle Veränderungen werden Anwendungen über asynchrone Events mitgeteilt, so dass sie sich daraufhin anpassen können. Dafür steht *Asnychronous Events*. Schließlich gibt es noch mit der *Environments*-Ebene einen zentralen Mechanismus, um Anwendungen zusammen zu bauen und zu strukturieren. Ein *Environment* dient als Container für Tupel, Komponenten von Anwendungen und rekursiv weiteren *Environments*. Darauf aufsetzend gibt es eine Menge von *System Services*, die als gemeinsame Bausteine für Anwendungen dienen. Dazu gehören ein Dienst, der Events weiterleitet und eine *Query Engine*, die Tupel filtert und dazu Konstante mit

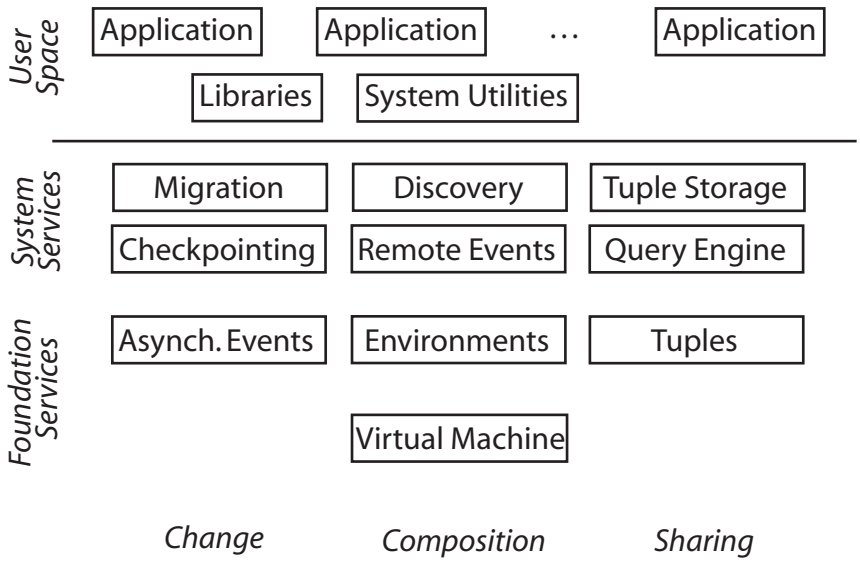


Abbildung 3.21: Übersicht über die *one.World*-Architektur.

dem Bezeichner oder dem Wert eine Feldes vergleichen kann, und Negation sowie die Bildung von Teil- und Vereinigungsmengen unterstützt.

Außerhalb des Kernels, im Nutzerbereich bietet *one.World* eine zusätzliche Programmbibliothek für pervasive Anwendungen. Diese umfasst Funktionalität zum Bau einer Benutzerschnittstelle und zur zeitgesteuerten Ausführung von *event handlers*.

Der Ansatz von *one.World* legt seinen Schwerpunkt offenbar eher auf die Architektur innerhalb eines Anwendungssystems als darauf, über Domänen- und Systemgrenzen hinweg den Austausch von Kontextinformationen zu gestalten. Auch das Datenmodell ist nicht ausdrucksstark genug für diese Verwendung.

Die Architektur „CoBrA“

Harry Lik Chen und andere von der Universität Maryland haben die *Context Broker Architecture* (CoBrA) vorgestellt. Ihr Ansatz, kontextsensitive Systeme in so genannten *smart spaces* zu unterstützen, ist agentenbasiert. Der *context broker* soll auf einem leistungsfähigen, stationären Rechner laufen und eine Reihe von Aufgaben erledigen:

- Er soll ein zentralisiertes Kontextmodell liefern, das alle Geräte, Dienste und Agenten in dieser Umgebung teilen können,
- Kontextinformationen von Quellen beschaffen, die von Geräten mit weniger Leistung und Funktionalität nicht erreicht werden können,
- Kontextinformationen ableiten, die nicht direkt von den Sensoren geliefert werden können,

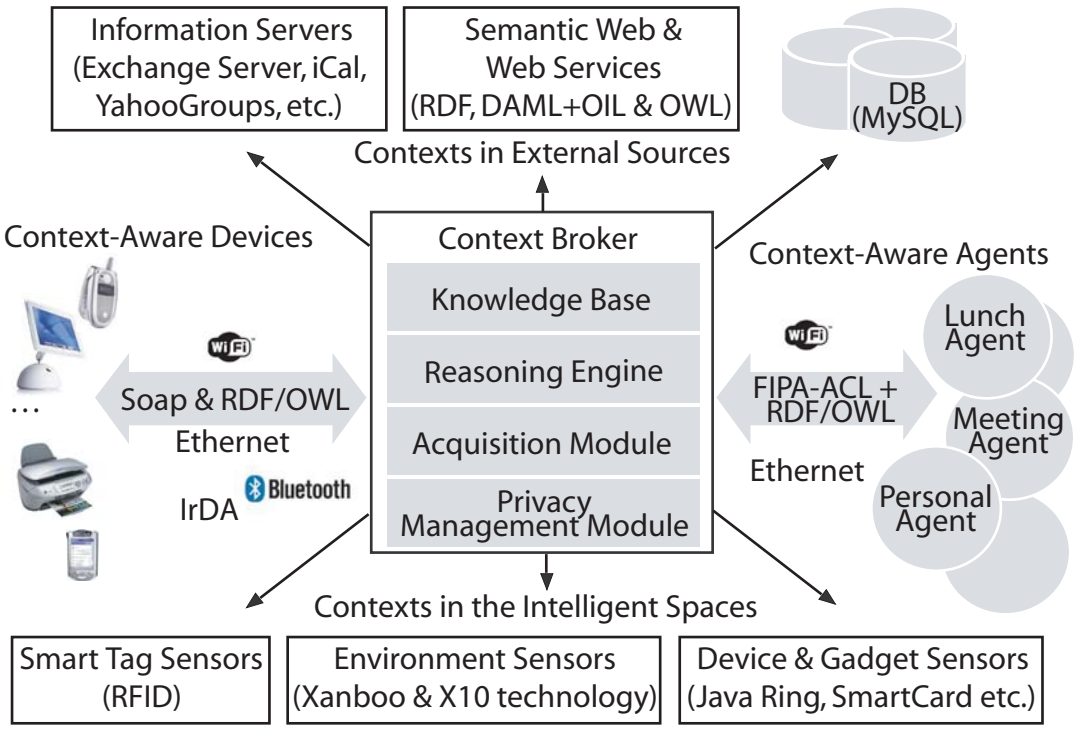


Abbildung 3.22: Mit dieser Übersicht will Chen die Architektur von CoBrA verdeutlichen.

- Inkonsistenzen im gesammelten Kontextwissen entdecken und auflösen
- sowie den Datenschutz der Benutzer entlang der von diesen aufgestellten Regeln sicher stellen.

Das einheitliche Kontextmodell ist dabei für CoBrA von zentraler Bedeutung. Mehr zu Soupa, einer Sammlung von OWL-basierten Ontologien, im Abschnitt 5.4.

Der Ansatz von CoBrA lohnt insofern eine nähere Betrachtung, als dass das beschriebene Szenario kontextsensitiver Systeme dem in dieser Arbeit vorgestellten Szenario ähnelt und die darin geforderte Komponente eines Brokers ebenfalls in seiner Funktionalität eng mit der Komponente des Kontextvermittlers verwandt ist.

Die Schwächen offenbaren sich jedoch in seiner Ausgestaltung. Die Mängel des Kontextmodells, das für CoBrA von zentraler Bedeutung ist, aber gar keine einheitliche Kontextmodellierung birgt, werden in Abschnitt 5.4 erläutert. Auch was die Beschreibung der beteiligten Dienste angeht oder den lediglich skizzierten Einsatz von Policies fehlt es dem Ansatz an Ausgestaltung und formaler Tiefe. Er vertraut darauf, dass sich Interoperabilität *bottom-up* durch den steten Ausbau kontextsensitiver Systeme von selbst herausmendelt und verkennt damit sowohl die normative Kraft als auch die prinzipielle Notwendigkeit von Standards.

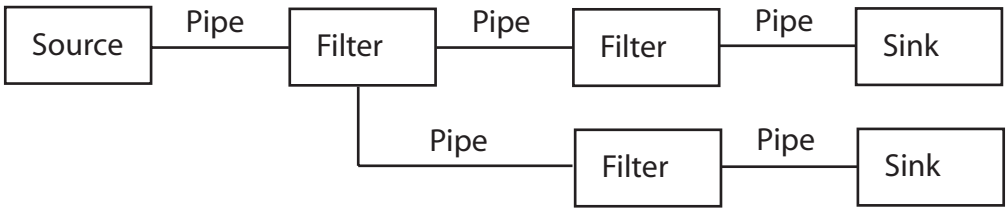


Abbildung 3.23: Solar arbeitet im Stil einer *filter-and-pipe* Architektur.

Das Context Fusion Network „Solar“

Guanling Chen hat mit David Kotz und anderen am Dartmouth College Hanover (USA) das Konzept eines *Context Fusion Networks* entwickelt und als „Solar“ implementiert. Grundannahme war, dass viele Anwendungen die gleichen Datenquellen benutzen oder es zumindest eine große Überlappung gibt, und die Daten jeweils ähnliche Verarbeitungsschritte durchlaufen.

Mit Solar lassen sich die Wege von Datenströmen im Stil einer *filter-and-pipe* Architektur (siehe Abb. 3.23) komponieren und wiederverwenden. Ein „*filter*“ ist eine unabhängige Komponente, die auf einer Menge an Eingaben eine bestimmte Funktionalität anbietet und eine Menge an Ausgaben erzeugt. Eine „*pipe*“ dient als Kanal für die Datenströme von der Ausgabe eines *filters* zur Eingabe eines anderen. So wird der Datenstrom ausgehend von einer Quelle über eine Reihung von *filters* und *pipes* bis zur Senke geleitet.

Im *context fusion model* von Solar heißt ein *filter* nun „*operator*“ und die *pipe* wird zum „*channel*“. Abbildung 3.24 zeigt einen *operator graph* von Solar. Ein Versus-Sensor, der die Position einer Menge von *ActiveBadges* überwacht, ist eine der beiden Datenquellen, eine Kalenderapplikation die andere. Der *Locator-operator* versendet Events, sobald sich die Position eines *badges* ändert. Die *ActiveMap* zeigt die Positionen aller *badges* an, während die *Reminder*-Anwendung über den *Filter-operator* nur die Positionsänderung für eine bestimmte Person zugesandt bekommt. Gemeinsam mit den Daten aus dem Kalender der Person kann so der *Reminder* entscheiden, wann es notwendig wird, den Benutzer an einen Termin zu erinnern.

Jeder Sensor registriert seinen Namen im *directory service* von Solar. Optional kann das auch jeder instanziierte *operator* tun, da er logisch gesehen ebenfalls eine Datenquelle darstellt. Anwendungen können anhand der Ergebnisse von *name queries* dann geeignete Datenquellen wählen.

Die Rechner auf denen die verschiedenen *operators* laufen, heißen bei Solar *planets*. Diese bilden zusammen ein Service-Overlay-Netz basierend auf dem Peer-to-Peer-Protokoll „*Pastry*“ [RD01]. Zur Adressierung des Datenaustauschs zwischen den Rechnern erhält jeder von ihnen einen eindeutigen Identifikator in einer verteilten Hash-Tabelle. Alternativ ist eine Adressierung direkt über die TCP/IP-Protokolle möglich. Dieser Ansatz ähnelt den Kontextkompositionsgraphen, die der Autor dieser Arbeit entwickelt hat und

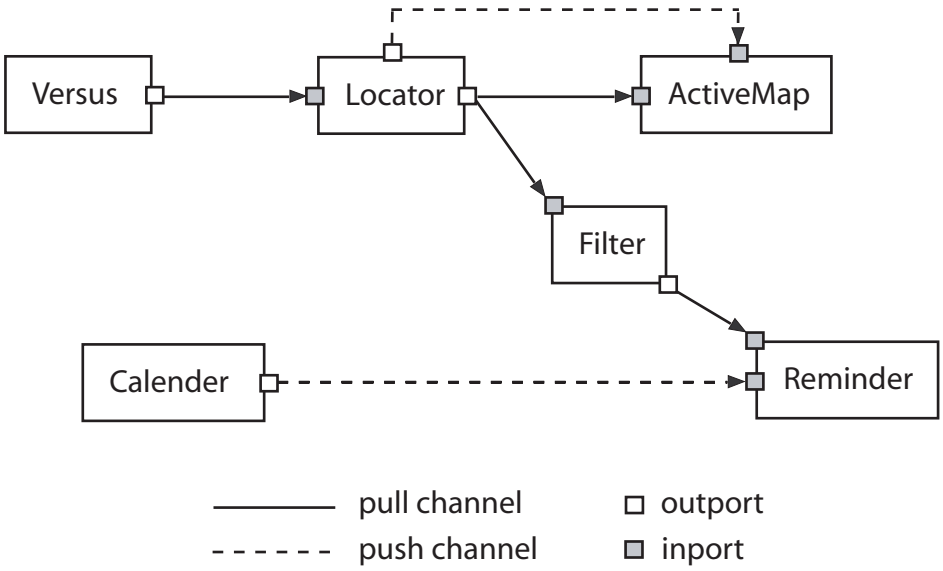


Abbildung 3.24: Ein Beispiel *operator*-Graph für Solar.

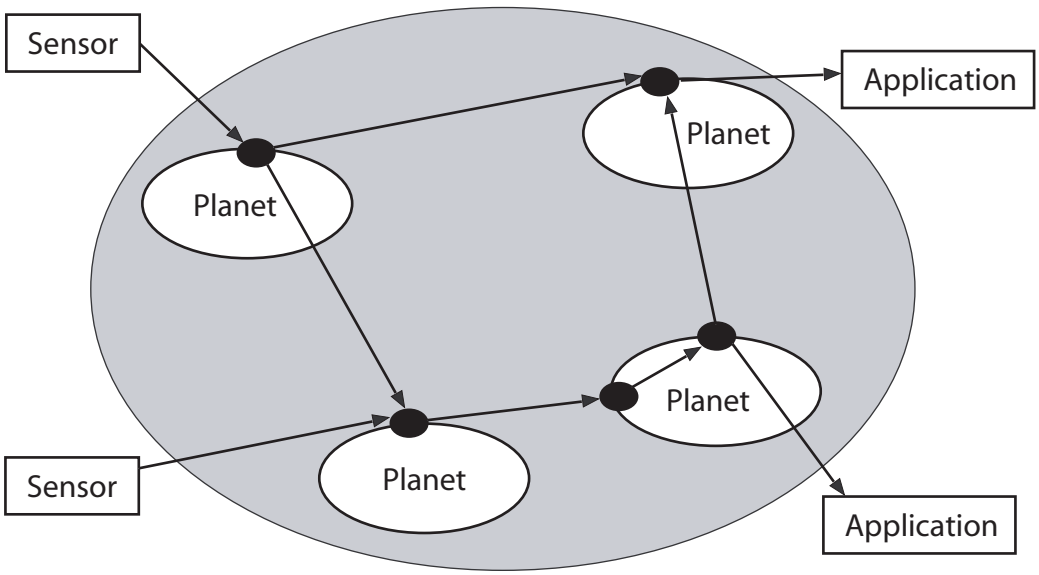


Abbildung 3.25: Die *Planet*-Komponenten in Solar.

in Abschnitt 7.1.1 näher beschreibt. Durch die persistente *Overlay*-Netzstruktur von Solar ist die Bindung der Graphknoten an eine existierende Konfiguration von Sensoren weit fester als im Kontextkompositionsgraphen-Ansatz. Solar basiert auf der Vorstellung, dass die Wege von den Sensorquellen zu den kontextsensitiven Diensten über einen größeren Zeitraum als Lieferkanäle bestehen bleiben. Dagegen adressiert die vorliegende Arbeit das Problem, dass sich Kontextquellen (beziehungsweise die sie kapselnden Kontextinformationsdienste) und kontextsensitive Dienste in hoch dynamischen Umgebungen stets neu suchen und finden müssen.

Die Service-Oriented Context-Aware Middleware „Socam“

Tao Gu und andere von der National University in Singapur haben die Socam-Architektur entworfen [GPZ05]. Kern dieser infrastrukturbasierten, dienstorientierten Middleware ist ein formales OWL-Kontextmodell als Informationsmodell der unabhängigen Komponenten (mehr zur Kontextmodellierung von Socam in Abschnitt 5.4). Dazu gehören ein Dienstverzeichnis namens *service location service*, in dem sich die *context providers* und der *context interpreter* registrieren, damit sie von Benutzern und anderen Applikationen gefunden werden können. Dabei kennt Socam zum einen *external context providers*, die Kontextinformationen von externen Quellen wie Wetter- und Ortungsdiensten beziehen. Auf der anderen Seite gibt es die *internal context providers*, die ihre Daten direkt von physischen Sensoren innerhalb der eigenen *Ubiquitous-Computing*-Subdomäne erhalten. Dabei sollen *context providers* abstrahieren von den systemnahen Daten, die die Sensoren liefern, und Informationen auf höherem Abstraktionsgrad nach außen bereitstellen. Gu versteht darunter zum Beispiel etwa, Videodaten maschinell auszuwerten um daraus die Situation auf höherem Niveau beschreiben zu können. Etwa „eine Person liegt im Bett“, „ein Eindringling ist entdeckt worden“ oder „eine Gruppe von Personen sitzt am Tisch im Besprechungsraum.“

Zentralen Raum nimmt bei Socam der *context interpreter* ein, bestehend aus einem Speicher für Kontextwissen und einer Komponente für logische Entscheidungen, dem *reasoner*. Interessant an diesem Ansatz ist, dass der *context interpreter* nicht nur als Hilfskomponente benutzt wird, um Entscheidungen für andere Dienste zu treffen, sondern selbst als *context provider* auftritt.

Socam und die in dieser Arbeit vorgestellten Ansätze ergänzen sich, da sie von sehr ähnlicher Problemstellung ausgehen und ähnliche Vorstellung der notwendigen Komponenten haben. Warum die Kontextmodellierung von Socam nicht mächtig genug ist, wird in Abschnitt 5.4 erläutert. Die Schlussfolgerungsmechanismen in Abschnitt 7.2.5 erweitern die Ansätze von Tao Gu et al. zum Einsatz von Bayes-Netzen beträchtlich.

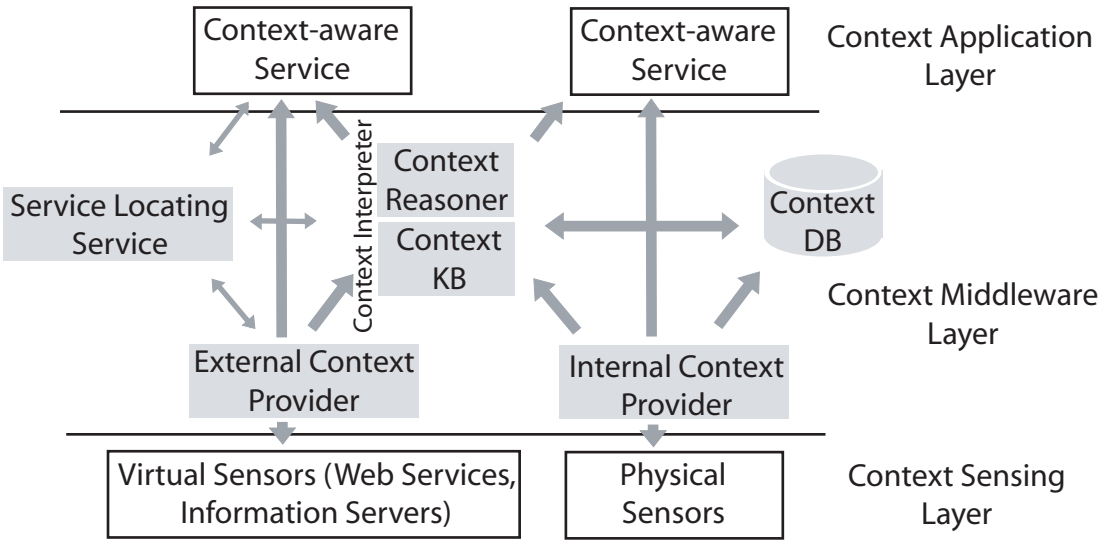


Abbildung 3.26: Das Middleware-Konzept der Socam Architektur.

3.5 Zusammenfassung

Um die Hauptkritikpunkte der eben vorgestellten verwandten Arbeiten zur Bereitstellung von Kontextinformationen zusammenzufassen: Die meisten Ansätze gehen von Umgebungen aus, in denen die Konfiguration fest oder nur wenig dynamisch ist, sie sind nicht generisch und damit auch nicht interoperabel genug, oder ihnen fehlt die detailliertere Ausgestaltung der Protokolle und Modellierungsstandards zwischen den jeweiligen Komponenten.

Damit ist in diesem Kapitel an Beispielen und einem abstrakten Szenario die Ausgangslage dieser Arbeit worden. Weiter wurde aufgezeigt, dass bestehende Arbeiten dieser Ausgangslage nicht gerecht werden und es wurde umrissen, welchen Anteil zur Lösung diese Arbeit beibringen will.

4 Ontologien und Sprachen zur Repräsentation von Wissen

> Ich möchte dem Leser zu wohlwollender Überlegung eine Doktrin empfehlen, die ihm vermutlich unerhört paradox und umstürzlerisch erscheinen wird. Die Doktrin ist die folgende: Es ist nicht wünschenswert, an eine Behauptung zu glauben, wenn kein Grund vorliegt, sie für wahr zu halten.
>
> *(Bertrand Russell)*

Rein formal unterscheiden sich Kontextinformationen von anderen Informationen nicht per se, sondern durch ihren Gebrauch (Vgl. Kontextdefinition in Abschnitt 2.1.2 und Terry Winograd in [wino01]: „*Features of the world become context through their use*“). Trotzdem lassen sich Merkmale angeben, die für Kontextinformationen in der Regel spezifisch sind: So sind Kontextinformationen überwiegend kurzlebige und schnell veränderliche Informationen. Sie sind abgesehen von den Grenzen der Privatsphäre im Allgemeinen leicht zugänglich (Schließlich sind sie für den Austausch über System- und Domänengrenzen bestimmt) und beantworten lediglich einfache Fragen. Komplexe Fragen, etwa die nach der Kriegsschuld im Zweiten Weltkrieg oder die nach dem Wetter in vier Monaten taugen nicht als Kontextanfragen.

Es ist seit Jahrzehnten Anliegen der Forschung zur Künstlichen Intelligenz, Repräsentationen von Wissen zu schaffen, die diese Möglichkeiten bieten [Str91]. Sie hat damit wiederum andere Forschungszweige mit ähnlich gelagerten Problemen befruchtet. Im Folgenden wird dazu unter anderem von der Informationsmodellierung und -verarbeitung im *Semantic Web* die Rede sein. Doch zunächst wird der Begriff der Ontologie geklärt und es werden Logiksprachen zur Wissensrepräsentation vorgestellt.

4.1 Was ist eine „Ontologie“?

Der Begriff der Ontologie wird mit stark abweichenden Bedeutungen gebraucht. Allgemein ist eine Ontologie ein Werkzeug, um sich über die Dinge, über die man kommuniziert, ein gemeinsames Verständnis zu verschaffen. Der Begriff ist aus der Philosophie

entlehnt worden. Im Folgenden wird dieser Ursprung erläutert, es wird die Verwendung des Begriffs in der Informatik diskutiert, und es wird eine Definition für diese Arbeit gegeben.

Historischer Ursprung des Ontologiebegriffs in der Philosophie

Die Ontologie („on“ ist griechisch für „sein“) ist in der Philosophie die Wissenschaft vom Sein und vom Seienden im Allgemeinen. Sie geht zurück auf den Wesensbegriff von Aristoteles, und seine Bücher über „Metaphysik“ werden als erste Ausarbeitung einer Ontologie in der Philosophiegeschichte genannt [KBW99]. Mit dem „Wesen“ eines Dinges sind demnach diejenigen seiner Eigenschaften gemeint, die sich nicht verändern können, ohne dass das Ding seine Identität einbüßt, wie Bertrand Russell [Rus04] ausführt. Russell lehnt den Begriff als „verworren“ ab, bescheinigt ihm aber, dass er in der nacharistotelischen Philosophie bis zur Neuzeit nicht wegzudenken ist. Erstmals nachgewiesen ist das Wort bisher in dem 1613 erschienenen „Lexicon Philosophicum“ des deutschen Logikers und Lexikographen Rudolph Goclenius („Göckel“) senior aus Marburg. Der deutsche Mathematiker und Philosoph Christian Wolff hat 1730 in seiner „Philosophia prima sive ontologia“ die Ontologie als „Erste Philosophie“ bezeichnet und als Disziplin klassifiziert, die sich mit dem befasst, „was den Gegenständen der Realphilosophie, nämlich Geist und Körper, gemeinsam sei“ (nach [Bra95]).

Musste das Ziel der Ontologie gemäß dieser Überlegungen eine einheitliche, allgemein gültige Systematik sein, da das „Sein“ etwas Unveränderliches ist, gilt dies für modernere Ansätze nicht mehr. Der Metaphysikkritiker Immanuel Kant etwa lehnte die traditionelle Ontologie als „anmaßend“ (nach [Hes02]) ab, da das vom Menschen Erfahrbare nichts über das vom Menschen unabhängig existierende „Sein“ aussagen könne. Für ihn galt:

> „Die Ontologie ist diejenige Wissenschaft (als Teil der Metaphysik), welche ein System aller Verstandesbegriffe und Grundsätze, aber nur so fern sie auf Gegenstände gehen, welche den Sinnen gegeben, und also durch Erfahrung belegt werden können, ausmacht. Sie berührt nicht das Übersinnliche, welches doch der Endzweck der Metaphysik ist, gehört also zu dieser nur als Propädeutik, als die Halle, oder der Vorhof der eigentlichen Metaphysik, und wird Transcendental-Philosophie genannt, weil sie die Bedingungen und ersten Elemente aller unserer Erkenntnis a priori enthält. In ihr ist seit Aristoteles' Zeiten nicht viel Fortschreitens gewesen.“ [Kan05]

Entsprechend sind die Vertreter der „Neuen Ontologie“ oder „Neuen Metaphysik“ auch Empiriker, die keine von der Erfahrung getrennte Spekulation wollen [Möl05]. Zu ihnen zählt auch der 1950 gestorbene deutsche Philosoph Nicolai Hartmann, der den kritischen Rationalismus begründete. Er schuf eine alle Probleme der Philosophie umfassende Ontologie und entwickelte darin einen Schichtenaufbau des Seins.

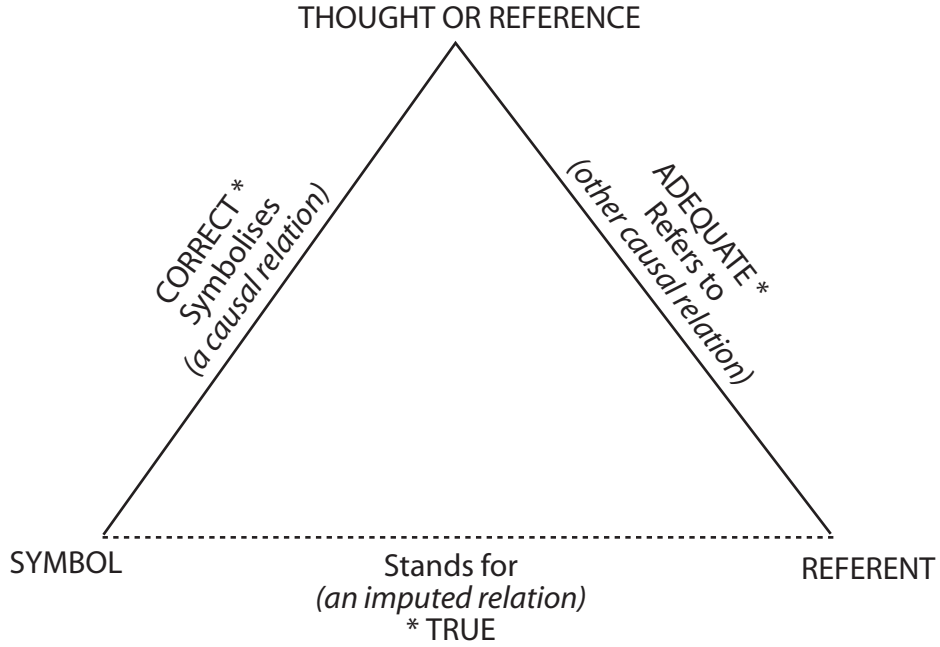


Abbildung 4.1: Das semiotische Dreieck von Charles K. Ogden und A. Richards [OR23].

Nicola Guarino [Gua98] beschreibt *eine* Ontologie (im Gegensatz zu *der* Ontologie als Disziplin) in der Philosophie als ein spezifisches System von Kategorien für eine bestimmte Weltsicht. Damit ist dieses unabhängig von der Sprache, in der es beschrieben wird.

Ontologien in der Informatik

Eine Ontologie in der Wissenschaft von der Künstlichen Intelligenz bezieht sich nach der Erläuterung von Nicola Guarino [Gua98] dagegen auf ein technisches Konstrukt (zum Beispiel einen Katalog, eine Sammlung von Thesauri oder eine Logik-basierte Wissensbasis), errichtet aus einem bestimmten Vokabular (formal oder nicht formal) und benutzt, um eine gewisse Realität (Gegenstände und Sachverhalte) zu beschreiben. Dazu kommt eine Menge von Prämissen über die beabsichtigte Bedeutung der Worte des Vokabulars.

Um die Doppeldeutigkeit des Begriffs „Ontologie“ zu umgehen, schlägt Guarino vor, innerhalb der Künstlichen Intelligenz das Wort „Konzeptualisierung“ zu gebrauchen. Der Begriff „Konzept“ ist dabei einer von leider vielen Synonymen für einen elementaren Begriff der Philosophie und der Sprachforschung, der zurück geht auf die Zeichenlehre von Aristoteles:

> „Nun sind die Äußerungen unserer Stimme ein Symbol (semiota) für das, was unserer Seele widerfährt, und das, was wir schriftlich äußern, (ist wiederum ein Symbol) für die Äußerungen unserer Stimme. Und wie nicht alle mit den-

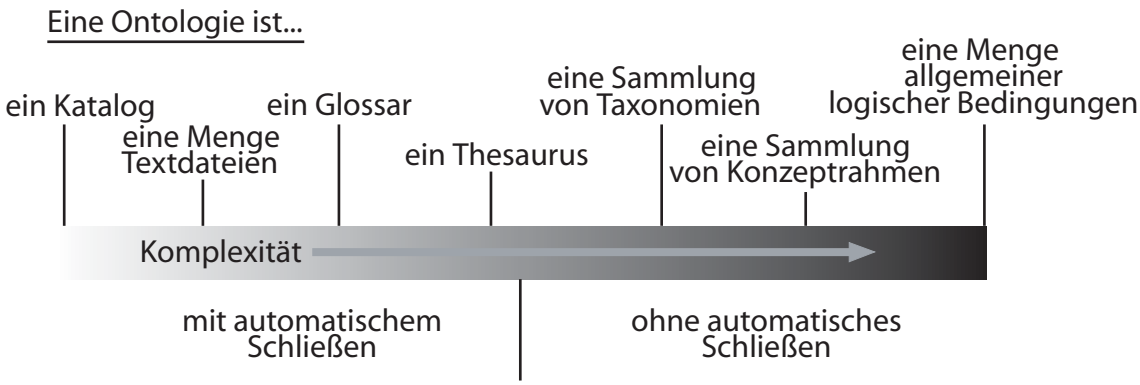


Abbildung 4.2: Ausprägungen von Ontologien nach Smith und Welty [SW01].

> selben Buchstaben schreiben, so sprechen auch nicht alle dieselbe Sprache. Die seelischen Widerfahrnisse aber, für welches dieses an erster Stelle ein Zeichen ist, sind bei allen dieselben; und überdies sind auch schon die Dinge (pragmata), von denen diese Abbildungen (homoiomata) sind, für alle dieselben.“ (Aus „Peri hermeneias“ nach [Mer02])

Neu formuliert haben diese Zeichenlehre 1923 Charles Ogden und K. Richards mit ihrem Semiotischen Dreieck (*meaning triangle*, siehe Abb. 4.1). Demnach haben wir eine gedankliche Vorstellung (*reference* oder „Konzept“) von einer realen Sache (*referent* oder „Objekt“). Der Name, den wir dem Ding in einer Sprache geben, ist eine Bezeichnung („Symbol“), der unsere Vorstellung symbolisiert und stellvertretend für die Sache steht.

Definition (Konzept)
Ein Konzept ist die Vereinigung aller Elemente, die alle eine bestimmte Eigenschaft oder mehrere bestimmte Eigenschaften besitzen.

Anzumerken ist: Der „Konzept“-Begriff entspricht dem „Klassen“-Begriff in der objektorientierten Welt, wird aber stärker mengentheoretisch erklärt, als Zusammenfassung aller Gegenstände und Sachverhalte, die sich durch gemeinsame Merkmale auszeichnen.

Michael R. Genesereth und Nils J. Nilsson haben eine Konzeptualisierung definiert ([GN87] nach [Gua98]) als Paar (D, R), wobei D eine Domäne ist und R eine Menge von relevanten Relationen auf D. Für sie ist eine Konzeptualisierung eine abstrakte Sicht der Welt. Mike Uschold und Martin King [UmK95] beschreiben eine Konzeptualisierung als „ein intensionales semantisches Konstrukt, das die impliziten Regeln enthält, die ein Konstruktes in der Realität beschreiben.“.

Während Mike Uschold und Michael Gruninger [UG96] ausführen, der Begriff Ontologie beziehe sich auf das „gemeinsame Verständnis eines gewissen Bereichs, der von Interesse ist“, ist die meistzitierte und übernommene Definition des Begriffs Ontologie in der Informatik die von Thomas R. Gruber. Demnach ist eine Ontologie die explizite

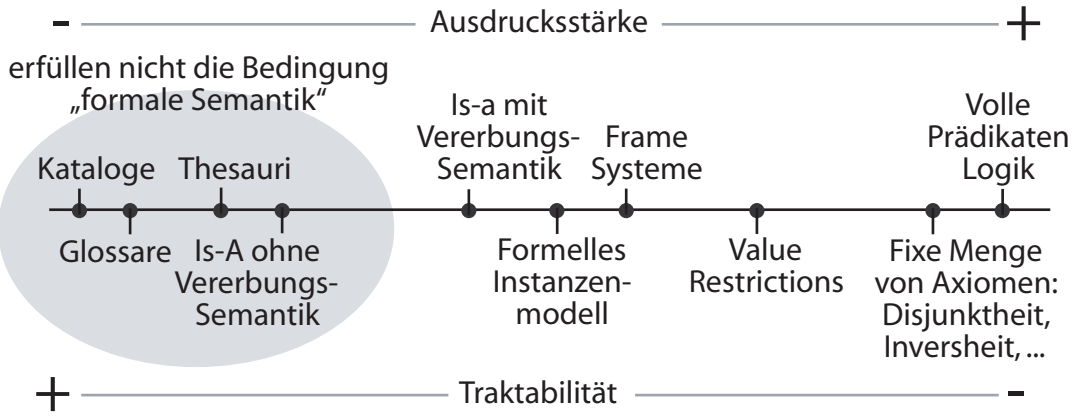


Abbildung 4.3: Klassifikation der Ontologiemodelle nach Volz [Vol01].

Spezifikation einer Konzeptualisierung [Gru93]. Hier wird der Unterschied zwischen der philosophischen Bedeutung von Ontologie und der Verwendung des Begriffs in der Informatik deutlich. Zwei Ontologien – im Sinne der Informatik – können sich unterscheiden, aber die gleiche Konzeptualisierung spezifizieren (Vgl. [Sch04]).

Gruber sagt zwar, wozu eine Ontologie gut ist. Das heißt aber noch lange nicht, dass es eine einheitliche Auffassung darüber gibt, wie eine Ontologie aufgebaut ist. Das zeigen die Übersichten von Barry Smith and Christopher Welty (siehe Abb. 4.2) auf der einen und Raphael Volz (siehe Abb. 4.3) auf der anderen Seite. Der Begriff „Ontologie“ wird für Beschreibungssysteme unterschiedlicher formaler Semantik und Komplexität verwandt, gleich, ob sie automatisches Schließen erlauben oder nicht. Im Bereich kontextsensitiver Dienste gibt es sogar Arbeiten, für die kann eine Ontologie ausschließlich mit der *Web Ontology Language* modelliert werden (vgl. [BBKG05]).

Volz will diese Verwirrung lösen, indem er „formale“ Ontologien einführt [Vol01], deren Definition allerdings noch nicht ausgereift ist: Demnach sei eine formale Ontologie

- ein fünf-Tupel $O(C, R, H^C, rel, A^O)$
- mit der Menge der Begriffe C,
- der Menge der Relationen R,
- der Begriffshierarchie $H^C \subseteq C \times C$,
- der Funktion $rel : R \rightarrow C \times C$ und
- der Menge der Axiome A^O.

Dabei gelten:

- C, R sind disjunkt.
- $H^C(C_1, C_2)$ heißt: C_1 ist *Unterbegriff* von C_2.
- Die Funktion *rel* assoziiert zwei Begriffe nicht-taxonomisch miteinander, das heißt $\forall R : rel(R) \notin H^C$. Notiert wird dies $R(C_1, C_2)$ statt $rel(R) = (C_1, C_2)$.
- Axiome A^O sind Ausdrücke in einer wählbaren Logiksprache.

Diese Definition hat noch formale Schwachstellen: Es gilt zum Beispiel zu beachten, dass die Begriffshierarchie es noch zulässt, dass ein Begriff Unterbegriff von sich selbst sein kann. Das Beispiel von Volz (Abb. 4.4) zeigt auch, dass er mit R gar nicht eine Menge von Relationen definieren wollte, sondern lediglich eine Menge von Bezeichnern (wie „angestellt_bei“ oder „beschäftigt_bei“). Erst die Funktion *rel* bildet die Bezeichner auf tatsächliche Relationen ab. Dennoch geht der Vorschlag von Volz, einen Formalismus für Ontologien zu entwickeln, in die richtige Richtung. Ohne formale Grundlage kann eine Ontologie für Kontextinformationen keine geeigneten Beziehung ausdrücken oder automatisches Schließen ermöglichen. Sie bürge auch immer die Gefahr von Zweideutigkeiten. Im Rahmen dieser Arbeit wird deshalb der Ontologiebegriff von Gruber leicht erweitert.

Definition (Ontologie)
Eine Ontologie ist die explizite, formale Spezifikation von Konzepten und deren Beziehungen zueinander. Dies umfasst auch Regeln, die bestimmen, wie die Konzepte voneinander abhängen.

Dazu passt auch der Begriff der Wissensbasis von Joachim Laubsch in [Lau91]:

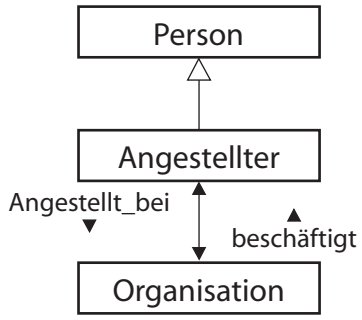
Definition (Wissensbasis)
Eine Wissensbasis W ist ein Paar $(W_0, \vdash_L)$, wobei W_0 für eine Menge von Aussagen steht, ausgedrückt in einer logischen Sprache L, und $\vdash_L$ die Ableitbarkeitsrelation in L bezeichnet. Eine Aussage α ist genau dann in W enthalten, wenn $W_0 \vdash_L \alpha$, also α aus W_0 ableitbar ist.

Eine Ontologie bestimmt mit ihrem Formalismus und ihren Konzepten die möglichen Aussagen eine Wissensbasis und deren Ableitbarkeitsrelation. Eine Wissensbasis kann sich auf mehrere Ontologien beziehen, solange diese auf demselben Formalismus basieren.

Im Folgenden werden Logiksprachen auf ihre Eignung hin untersucht, Grundlage einer formalen Modellierung von Kontextinformationen zu sein.

4.2 Logiken zur Wissensrepräsentation

Logiken wie die Prädikatenlogik oder die Aussagenlogik sind in erster Linie Sprachen. Mit ihren Begriffen lassen sich reale Dinge der Welt und ihre Beziehungen zueinander



C = { Person, Organisation, Angestellter }

R = { angestellt_bei, beschäftigt }

H^C (Angestellter, Person)

rel: angestellt_bei(Angestellter, Organisation),
beschäftigt(Organisation, Angestellter)

A_0 = { „FORALL X, Y :
Angestellter: X[angestellt_bei->Y]
AND Organisation:Y[beschäftigt->X]" }

Abbildung 4.4: Beispiel für eine „formale Ontologie" nach [Vol01].

beschreiben. Sie eignen sich deshalb zur Repräsentation von Wissen und sind somit auch Kandidaten zur Repräsentation für spezielleres Wissen wie das Kontextwissen [Kob93].

In diesem Kapitel sollen nun zunächst die für die Wissensrepräsentation wichtigsten Logiken dargestellt werden: Die Aussagenlogik, die Prädikatenlogik als Erweiterung der Aussagenlogik und die Familie der Beschreibungslogiken, die Spezialformen der Prädikatenlogik erster Stufe sind. Soweit es im Rahmen dieser Arbeit sinnvoll ist, werden auch die Grundlagen der jeweiligen Logiken eingeführt. Im Anschluss werden andere Formen der Wissensrepräsentation vorgestellt wie etwa Semantische Netze oder *frames* (auch deutsch „Konzeptrahmen"). Diese lassen sich in der Regel auf Logiksprachen abbilden, womit der Unterschied zur Logik kein struktureller, sondern nur noch ein kognitiver ist.

4.2.1 Formale Aussagenlogik

Wird Aussagenlogik (Quellen für diesen Abschnitt: [LC04, Lug01, Sch95]) zur Wissensrepräsentation verwendet, dann interessiert die Frage, ob eine Aussage „wahr" oder „falsch" ist. Elementare Aussagen werden durch aussagenlogische Variablen dargestellt. So kann die Variable *A* etwa für die Aussage „Es regnet" stehen und *B* für „Die Straße ist nass".

Aussagenlogik ist eine Ausprägung der Booleschen Algebra: Mit den Verknüpfungsoperatoren Konjunktion, Disjunktion, Negation, Implikation und Äquivalenz können zusammengesetzte Aussagen gebildet werden. Dafür gibt es auch den Begriff der *Formeln* (vgl. [LC04]). Es gilt:

Definition (Formeln der Aussagenlogik)
Alle Aussagenvariablen und die Wahrheitswerte „wahr" und „falsch" sind Formeln. Sind A und B aussagenlogische Formeln, dann auch $A \vee B$, $A \wedge B$, $\neg A$, $A \Rightarrow B$ *und* $A \Leftrightarrow B$.

Definition (Literale der Aussagenlogik)
Eine atomare Formel heißt Literal. Ist sie nicht negiert, heißt sie positives Literal. Ist sie negiert, heißt sie negatives Literal.

Beispiel dazu: *A, B,* „wahr“ und „falsch“ sind positive Literale, $\neg A$ oder $\neg$„falsch“ sind dagegen negative Literale.

Definition (Klauseln)
Eine Disjunktion von Literalen ist eine Klausel.

Beispiel: $A \vee B$, $A \vee \neg B \vee C$ und C sind Klauseln. Eine Konjunktion von Klauseln heißt Konjunktive Normalform (KNF). Jede Formel kann in Konjunktive Normalform gebracht werden. Das folgende Beispiel für eine derart normalisierte Formel ist sogar *kanonisch*, das heißt jede in der Gesamtformel vorkommende Variable kommt pro Klausel genau einmal vor:

$$(A \vee B \vee \neg C) \wedge (\neg A \vee \neg B \vee \neg C) \wedge (\neg A \vee B \vee C)$$

Eine besondere Form von Klauseln sind die *Hornklauseln*, die wichtig sind für viele Inferenzverfahren (Verfahren zum Schließen).

Definition (Hornklauseln, Ziele, definite Hornklauseln, Fakten, Regeln)
Klauseln mit maximal einem positiven Literal heißen Hornklauseln. Eine Hornklausel mit ausschließlich negativen Literalen ist ein Ziel, und eine Hornklausel mit genau einem positiven Literal ist eine definite Hornklausel. Eine definite Hornklausel, die ausschließlich aus einem positiven Literal besteht, heißt Fakt und eine definite Hornklausel, die neben dem positiven Literal auch mindestens ein negatives enthält, ist eine Regel.

Ein Beispiel für ein Ziel:

$$\neg x_1 \vee \neg x_2 \vee \ldots \vee \neg x_n$$

Ein Beispiel für eine definite Hornklausel:

$$\neg x_1 \vee \neg x_2 \vee \ldots \vee \neg x_n \vee y$$

Hornklauseln lassen sich auch als Implikationen schreiben. Ein Ziel lässt sich dann mit Worten ausdrücken als „$x_1 \vee x_2 \vee \ldots \vee x_n$ ergeben eine falsche Aussage.“ Eine definite Hornklausel lässt sich in Worten ausdrücken als „Unter der Bedingung dass $x_1 \vee x_2 \vee \ldots \vee x_n$ gilt, folgt y.“ Die Programmiersprache Prolog zum Beispiel ist gänzlich auf der Basis von Hornklauseln entwickelt worden. Eine Konjunktion von Hornklauseln ergibt so genannte *Hornformeln*. Für Hornformeln lässt sich in polynomieller Zeit feststellen, ob eine Variablenbelegung existiert, so dass die Formel wahr wird. Das heißt: Ihre Erfüllbarkeit ist entscheidbar.

Werden den Variablen in den Formeln Ereignisse der realen Welt zugeordnet, spricht man von einer *Interpretation*. Jede aussagenlogische Variable erhält dabei einen Wahrheitswert („wahr“ oder „falsch“).

Definition (Interpretation in der Aussagenlogik)
Sei M die Menge aller aussagenlogischen Formeln. Eine Funktion $I : M \to \{W, F\}$ heißt Interpretation.

Zum Beispiel nehme man die Formel $A \wedge B$. Wird A in der realen Welt die Aussage „München liegt in Bayern“ und B die Aussage „Bayern ist größer als Deutschland“ zugeordnet, dann ist A genauso offensichtlich wahr wie B falsch ist. Also kennt man den Wahrheitswert der zusammengesetzten Aussage: $A \wedge B$ ist falsch. Wenn in einer anderen Interpretation A für „Bayern ist ein Freistaat“ und B für „Bayern ist in Deutschland“ steht, wird $A \wedge B$ dagegen eindeutig wahr.

Definition (Modell)
Ist eine Formel ϕ unter einer Interpretation I wahr, so bezeichnet man I als Modell von ϕ.

Definition (Erfüllbarkeit)
Eine Formel heißt erfüllbar (falsifizierbar), falls eine Interpretation existiert, so dass die Formel wahr (falsch) wird. Sie heißt allgemeingültig, wenn die Formel unter jeder Interpretation wahr wird und sie heißt unerfüllbar, wenn keine Interpretation existiert, so dass sie wahr wird.

Aussagenlogik kann nicht nur zur Repräsentation, sondern auch zur (automatischen) Verarbeitung benutzt werden. Dabei interessiert, ob aus einer gegebenen Menge von Aussagen eine Behauptung zwingend folgt oder nicht. Dazu dient der Begriff der *semantischen Folgerung*.

Definition (Semantische Folgerung)
Eine Formel ϕ ist eine semantische Folgerung der Menge Φ von Formeln, falls jedes Modell von Φ auch ein Modell von ϕ ist. Man schreibt: $\Phi \models \phi$ und sagt auch ϕ folgt aus Φ.

Wie beantwortet man die Frage, ob eine Formel ϕ aus dem bereits existierenden Wissen Φ folgt? Man kann versuchen, eine Antwort mit Hilfe von Inferenzregeln wie dem *Modus ponens* und dem *Modus tollens* zu finden. Der Modus ponens besagt:

$$\frac{\begin{array}{l} A \Rightarrow B \\ A \end{array}}{B}$$

Das bedeutet: Wenn A impliziert B und A gilt, dann gilt auch B. Entsprechend gilt auch der Modus tollens:

$$\frac{\begin{array}{l} A \Rightarrow B \\ \neg B \end{array}}{\neg A}$$

Definition (Ableitbarkeit)
Existiert eine endliche Folge von (syntaktischen) Inferenzschritten, so dass man von Φ *zu* ϕ *gelangt, so heißt* ϕ *ableitbar aus* Φ*. Man schreibt:* $\Phi \vdash \phi$.

Der Modus ponens alleine ermöglicht nur eine sehr begrenzte Menge von Beweisen. Im Verbund mit anderen Inferenzregeln wird das Verfahren dagegen sehr komplex. Deshalb nimmt man das 1965 von J. A. Robinson eingeführte Resolutionsverfahren zum automatischen Beweisen her, da es allein auf einer einzigen Inferenzregel aufbaut. Die Idee besagt, dass man dem (widerspruchsfreien) Wissen Φ die Formel $\neg\phi$ hinzufügt und dann einen Widerspruch daraus ableitet, womit ϕ bewiesen wäre.

Dazu formt man Φ zu einer Menge von *Klauseln* um, das heißt zu Formeln, die nur aus Disjunktionen von Literalen (positiven oder negierten Aussagevariablen) bestehen. Die Resolutionsregel ist anwendbar auf Klauseln, die komplementäre Literale enthalten. Hat man zwei Klauseln der Form

$$L_1 \vee L_2 \vee L_3 \vee \cdots \vee L_n$$
$$\neg L_1 \vee K_2 \vee K_3 \vee \cdots \vee K_m$$

so werden die komplementären Literale gelöscht und die übrigen zusammengefügt.

$$L_2 \vee L_3 \vee \cdots \vee L_n \vee K_2 \vee K_3 \vee \cdots \vee K_m$$

Erhält man dabei eine leere Klausel (weil sich Literale wie $A \vee \neg A$ aufheben), so ist der Widerspruch geglückt bewiesen. Das bedeutet ϕ ist ableitbar aus Φ.

Die Umwandlung der Formeln des Wissens Φ in konjunktive Normalformen, also zu einer Menge von *Klauseln*, ist ein einfacher, maschinenausführbarer Algorithmus. Weiter verlangt die Resolutionsregel nur das iterative Suchen nach Klauselpaaren, die komplementäre Literale enthalten. Auch dies ist einfach in einen Algorithmus umzusetzen, und damit auf einen Rechner übertragbar.

Inferenzverfahren wie das Resolutionsverfahren bewertet man daran, ob sie korrekt sind, und ob sie vollständig sind.

Definition (Korrektheit)
Ein Beweisverfahren heißt korrekt, wenn für beliebige Formeln Φ, ϕ *gilt: Falls* $\Phi \vdash \phi$, *dann gilt auch* $\Phi \models \phi$.

Definition (Vollständigkeit)
Ein Beweisverfahren heißt vollständig, wenn für beliebige Formeln Φ, ϕ *gilt: Falls* $\Phi \models \phi$, *dann gilt auch* $\Phi \vdash \phi$.

Das Resolutionsverfahren ist zwar korrekt, aber nicht vollständig. Die fehlende Vollständigkeit zeigt sich beispielsweise darin, dass man allein mit der Resolution aus den beiden Klauseln A und B nicht ableiten kann, so dass gilt: $A \vee B$. Diese Art der Vollständigkeit ist aber gar nicht entscheidend für die Resolution, da ihr Anwendungsgebiet lediglich das

Auffinden von Widersprüchen ist. Man spricht deshalb davon, dass das Resolutionsverfahren „widerspruchsvollständig“ ist, was bedeutet, dass eine widersprüchliche Klauselmenge immer in endlichen Resolutionsschritten als solche erkannt wird.

Die Vorteile der Aussagenlogik als Kandidat zur Repräsentation von Kontextwissen sind ihre starke Formalität, ihre Entscheidbarkeit und die Einfachheit ihrer Inferenzverfahren. Allerdings ist die Aussagenlogik nicht mächtig genug hinsichtlich ihrer Ausdrucksstärke. Es sind nur elementare Aussagen möglich. Kontextinformationen müssen dagegen in Entität, Kontextinformationsart und Wert der Kontextinformation aufgeteilt werden. Außerdem müssen Beziehungen zwischen diesen Elementen ausgedrückt werden können. Auch Quantifizierungen lässt die Aussagenlogik nicht zu. Im folgenden Abschnitt wird deshalb eine mächtigere Logik vorgestellt.

4.2.2 Prädikatenlogik

Die Prädikatenlogik (Quellen für den Abschnitt über Prädikatenlogik: [Her97, Lug01, Sch95]) (engl. *„first-order logic“ (FOL)*) erweitert die Aussagenlogik um Prädikate und Quantoren. Ihr historischer Ursprung ist die „Begriffsschrift“, die der deutsche Logiker Gottlob Frege 1879 vorgestellt hat, und die abgesehen von syntaktischen Abweichungen mit der Sprache der Prädikatenlogik (nicht notwendigerweise der ersten Stufe) übereinstimmt [BHS93]. Alison Cawsey bezeichnet die Prädikatenlogik (der ersten Stufe) als wichtigste Wissensrepräsentationssprache überhaupt [Caw03]. Erweiterungen der Prädikatenlogik erster Stufe sind unter anderem die Temporale Logik und die Modallogik [Mat97].

Stand in der Aussagenlogik eine Variable X noch für eine komplette Aussage, die wahr oder falsch sein kann, wie etwa „Rolf ist größer als Peter“, so werden Aussagen in der Prädikatenlogik modular modelliert. Das Beispiel von eben könnte dann in der Präfixnotation der Prädikatenlogik so lauten: $Ist\_groesser(rolf, peter)$. Was die Prädikatenlogik der Aussagenlogik voraus hat, sind Quantoren, Funktions- und Prädikatensymbole. Damit wird es möglich, bestimmte Objekte in Beziehung zueinander zu setzen, auszudrücken, dass eine Eigenschaft *für alle* Objekte gilt, oder dass ein Objekt mit einer gewissen Eigenschaft *existiert* [Sch95].

Definition (Terme der Prädikatenlogik)
Ein Term ist eine Konstante k, eine Variable V oder ein n-stelliger Funktionsausdruck $f(t_1, \cdots, t_n)$ *mit* $n \geq 0$ *und Termen* $t_1, \cdots, t_n$.

Mit Konstanten werden Objekte oder Eigenschaften der realen Welt bezeichnet, wie „mobiltelefon“, „alice“, „klein“ oder „rot“. Dabei werden Konstantensymbole grundsätzlich mit Kleinbuchstaben geschrieben. Die Konstanten „wahr“ und „falsch“ sind für die Wahrheitswerte reserviert. Variablensymbole dagegen beginnen immer mit einem Großbuchstaben.

Definition (Atome der Prädikatenlogik)
Ein Atom ist ein n-stelliges Prädikatssymbol P über Termen.

Beispiele dafür sind etwa $Kalt(X)$ oder $Neben(a,b)$. Auch wenn Prädikate wie Funktionen aussehen, beschreiben sie keine Abbildung, sondern treffen Aussagen über Eigenschaften ihre Terme, die unabhängig vom konkreten Term sind. Es ist sinnvoll, Funktionen- und Prädikatssymbole zu unterscheiden, indem erstere klein und letztere groß geschrieben werden.

Definition (Formeln der Prädikatenlogik)
Anwendungen von $\forall, \exists, \wedge, \vee, \neg$ *auf Atome sind Formeln.*

Definition (Literale der Prädikatenlogik)
Literale sind Atome und negierte Atome.

Literale und Formeln der Prädikatenlogik sind mächtiger als Literale und Formeln der Aussagenlogik. Die aussagenlogische Definition von Klauseln und Hornklauseln, sowie Ableitbarkeit, Semantische Folgerung, Erfüllbarkeit und Korrektheit gelten in der Prädikatenlogik ebenfalls, allerdings entsprechend des mächtigeren Literal- und Formelbegriffs. Neu definiert in der Prädikatenlogik ist dagegen der Begriff der Interpretation, der die Semantik der Prädikatenlogik beschreibt:

Definition (Interpretation in der Prädikatenlogik)
Eine Domäne D ist eine nicht leere Menge von Entitäten. Eine Interpretation I in Bezug auf D ist eine Zuweisung der Entitäten zu den einzelnen Konstanten-, Variablen-, Prädikats- und Funktionssymbolen eines prädikatenlogischen Ausdrucks derart, dass Folgendes gilt:

1. *Jedem Konstantensymbol wird ein Element aus D zugewiesen.*
2. *Jeder Variablen wird eine nicht leere Teilmenge von D zugewiesen, wobei diese Teilmengen die zulässigen Substitutionen der Variablen darstellen.*
3. *Jede Funktion f mit der Stelligkeit m beschreibt eine Abbildung von D^m auf D.*
4. *Jedes Prädikat P mit der Stelligkeit n beschreibt eine Abbildung von D^n auf der Menge $\{wahr, falsch\}$.*

Ist eine konkrete Interpretation gegeben, lässt sich der Wahrheitswert für einen prädikatenlogischen Ausdruck ermitteln.

Mit den Prädikaten lassen sich Entitäten Eigenschaften zuordnen, wie etwa Beziehungen zu anderen Entitäten. Variablen dürfen in Formeln nur noch verwendet werden, wenn sie durch einen Allquantor $\forall$ oder einen Existenzquantor $\exists$ quantifiziert sind. Solange lediglich Variablen und nicht Prädikate oder Funktionen quantifiziert sind, spricht man von der *Prädikatenlogik erster Stufe*.

Nach dem Satz von Church ist die Prädikatenlogik hinsichtlich ihrer Gültigkeit und ihrer Erfüllbarkeit unentscheidbar [Sch95]. Das heißt zum einen, es gibt keinen algorithmischen Test, mit dem festgestellt werden kann, ob eine Formel (allgemein-)gültig ist. Besonders wichtig wäre das für Implikationen. Und zum anderen bedeutet der Satz von Church, dass

es keinen algorithmischen Test gibt, ob eine Formel erfüllbar ist, sprich, ob eine Interpretation existiert, für die die Formel wahr ist.

Die Prädikatenlogik wäre zwar bezüglich ihrer Aussagekraft mächtig genug, Kontextwissen auszudrücken. Doch ihre Nicht-Entscheidbarkeit ist ein zu großer Nachteil, um sie als Kandidaten in Erwägung zu ziehen. Die Verarbeitung von Kontextwissen und das Schließen auf diesem Wissen wären zu sehr eingeschränkt.

Deshalb wird im nächsten Kapitel eine Familie von Logiksprachen vorgestellt, die diesen Nachteil nicht besitzt. Dazu muss zunächst eine Spezialisierung der Prädikatenlogik eingeführt werden:

Definition (Prädikatenlogik erster Stufe)
In der Prädikatenlogik erster Stufe dürfen sich Quantoren nur auf Elemente der Domäne beziehen, nicht auf Funktionen und Prädikte.

Damit werden Formeln der Art „Für jede Eigenschaft gilt…“ oder „Es existiert eine Funktion, die…“ in der spezielleren Prädikatenlogik der ersten Stufe nicht behandelt. Sie ist aber immer noch mächtig genug, um etwa die gesamte Mengenlehre zu formalisieren.

4.2.3 Beschreibungslogiken

Die heutigen Beschreibungslogiken (ältere Bezeichnungen sind „*terminological systems*“ oder „*concept languages*“) bestehen aus einer entscheidbaren Teilmenge der Prädikatenlogik erster Stufe. Sie haben sich entwickelt aus frühen Vertretern der Semantischen Netze und der Konzeptrahmen (engl. „*frames*“). Diese sollen nun zunächst vorgestellt werden.

Konzeptrahmen (Frames)

Marvin Minsky beschreibt 1975 einen *frame* als statische Datenstruktur, mit deren Hilfe stereotype Situationen dargestellt werden können (nach [Lug01]). Er enthält *slots* für Attribute, die wieder in *Facetten* (zum Beispiel für den Namen des Attributes, für den Wert, für den Typ usw.) unterteilt sind. Ein solcher Konzeptrahmen (ein „Konzept“ hieße im objektorientierten Umfeld „Klasse“) beinhaltet zum Beispiel

- Informationen zur Identifikation des Konzeptrahmens,
- Beziehungen zu anderen Konzeptrahmen,
- Beschreibungen der Anforderungen für einen Konzeptrahmen,
- Prozedurale Informationen über die Verwendung der beschriebenen Struktur,
- Standardinformationen für den Konzeptrahmen. Das sind Werte, die gelten, wenn keine gegenteiligen Informationen bekannt sind und
- Informationen über neue Instanzen.

Konzepthierarchie	Attributdefinitionen

Objekt[].
 	
Person :: Objekt.
 	
Angestellter :: Person.
 	
Forscher :: Angestellter.
 	
Publikation :: Objekt.
	Person[
	vorname =>> STRING;
	nachname =>> STRING;
	eMail =>> STRING;
	publikation =>> Publikation;
	...].
	
	Angestellter[
	zugehoerig =>> Organisation;
	...].
	
	Forscher[
	forschungsGebiet =>> Thema;
	kooperiertMit =>> Forscher;
	...].
	
	Publikation[
	autor =>> Person;
	titel =>> STRING;
	jahr =>> NUMBER;
	zusammenfassung =>> STRING].

Regeln

```
FORALL Person1, Person2
  Person1:Forscher[kooperiertMit ->> Person2] <-
    Person2:Forscher[kooperiertMit ->> Person 1].

FORALL Person1, Publ1
  Publ1:Publikation[autor ->> Person1] <->
    Person1:Person[publikation ->> Publ1].

FORALL O,C,A,V,T
  V:T <- C[A=>>T] AND O:C[A->>V]
```

Abbildung 4.5: Ausschnitt einer Beispiel-Ontology mit F-Logik (nach [DEFS99]).

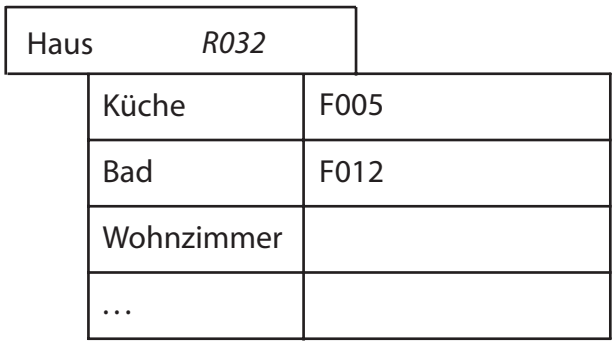


Abbildung 4.6: Ausschnitt eines *frames*.

Ein System mit *frames* unterstützt Klassen- und Vererbungshierarchien. Die Möglichkeit der Angabe von Standardwerten führt zu einem *nichtmonotonen* Verhalten beim Schließen auf *frames*. Das bedeutet, dass das Ergebnis von Schlussregeln nicht immer deterministisch ist.

In einem Beispiel aus [BHS93] repräsentiert ein *frame* namens „Haus“ ein typisches Haus und hält *slots* vor für die Attribute „Küche“, „Bad“, „Wohnzimmer“, „Schlafzimmer“, „Kinderzimmer“, „Besitzer“ und so weiter. Um ein bestimmtes Haus (zum Beispiel das Haus „R032“) zu beschreiben, genügt es im einfachsten Fall, Bezeichner für die Wertfacetten anzugeben. Wenn das Bad den Bezeichner „F012“ hat und die Küche mit „F05“ bezeichnet ist, sieht der Konzeptrahmen aus wie in Abbildung 4.6.

Logisch entspricht diesem teilweise instanziierten *frame* die Formel

$$\begin{aligned} \mathit{ist\_ein\_Haus}(\mathrm{R032}) \Leftrightarrow \quad & \mathit{ist\_die\_Kueche\_von}(\mathrm{F005},\mathrm{R032}) \wedge \\ & \mathit{ist\_ein\_Bad\_von}(\mathrm{F012},\mathrm{R032}) \wedge \\ & \exists y\ \mathit{ist\_das\_Wohnzimmer\_von}(y,\mathrm{R032}) \wedge \ldots \end{aligned}$$

Allgemein gilt für ein durch einen *frame* definiertes Konzept K (im Beispiel $K =$„Haus“) die Beziehung:

$$Kx \Leftrightarrow \exists y_1 y_2 \ldots y_n [P_1(y_1,x) \wedge P_2(y_2,x) \wedge \cdots \wedge P_n(y_n,x)]$$

Das x ist hier als allquantifiziert zu betrachten. K wird auch *generisches Konzept* genannt, weil durch die Instanziierung von $y_1,\ldots,y_n$ eine Menge von individuellen Konzepten bestimmt ist.

Die Forschungsarbeiten des MIT zu *frames* und ähnliche Arbeiten am Xerox Palo Alto Research Center führten zur Entwurfsphilosophie der objektorientierten Programmierung. In Verbindung zu den *frame*-Sprachen steht auch F-Logik (von „Frame-Logik“). Diese Datenbanksprache erläutern Michael Kifer, Georg Lausen und James Wu 1995 in [KLW95]. *Frame*-Sprachen und F-Logik haben gemeinsam, dass sie auf den Konzepten von komplexen Objekten, Vererbung und logischem Schließen aufbauen. F-Logik verbindet die

Ansätze zu objektorientierten Datenbanken auf der einen und deduktiven Datenbanken auf der anderen Seite. Dabei kann F-Logik zur Definition, Abfrage und Manipulation von Daten und des Datenbankschemas verwandt werden. Die Sprache besitzt eine modelltheoretische Semantik und eine vollständige und korrekte resolutionsbasierte Beweistheorie.

Mit einer leicht abgewandelten Variante von F-Logik haben Stefan Decker und andere [DEFS99] 1999 ein System zur ontologiebasierten Beschreibung von Informationen im World Wide Web vorgestellt. Diese Ontologien bestehen aus drei Teilen (siehe auch Abb. 4.5:

- Die *Concept Hierarchy* definiert die Vererbungsbeziehung zwischen verschiedenen Konzepten.
- Unter *Attribute Definitions* sind die von Attribute der Konzepte definiert.
- Regeln definieren die Beziehungen zwischen verschiedenen Konzepten und Attributen im *Rules*-Teil.

Dafür werden die folgenden, elementaren Modellierungsprimitive benutzt (Beispiel siehe Abb. 4.5):

Unterklassen `C1::C2` bedeutet, dass `C1` eine Unterklasse von `C2` ist.

Instanzen `O:C` bedeutet, dass `O` eine Instanz von `C` ist.

Attributdeklaration `C1[A =>> C2]` bedeutet, dass für eine Instanz der Klasse `C1` ein Attribut `A` definiert ist, dessen Wert eine Instanz von `C2` sein muss.

Attributwert `O[A ->> V]` bedeutet, dass eine Instanz `O` ein Attribut `A` mit dem Wert `V` hat.

Komposition `O1 <: O2` bedeutet, dass O1 ein Teil von O2 ist.

Beziehungen Prädikatausdrücke wie `p(a1,...,a2)` können benutzt werden, wie man das von Logiksprachen gewohnt ist, mit der Erweiterung, dass nicht nur Terme, sondern auch Objekte als Argumente erlaubt sind.

Nach Wolfgang Bibel [BHS93] handelt es sich bei *frames* lediglich um eine „Verfeinerung der Ideen (...), die bereits bei assoziativen Netzen behandelt wurden“. Aus logischer und algorithmischer Sicht erkennt er keinen originellen Beitrag zur Wissensrepräsentation, da die Organisation des Wissens äquivalent zu einer Repräsentation mit Mitteln der Logik sei. Seiner Einschätzung nach beruht die Attraktivität der *frames* auf einem kognitiven Aspekt: Sie organisieren Wissen auf eine Art und Weise, die der menschlichen ähnlich und dem Menschen damit vertraut ist.

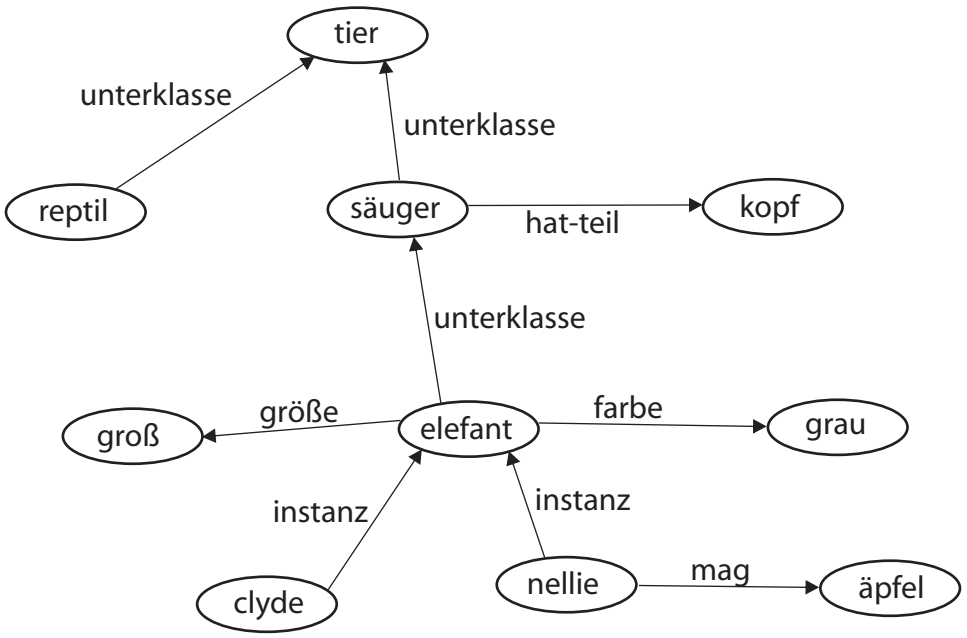


Abbildung 4.7: Beispiel eines einfachen semantischen Netzes (nach [Caw03]).

Semantische Netze

So wie die mathematische Logik historisch auf Versuche zurück geht, die natürliche Sprache und menschliches Denken zu formalisieren, so sind auch Semantische Netze in den sechziger Jahren des 20. Jahrhunderts entwickelt worden, um die Bedeutung von Worten darzustellen. Viele frühe Entwürfe von semantischen Netzen (siehe auch Abbildung 4.7) erkannten noch nicht die Probleme bei der Definition der Beziehung einer Klasse zu einem Element oder der Beziehung zwischen Klasse und Unterklasse.

Konzeptgraphen Moderne Netzwerkrepräsentationssprachen wie die 1984 von John Sowa eingeführten Konzeptgraphen (nach [Lug01]) sind da semantisch stärker formalisiert. Ein Konzeptgraph ist ein endlicher, verbundener bipartiter Graph (die Knoten eines *bipartiten* Graphen lassen sich so in zwei Mengen teilen, dass jede Kante einen Knoten aus der einen und einen Knoten aus der anderen Menge besitzt). Bei den Knoten handelt es sich entweder um Konzepte oder Konzeptrelationen. Beschriftete Kanten (wie in Abbildung 4.7) gibt es in Konzeptgraphen nicht. Ein- oder mehrstellige Relationen zwischen Konzepten werden durch Konzeptrelationsknoten dargestellt. In Abbildung 4.8 etwa ist der Graph für die Aussage „Mary gab John ein Buch“ dargestellt.

Konzeptgraphen sind in ihrer Ausdrucksstärke genauso mächtig wie die Prädikatenlogik, da sich eine Abbildung von Konzeptgraphen in die Notation der Prädikatenlogik finden lässt. Allerdings stellen Konzeptgraphen eine Reihe von Inferenzmechanismen für

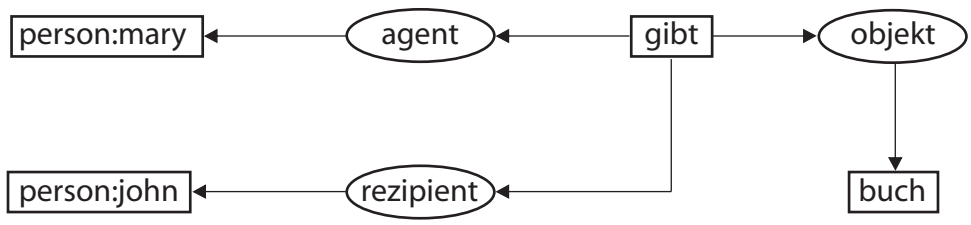


Abbildung 4.8: Beispiel eines Konzeptgraphen (nach [Lug01]).

besondere Zwecke bereit, die normalerweise nicht Teil der Prädikatenlogik sind. Dazu gehören die *join*- und die *restrict*-Operation. Diese beiden sind Spezialisierungsregeln, die eine partielle Ordnung über der Menge der ableitbaren Graphen definieren. Die *restrict*-Operationen ermöglichen es, Konzeptknoten in einem Graphen durch einen Knoten zu ersetzen, der eine Spezialisierung dieser Knoten darstellt. Mit Hilfe der *join*-Operation können zwei Graphen zu einem Graphen verknüpft werden, wenn ein Konzeptknoten in einem Graphen identisch mit einem Konzeptknoten des anderen ist. Wichtig dabei: Diese Operationen sind so genannte kanonische Formationsregeln, die zwar nicht wahrheitserhaltend sind, aber bedeutungserhaltend.

Topic Maps Eine weitere Ausprägung von modernen semantischen Netzen sind die Topic Maps, die 1999 als ISO Standard spezifiziert worden sind und später als XML Topic Maps (XTM) in XML formuliert worden sind. [BNRM+01]. Topic Maps beschreiben Subjekte (adressierbare, also solche, die in Rechnern gespeichert werden können, und nicht adressierbare, die nicht in Rechnern gespeichert werden können) in einer Art Overlay-Netz und sind damit auf gewisse Art implementierungsunabhängig. Die Grundkonstrukte sind *topics*, *associations* und *occurrences* (Instanzen). Topic Maps unterstützen Reifikation (der Typ eines *topics* ist wieder ein *topic*) und sind sehr ausdrucksstark. Allerdings fehlt es ihnen an formaler Grundlage. Ziel ist es, ein *Topic Maps Reference Model* als Metamodell für Topic Maps zu entwerfen (nach [Fuc04]). Dafür sind dem *World Wide Web Consortium (W3C)* bereits *drafts* vorgelegt worden [PG04]. Und es existiert ein Entwurf für eine mögliche OWL-DL-Umsetzung des Datenmodells der Topic Maps [Cre05].

Grundlagen der Beschreibungslogik

Aus den Semantischen Netzen haben sich die so genannten Strukturierten Vererbungsnetze entwickelt, die Ronald J. Brachman 1977 eingeführt hat [BN03]. Ihre nachfolgend genannten, grundlegenden Ideen haben die Entwicklung der Beschreibungslogiken [BN03] bestimmt:

- Die grundlegenden syntaktischen Elemente sind *atomare Konzepte* (unäre Prädikate), *atomare Rollen* (binäre Prädikate) und Individuen (Konstanten).

- Die Ausdruckskraft der Sprachen ist dadurch eingeschränkt, dass sie zur Bildung von komplexen Konzepten und Rollen nur eine kleine Menge von Konstrukten benutzt.
- Implizites Wissen über Konzepte und Individuen kann automatisch mit Hilfe von Inferenzprozeduren geschlossen werden. Insbesondere Unterklassenbeziehungen zwischen Konzepten und Individuen spielen dabei eine große Rolle: Anders als die „*IS-A*"-Verknüpfung in semantischen Netzen, die der Benutzer explizit einzieht, können die Unterklassen- und die Instanzbeziehungen aus der Definition der Konzepte und der Eigenschaften der Individuen automatisch geschlossen werden.

Zur Begriffsklärung: In der objektorientierten Welt wäre die „Klasse" die Entsprechung des „Konzepts" und das „Objekt" die Entsprechung des „Individuums". Die „Rolle" der Beschreibungslogik findet in Attributen und Assoziationen ihre Entsprechung.

Definition (Konzeptterme in der Beschreibungslogik)
Ein Konzeptterm C beschreibt die hinreichenden und notwendigen Bedingungen für ein Konzept namens A.

Eine Beschreibungslogik wird dadurch bestimmt, welche Fragmente der Prädikatenlogik sie zulässt. Die minimale Sprache haben Manfred Schmidt-Schauß und Gert Smolka 1991 mit der *attributive language* $\mathcal{AL}$ eingeführt (nach [BN03]). Seien A,B atomare Konzepte, R eine atomare Rolle, und C,D Konzeptterme. Dann sind die folgenden Konzeptterme erlaubt:

- A (ein atomares Konzept wie „Mann", „Frau", „Mobiltelefon"),
- $\top$ (das universelle Konzept, zu dem alle Individuen gehören),
- $\bot$ (das leere Konzept, zu dem kein Individuum gehört),
- $\neg A$ (die atomare Negation, die bedeutet: alles, was nicht zum atomaren Konzept A gehört),
- $C \sqcap D$ (der Durchschnitt bzw. die Konjunktion, was bedeutet: alle Individuen, die sowohl mit dem Konzeptterm C als auch mit dem Konzeptterm D beschrieben werden),
- $\forall R.C$ (die Wertrestriktion; zum Beispiel $\forall besitzt.Mobiltelefon$, was bedeutet: alle Individuen, die als Eigenschaft „besitzt" ausschließlich Individuen haben, die zum Konzept „Mobiltelefon" gehören) oder
- $\exists R.\top$ (die limitierte Existenzquantifikation; zum Beispiel $\exists name.\top$, was bedeutet: alle Individuen, die über die Eigenschaft „name" mindestens ein anderes Individuum besitzen, sprich, mindestens einen Namen haben).

Die Sprache wird erweitert, indem weitere Fragmente der Prädikatenlogik als Konstruktoren erlaubt werden. Es ist Konvention, dabei den Namen der Beschreibungslogik um die jeweiligen Stellvertreterbuchstaben der Konstruktoren zu erweitern. So gibt es auch

- ($\mathcal{U}$) $C \sqcup D$ (die Vereinigung bzw. die Disjunktion, die bedeutet: Alle Individuen, die durch den Konzeptterm C oder den Konzeptterm D beschrieben werden),
- ($\mathcal{E}$) $\exists R.C$ (die volle existenzielle Restriktion; zum Beispiel $\exists nutzt.Mobiltelefon$ bedeutet: Alle Individuen, die über die Eigenschaft „nutzt“ mindestens ein Individuum des Konzeptes „Mobiltelefon“ besitzen),
- ($\mathcal{C}$) $\neg C$ (die volle Negation, die bedeutet: alle Individuen, die nicht durch den Konzeptterm C beschrieben werden),
- ($\mathcal{N}$) $\geqslant_n R$ u. $\leqslant_n R$ (die Kardinalitätsrestriktionen; zum Beispiel $\geqslant_2 hatFreund$, was bedeutet: alle Individuen, die mindestens zwei Eigenschaften „hatFreund“, sprich zwei Freunde besitzen),
- ($\mathcal{H}$) $R_1 \sqsubseteq R_2$ (die Rollenhierarchie, zum Beispiel: $hatTocher \sqsubseteq hatKind$),
- ($\mathcal{O}$) a (der Singleton Konstruktor für ein Konzept, das nur aus dem Individuum a besteht),
- ($\mathcal{I}$) $R_1 \equiv R_2^-$ (die inversen Rollen, zum Beispiel: $hatTochter \equiv istTochterVon^-$),
- ($\mathcal{Q}$) $\geqslant_n R.C$ u. $\leqslant_n R.C$ (die qualifizierte Kardinalitätsrestriktion; zum Beispiel $\geqslant_2 hatFreund.Frau$, was bedeutet: alle Individuen, die mindestens zwei Freundinnen haben).

Eine Sprache, die $\mathcal{AL}$ um transitive Rollen und volle Negation erweitert, wird oft mit $\mathcal{S}$ abgekürzt. Eine wichtige Beschreibungslogik heißt deshalb $\mathcal{SHIQ}$. Ihre Spezialform $\mathcal{SHOIN}(D)$ ist unter anderem die Basis für Dialekte der *Web Ontology Language (OWL)*, die im folgenden Abschnitt vorgestellt wird.

Ein Wissensrepräsentationssystem, das auf einer Beschreibungslogik basiert (siehe Abb. 4.9) bietet die Möglichkeit, Wissensbasen zu realisieren, Schlüsse aus ihrem Inhalt zu ziehen und diese zu manipulieren. Eine Wissensbasis beinhaltet zwei Komponenten: Eine *TBox* (vom englischen „*terminology box*“) mit der Terminologie für die Wissensbasis und eine ABox (vom englischen „*assertional box*“) mit Aussagen über Individuen in Begriffen der Terminologie. In einem Wissensrepräsentationssystem ist es auch möglich, neben TBox und ABox weitere Regeln zu installieren. Das System bietet Schnittstellen, um auf dem gespeicherten Wissen zu schließen. Etwa, ob es ein Modell (im Sinn einer Interpretation) gibt, das die Beschreibungen erfüllt.

Je nachdem, welche Fragmente der Prädikatenlogik in einer Beschreibungslogik zugelassen werden, verändern sich Mächtigkeit der Ausdrucksstärke und Komplexität. Alle

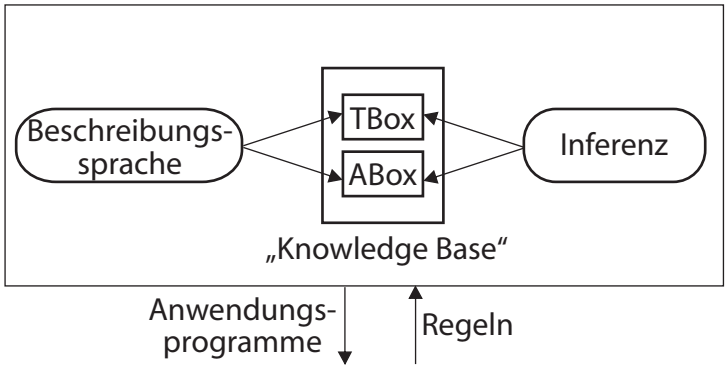


Abbildung 4.9: Architektur eines Wissensrepräsentationssystem basierend auf Beschreibungslogik (nach [BMNPS03]).

Beschreibungslogiken haben den entscheidenden Vorteil, entscheidbar zu sein. Damit bieten sie sich zur Kontextmodellierung an. Im Folgenden soll nun die bereits erwähnte *Web Ontology Language* mit einem auf Beschreibungslogik basierenden Dialekt erläutert werden.

4.3 Das Semantic Web und die Web Ontology Language

In ihrem berühmten Artikel [BLHL01] beschreiben Tim Berners-Lee, James Hendler und Ora Lassila ein Szenario, das den typischen Szenarien für kontextsensitive Dienste sehr ähnelt: Bei einem Telefonanruf werden alle Geräte, die über einen Lautstärkeregler verfügen, automatisch leiser gestellt. Persönliche „Semantic-Web“-Agenten suchen innerhalb eines 20-Meilen-Radius’ nach einem Facharzt, der noch freie Termine hat, eine sehr gute Reputation, und bei dem ein Besuch von der eigenen Krankenkasse erstattet wird.

Anders als die Forschung kontextsensitiver Dienste legen Berners-Lee et al. den Schwerpunkt auf das World Wide Web im Internet als Informationsquelle und vernachlässigen damit notgedrungen den direkten Kontext des Nutzers, der nur dynamisch über Sensoren erfasst werden könnte. Bevor das Web aber als angemessene Informationsquelle dienen kann, müsse es erweitert werden, schreiben sie. Zwar besitze es bisher die Stärke, „alles mit allem“ verbinden zu können, doch sei das Web eher ein „Medium von Dokumenten für Menschen, als eines, in dem Daten und Informationen automatisch verarbeitet werden können.“ Dazu müssten die Informationen so hinterlegt sein, dass Maschinen ihre Bedeutung erfassen könnten:

> „Die Herausforderung des *Semantic Web* besteht darin, eine Sprache zu finden, die sowohl Informationen als auch Regeln zur Inferenz auf den Informationen beschreiben kann. Sie muss es außerdem erlauben, dass Regeln jedes existierenden Inferenzsystems in das *Web* exportiert werden können.“

4.3.1 Das Resource Description Framework

Das *Resource Description Framework (RDF)* baut auf dem Konzept des *Uniform Resource Identifier (URI)* [UPIG01] und der Auszeichnungssprache *Extensible Markup Language (XML)* [YCB+04] auf. Es besteht aus den sechs *W3C-Recommendations „RDF/XML Syntax Specification (Revised)“, „RDF Vocabulary Description Language 1.0: RDF Schema“, „RDF Primer“, „Resource Description Framework (RDF): Concepts and Abstract Syntax“, „RDF Semantics W3C Recommendation“* und *„RDF Test Cases“* [MSB04]. Diese Spezifikationen wollen eine *„lightweight ontology“* für den Austausch von Wissen im Web bieten.

Sie dienen dazu, Ressourcen über einen URI eindeutig zu identifizieren (auch zu lokalisieren, wenn der URI ein gültiger *Uniform Resource Locator (URL)* ist) und zu beschreiben. Alle Aussagen in RDF haben dabei die Form eines Tripels mit den Elementen:

- *„Subject“* kann ein beliebiges Objekt sein, welches über einen URI eindeutig identifiziert ist.
- *„Predicate“* oder *„Property“* ist die Eigenschaft des Objekts, das beschrieben wird.
- *„Object“* ist der Wert, den die Eigenschaft hat. Dies kann ein Datenwert sein oder wieder ein Verweis auf eine andere Ressource.

Listing 4.1 zeigt ein Beispiel (entnommen www.w3schools.com/rdf/rdf_example.asp) für eine RDF-Beschreibung. Hier sind die beiden CDs „Empire Burlesque“ und „Hide your heart“ eines Online-Shops die Subjekte, denen Werte für die Prädikate „Künstler“, „Land“, „Gesellschaft“, „Preis“ und „Jahr“ zugeordnet werden.

```
<?xml version="1.0"?>

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:cd="http://www.recshop.fake/cd#">

<rdf:Description
 rdf:about="http://www.recshop.fake/cd/Empire␣Burlesque">
  <cd:artist>Bob Dylan</cd:artist>
  <cd:country>USA</cd:country>
  <cd:company>Columbia</cd:company>
  <cd:price>10.90</cd:price>
  <cd:year>1985</cd:year>
</rdf:Description>

<rdf:Description
 rdf:about="http://www.recshop.fake/cd/Hide␣your␣heart">
  <cd:artist>Bonnie Tyler</cd:artist>
  <cd:country>UK</cd:country>
  <cd:company>CBS Records</cd:company>
```

```
  <cd:price>9.90</cd:price>
  <cd:year>1988</cd:year>
</rdf:Description>
 .
 .
 .
</rdf:RDF>
```

Listing 4.1: RDF Beispiel.

Im Zusammenspiel mit *RDF-Schema (RDF-S)* sind weitere Einschränkungen und Daten-Typisierungen möglich. RDF/RDF-S erlaubt Klassen und Eigenschaften sowie eine Vererbungsbeziehung unter diesen. Damit ermöglicht es die Erstellung einfacher Ontologien.

4.3.2 Die Web Ontology Language

Die *Web Ontology Language (OWL)* [owl04] ist eine Überarbeitung der *DAML+OIL web ontology language* [CvHH$^+$01]. Sie ist entwickelt worden, um Begriffe von Vokabularen und deren Beziehungen untereinander in einer Ontologie auszudrücken. OWL baut auf XML und RDF/RDF-S auf und übertrifft diese an Ausdrucksstärke und Interpretierbarkeit durch Maschinen.

Dabei teilt sich OWL in drei Untersprachen mit jeweils aufsteigender Mächtigkeit auf:

- OWL LITE bietet in erster Linie eine Klassifikationshierarchie und einfache Abhängigkeiten (erlaubt beispielsweise nur die Kardinalitäten 0 und 1).
- OWL DL erlaubt ein Maximum an Ausdrucksstärke, wobei alle Schlussfolgerungen in endlicher Zeit entscheidbar bleiben. OWL DL erlaubt dabei zwar die Verwendung aller OWL-Sprachkonstrukte, erhebt dafür allerdings Einschränkungen. Während eine Klasse zwar Unterklasse von mehreren Klassen sein kann, kann sie nicht Instanz einer anderen Klasse sein. Das „DL“ steht für *Description Logic*, da die Untersprache eine syntaktische Variante der Beschreibungslogik $\mathcal{SHOIN}(D)$ ist (Vgl. [HPS03, HPS04]).
- OWL Full erlaubt ein Maximum an Ausdrucksstärke mit allen Freiheitsgraden von RDF, bietet dafür keine Garantien, was das rechnergestützte Schließen angeht. Beispielsweise kann eine Klasse gleichzeitig als Gruppe von individuellen Klassen behandelt werden und als eigene individuelle Klasse. Es ist unwahrscheinlich, dass Inferenzsysteme für jede Möglichkeit von OWL Full vollständiges Schlussfolgern anbieten können.

Das OWL-Beispiel in Listing 4.2 (entnommen der Website www.lexikon-definition.de/Ontology-Web-Language.html) definiert unter anderem eine Klasse *Woman* als Unterklasse von *Person* und der Einschränkung, dass das Prädikat *gender* nur auf die *Gender*-Klasse

mit dem Wert *female* zeigen darf. Desweiteren werden die Prädikate *gender*, *name* und *firstname* definiert (*domain* spezifiziert die zulässigen Subjekte, die ein Prädikat beschreibt, und *range* die zulässigen Werte, die es annimmt) und eine Objektinstanz von *person* beschrieben für eine Frau namens „Susanne Tilgner“.

```
<rdf:RDF
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:j.0="http://localhost:8080/OWLBuergerInformation.owl#">

   <owl:Class rdf:ID="Gender"/>

   <owl:Class rdf:ID="Woman">
     <rdfs:subClassOf rdf:resource="#Person"/>
     <owl:equivalentClass>
       <owl:Restriction>
         <owl:onProperty rdf:resource="#gender"/>
         <owl:hasValue rdf:resource="#female" rdf:type="#Gender"/>
       </owl:Restriction>
     </owl:equivalentClass>
   </owl:Class>

   <owl:ObjectProperty rdf:ID="gender"
      rdf:type="http://www.w3.org/2002/07/owl#FunctionalProperty">
     <rdfs:range rdf:resource="#Gender"/>
     <rdfs:domain rdf:resource="#Person"/>
   </owl:ObjectProperty>
   <owl:DatatypeProperty rdf:ID="name"
      rdf:type="http://www.w3.org/2002/07/owl#FunctionalProperty">
     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"
        />
     <rdfs:domain rdf:resource="#Person"/>
   </owl:DatatypeProperty>
   <owl:DatatypeProperty rdf:ID="firstname"
      rdf:type="http://www.w3.org/2002/07/owl#FunctionalProperty">
     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"
        />
     <rdfs:domain rdf:resource="#Person"/>
   </owl:DatatypeProperty>

   <j.0:Person rdf:ID="STilgner"
      j.0:firstname="Susanne"
      j.0:name="Tilgner">
     <j.0:gender rdf:resource="#female"/>
   </j.0:Person>
 </rdf:RDF>
```

Listing 4.2: OWL Beispiel.

Markus Krötzsch [Krö05] sieht drei Möglichkeiten, OWL um Regeln zu erweitern:

- Bei einer **Erweiterung von OWL** zur Prädikatenlogik erster Stufe ergibt sich eine klare Semantik und ein homogenes, voll deklaratives System. Außerdem ließe sich dies nahtlos in OWL integrieren. Allerdings bleiben dabei viele Nachteile von OWL weiter bestehen. So lässt sich nicht alles ausdrücken, wie etwa die Regel $(\forall X)(\forall Y)(\forall Z)bruder(Y,Z) \wedge vater(X,Y) \rightarrow onkel(X,Z)$. Auch bleibt OWL so statisch: Es dient weiterhin allein der Repräsentation von Wissen, nicht zur Programmierung. Wichtigster Vertreter dieser Möglichkeit ist die *Semantic Web Rule Language SWRL* (siehe Abschnitt 4.3.3).

- Daneben gibt es die Möglichkeit, **Regeln on top von OWL** zu installieren. Zum Beispiel, indem OWL-Wissensbasen in Logikprogramme eingebunden werden oder durch Anfragen von Logikprogrammen an externe OWL-Reasoner. Klarer Vorteil dabei wäre, dass eine Programmierung durch die Regeln ermöglicht würde. Es entsteht ein so genanntes *hybrides System* (Vgl. [ADG+05]). Den Vorteilen steht allerdings gegenüber, dass so ein heterogenes System mit verschiedenen Semantiken entstünde, dass schließlich unhandlich wäre.

- Die dritte Alternative sind **Regeln neben OWL**. Hier gelingt die Interaktion mit OWL durch ein gemeinsames Fragment, wofür sich etwa F-Logik (siehe Abschnitt 4.2.3) eignen würde. Dabei würde Programmierung in einem homogenen System möglich. Dies erkauft man sich allerdings mit einer unklaren oder sogar unvereinbaren Semantik und einem neuen inkompatiblen Standard.

Da Programmierung bei der Kontextmodellierung nicht entscheidend ist, ist die vorteilhafteste Variante die erste, bei der die Regeln in die Wissensrepräsentation integriert werden. Im Folgenden wird eine Realisierung basierend auf dieser ersten Variante vorgestellt.

4.3.3 Die Semantic Web Rule Language SWRL

Die grundlegende Idee der *Semantic Web Rule Language (SWRL)* ist es, OWL DL um die Möglichkeit zu erweitern, Regeln auszudrücken, und gleichzeitig so wenig wie möglich mit der existierenden Syntax und Semantik von OWL in Konflikt zu geraten. Sprich: SWRL soll abwärtskompatibel zu OWL sein. (Vgl. [ADG+05]) Durch die Kombination mit der unären/binären Regelsprache Datalog RuleML [HBD04] wird OWL derart erweitert, dass die Menge der OWL Axiome auch Hornklauseln umfasst. SWRL Regeln bestehen, wie in Listing 4.3 zu sehen ist, im Grundsatz aus zwei Teilen, dem Antezedenzteil („*body*“) und dem Konsequenzteil („*head*“). Dabei gilt für jede Regel: Wenn die im Antezedenzteil formulierten Bedingungen zutreffen, dann sind auch die im Konsequenzteil gemachten Aussagen wahr.

```
<Imp>
 <body rdf:parseType="Collection">
  [atoms]
 <body>
 <head rdf:parseType:"Collection">
  [atoms]
 <head>
</Imp>
```

Listing 4.3: Rahmen für eine SWRL-Regel.

Die Atome sind jeweils Ausdrücke der Form (Beispiele): $C(x)$, $P(x,y)$, $sameAs(x,y)$, $differentFrom(x,y)$ oder $builtin(r,x,\ldots)$. C ist eine OWL-Klasse, P eine OWL-*Property* und x wie y sind Variablen, OWL-Individuen oder Elemente einer OWL *concrete domain*. *BuiltIns* existieren für Strings, Zahlen, Zeiten, Listen und so weiter.

Ein Problem von SWRL ist, dass durch die Erweiterung von OWL DL die Entscheidbarkeit eingeschränkt wird. Eine unter anderem von Markus Krötzsch [Krö05] propagierte Idee sieht vor, die SWRL-Regeln soweit einzuschränken, dass die Entscheidbarkeit doch wieder garantiert ist. Das heißt: Jede Variable im Konsequenzteil muss auch in einem nicht-DL-Ausdruck im Antezedenzteil auftreten. Ein Beispiel: Statt $onkel(X,Y) \leftarrow bruder(X,Z), vater(Z,Y)$ wird die Regel $onkel(X,Y) \leftarrow bruder(X,Z), vater(Z,Y), O(X), O(Y), O(Z)$ formuliert, damit sie sich nur auf *bekannte* Individuen bezieht. Der Fakt $O(a)$ wird dabei für alle Individuen a definiert.

Daneben gibt es mit der *Rules Lite Concrete Syntax* [BT03] einen Entwurf, eine eingeschränkte XML-Syntax von RuleML zu entwickeln, die voll mit RDF und OWL DL kompatibel ist. Des weiteren hat das Institut für Angewandte Informatik und Formale Beschreibungsverfahren an der Universität Karlsruhe mit KAON2 eine Inferenzmaschine entwickelt, die auf die Mächtigkeit einer Beschreibungslogik eingeschränkte SWRL-Regeln unterstützt [MSS04].

SWRL ist also ein geeigneter Kandidat, Regeln für Kontextinformationen auszudrücken, die in OWL DL modelliert sind, da es DL-sichere Untermengen beziehungsweise Einschränkungen von SWRL bereits gibt und weitere entwickelt werden.

In diesem Kapitel sind Techniken zur Wissensrepräsentation, Wissensmodellierung und Wissensverarbeitung vorgestellt worden. Dies dient als Grundlage für das anschließende Kapitel, in dem eine Modellierungsart für Kontextinformationen als speziellere Informationsart erläutert wird.

5 Modellierung von Kontextinformationen mit CMPlus

> Die Welt ist die Gesamtheit der Tatsachen, nicht der Dinge.
>
> *(Ludwig Wittgenstein)*

In diesem Kapitel wird CMPlus vorgestellt, ein formale Modellierung für Kontextinformationen basierend auf der *Web Ontology Language* (OWL) in deren *Description-Logic*-Variante. Zunächst werden dazu Anforderungen aufgestellt und verwandte Arbeiten betrachtet.

5.1 Motivation

Kontextinformationen werden in offenen Systemen über Dienst- und Domänengrenzen hinweg ausgetauscht. Dies verlangt, dass sich die beteiligten Akteure darüber einig sind, wie sie Kontextinformationen ausdrücken und interpretieren. Wie in jedem Informationsmodell gibt es den Trade-Off zwischen möglichst großer Ausdrucksstärke und möglichst geringer Komplexität. Des bedeutet, dass je mächtiger ein Informationsmodell ist, desto aufwändiger wird dessen Verarbeitung. Im Extremfall können Anfragen nicht mehr berechnet werden.

Im einfachsten Fall ginge es lediglich darum, dass sich zwei Parteien beim Austausch von Kontextinformationen versichern können, bei Anforderung und Lieferung jeweils die selbe Kontextinformation zu meinen. Hierfür würde ein einfacher Katalog von Zuordnungen von Symbolen zu Fakten der realen Welt genügen. Zum Beispiel: „Michaels aktueller Ort“ $= A$, „Temperatur in Raum D6 im Institut für Informatik“ $= B$ und so weiter – ähnlich einer Artikelliste mit Artikelnummern. Hierbei gäbe es im gemeinsamen Modell keine Aussagen über Abhängigkeiten und Beziehungen der Elemente untereinander.

Damit wäre allerdings nur ein minimaler Beschaffungsprozess möglich. „Michaels aktuelle Position“ oder „der aktuelle Ort von Herrn Krause“ würden nicht als äquivalent zu „Michaels aktueller Ort“ erkannt, da die Beschreibungen semantisch nicht auswertbar sind. Es ließe sich auch nicht schließen, dass „die Position von Michaels Mobiltelefon“ äquivalent zu „Michaels aktueller Ort“ ist, solange gilt „Michael trägt sein Mobiltelefon bei sich“. Es bestünde zudem Verwechslungsgefahr mit dem Listeneintrag „Michaels aktueller Ort“, der innerhalb einer anderen Domäne nicht für die Position von Michael Krause, sondern für die von einem Michael Müller steht. Erst mit ausdrucksstärkeren Ontologien werden Eindeutigkeit und automatisches Schließen möglich und können Vererbungs- und

andere Beziehungen ausgedrückt werden. Dies ist unerlässlich, wenn man Kontextinformationen über Domänengrenzen austauschen und wieder verwenden will.

5.2 Die Begriffe des Kontextmodells und der Kontextmodellierung

In der Literatur wird der Begriff „Kontextmodell“ zweideutig gebraucht. Zum einen wird er benutzt als Modell der Kontextinformationen für eine bestimmte Anwendungsart (wie Health Care, Transport, Logistik und so weiter), also Synonym zu „Ontologie“, zum anderen wird er benutzt, um zu beschreiben, wie Kontextinformationen allgemein modelliert werden können (vgl. zum Beispiel „ASC-Modell“ in [Str03]). Besser ist eine Unterteilung wie in [FHKB05], wo zwischen „Kontextmodell“ (für Ontologien) und „Kontextmetamodell“ für die allgemeine Art der Modellierung unterschieden wird. In dieser Arbeit wird ebenfalls der Begriff „Kontextmodell“ nur für Kontextontologien verwendet. Der Begriff „Kontextmetamodell“ wird dagegen nicht übernommen, da nicht alle Modellierungsarten für Kontextinformationen auf der Grundlage eines formalen Metamodells fußen. Stattdessen werden dafür die Begriffe „Kontextmodellierung“ oder „Kontextmodellierungsart“ benutzt.

5.3 Anforderungen an die Modellierung von Kontextinformationen

Kontextinformationen sind Abstraktionen von Charakteristika der realen Welt. Deren Modellierung definiert dafür den Beschreibungsrahmen und bestimmt somit Ausdrucksstärke von Kontextinformationen und deren Möglichkeiten zur Verarbeitung. In diesem Abschnitt werden die Anforderungen an die Modellierung von Kontextinformationen definiert. Auf Basis einer Modellierung können als Ausprägung unterschiedliche Kontextmodelle geschaffen werden. Für Wolfgang Bibel et al. [BHS93] gehören vier Operationen zu jeder Wissensrepräsentation: Das Hinzufügen von Wissen zu bereits gegebenem Wissen, das Entfernen von (etwa überholtem, unbrauchbarem oder falschem) Wissen, das Auffinden von Wissen zu bestimmten Vorgaben und das Ziehen von Schlussfolgerungen aus dem vorhandenen Wissen. Diese Anforderungen sind in die folgenden, spezielleren Anforderungen für eine Kontextmodellierung mit eingegangen.

Grundanforderungen Dies sind allgemeine Anforderungen, die an eine Modellierung von Kontextinformationen gestellt werden.

A1 Die Modellierung von Kontextinformationen muss **generisch** sein. Jede Information, die als Kontextinformation in einem bestimmten Anwendungsbereich relevant sein könnte,

muss damit modellierbar sein. Außerdem muss die Modellierung unterschiedliche Kontextmodelle abhängig vom jeweiligen Anwendungsfall erlauben und damit unterschiedliche Abstraktionen.

A2 Die Modellierung muss geeignet sein, Kontextinformationen **eindeutig** zu beschreiben, so dass Kommunikationspartner, die Kontextinformationen austauschen, diesen dieselbe Bedeutung zuweisen können.

A3 Die Informationen müssen so beschrieben sein, dass sie Grundlage für **automatisches Schließen (Inferenz)** sein können, um implizit vorhandenes Wissen (über Kontextinformationen und deren Beziehungen zueinander) explizit zu machen. Dazu müssen Kontextinformationen Grundlage entscheidbarer, logischer Regeln sein können.

Ausdrucksstärke Diese Anforderungen sagen aus, welche Informationen und Bestandteile eine Kontextinformation enthalten muss. Dabei sind die drei grundlegenden Elemente analog zum Subjekt-Prädikat-Objekt-Tripel (vgl. Kapitel 3) die Entität (als Subjekt), die Kontextinformationsklasse (als Prädikat) und eine Datenstruktur oder eine Entität (als Objekt und damit als „Wert“ der Kontextinformation). Die in B5-B7 geforderten Bestandteile sind optional, was bedeutet, dass eine Kontextinformation auch ohne sie valide sein muss.

B1 Eine **Entität** muss sowohl als Subjekt als auch als Objekt einer Kontextinformation auftreten können und dabei Informationen über ihre Klasse enthalten.

B1.1 Die Identität muss über einen eindeutigen Identifikator verfügen. Je nach Klasse der Entität müssen unterschiedliche Arten von Identifikatoren möglich sein.

B2 Eine Kontextinformation muss aussagen, welcher **Kontextinformationsklasse** sie angehört.

B2.1 Die Modellierung muss Kontextinformationsklassen erlauben, die durch **Parameter** weiter charakterisiert werden können (um zu verhindern, dass für jeden möglichen Parameterwert eine eigene Kontextinformationsklasse definiert werden muss).

B3 Ist das Objekt einer Kontextinformation keine Entität, dann ist es eine typisierte **Datenstruktur**, die den Wert der Kontextinformation benennt. Diese Datenstruktur kann komplex und/oder verschachtelt sein. Sie muss enthalten, welches Repräsentationsformat (Klasse der Datenstruktur) sie darstellt. Die selbe Kontextinformation kann unter Umständen in verschiedenen Repräsentationsformaten ausgedrückt werden.

B4 Durch die Veränderlichkeit von Kontextinformationen ist auch der **Zeitpunkt**, zu dem eine Kontextinformation aktuell ist, notwendiger Bestandteil von ihr.

B5 Es muss möglich sein, einer Kontextinformation Metainformationen anzufügen über die **Qualität** ihres Informationsgehalts (wie Genauigkeit, Abweichung, Fehlerwahrscheinlichkeit und so weiter).

B6 Ebenso muss es möglich sein, Aussagen über die **Quellen** der Kontextinformation hinzuzufügen. Dazu gehört die Angabe des Kontextinformationsdienstes, der die Kontextinformation bereitgestellt hat, wie gegebenenfalls die Angabe, aus welchen anderen Kontextinformationen diese abgeleitet worden ist.

B7 Schließlich muss es auch möglich sein, eine Aussage über die *Wahrscheinlichkeit der Gültigkeit* der Kontextinformation hinzuzufügen.

Organisation der Klassen in Kontextmodellen Diese Anforderungen beziehen sich auf die Mächtigkeit möglicher Kontextmodelle. Darin sind die jeweiligen Entitäts-, Kontextinformations- und Datenstrukturklassen definiert, wie auch die Klassen der Metainformationen.

C1 Mit **Vererbungsbeziehungen** müssen Spezialisierungen modellierbar sein.

C2 Bestandteil eines Kontextmodells müssen auch **Domänen- und Bereichsangaben** der Beziehungen zwischen Klassen sein, um festzulegen, welche Klassen mit welchen verknüpft werden können. (Zum Beispiel welche Datenstrukturklassen für eine bestimmte Kontextinformationsklasse erlaubt sind, oder welche Identifikatoren für eine Entitätsklasse möglich sind.)

C3 Die Möglichkeit, **Äquivalenzbeziehungen** zwischen Klassen auszudrücken, ist ebenfalls notwendig, etwa um Klassen verschiedener Kontextmodelle zu integrieren.

C4 Ziel ist es, ein **Nebeneinander von Kontextmodellen** zu erreichen, die sich im Ganzen thematisch oder durch ihre Anwendungsfälle unterscheiden, in Teilen aber durchaus überlappen können. Jedes Kontextmodell beschreibt einen eigenen für es relevanten Ausschnitt der realen Welt.

C5 Erweiterungen der Kontextmodelle müssen möglich sein, ohne dass Kontextinformationen auf Basis der ursprünglichen Modelle ihre Verwendbarkeit verlieren.

Verwendung der Kontextmodelle Die verschiedenen Kontextmodelle aus D4 besitzen zwar die gleiche Modellierung (*wie* wird etwas beschrieben?), beziehen sich aber auf unterschiedliche Ausschnitte der realen Welt (*was* wird beschrieben?). An die Verwendung dieser Kontextmodelle gibt es Anforderungen und auch an die Einbindung von Kontextmodellen mit andersartiger Modellierung.

D1 Die Kontextinformationen müssen dazu geeignet sein, in einer **Wissensbasis** gespeichert zu werden, die Such-, Einfüge- und Änderungsoperationen sowie Inferenzoperationen ermöglicht und Regeln über die Kontextinformationen enthält.

D2 Die Modellierung muss Grundlage für eine **Kontextanfragesprache** sein können, mit der gezielt eine bestimmte Kontextinformation angefragt werden kann.

D3 Ebenso muss auf Grundlage der Modellierung ein **Kontextinformationsdienst** in der Lage sein, zu veröffentlichen, welche Kontextinformationen er anbieten kann.

D4 Kontextinformationen sollen auch unter Rückgriff auf **verschiedene Kontextmodelle** zu bilden sein. (Etwa, wenn die Entitätsklasse aus einem anderen Kontextmodell stammt als die Kontextinformationsklasse – vgl. D3 und D4).

D5 Soweit möglich sollten selbst Kontextinformationen eingebunden werden können, die Kontextmodellen mit **anderer Kontextmodellierung** genügen („fremde Ontologien“).

Weitere Anforderungen

E1 Es ist von Vorteil, wenn **Tools** generierbar sind, die das Entwickeln und Editieren der Kontextmodelle, das Entwickeln und Ausführen der Regeln oder das Management der Wissensbasen unterstützen.

E2 Die **Übertragung** von Informationen auf Basis des Kontextmodells zwischen Rechnern sollte einfach und schnell sein.

Einige dieser Anforderungen bergen – wie so häufig – einen Zielkonflikt. Vor allem stehen sich die Forderungen nach möglichst großer Ausdrucksstärke (B) und die nach möglichst geringer Komplexität (wegen der Entscheidbarkeit (A3), der einfachen Verarbeitung mit Hilfe von Tools (D1), der schnellen Übertragung (D2) usw.) gegenüber. Hier gilt es, eine geeignete Balance der Erfüllung dieser Anforderungen zu schaffen.

Im Folgenden werden nun bestehende Ansätze zur Kontextmodellierung gegenüber diesen Anforderungen diskutiert.

5.4 Bestehende Ansätze zur Kontextmodellierung

Streng genommen arbeitet jedes Projekt zur Realisierung eines kontextadaptiven Systems mit seinem eigenen Kontextmodell (vgl. 3.4). Dies ist jedoch nicht Sinn dieser Untersuchung. Im Folgenden sind deshalb nur die wichtigsten Arbeiten aufgenommen worden, die auch nach heutigem Stand der Forschung einen Anspruch auf Allgemeinheit erfüllen.

Context Modeling Language (CML)

Karen Henricksen, Jadwiga Indulska und andere [HIM05, IHMM04] erweitern *Object Role Modeling (ORM)* und nennen diese Variante *Context Modeling Language (CML)*. ORM ist eine unter anderem von Terry Halpin entwickelte Methode, Informationen auf konzeptueller Ebene zu analysieren und ist zum Design von Datenbanken gedacht [HB99]. Der größte Unterschied zu UML besteht darin, dass ORM keine Attribute benutzt, sondern diese über Beziehungen modelliert. In großen Teilen können UML-Klassendiagramme

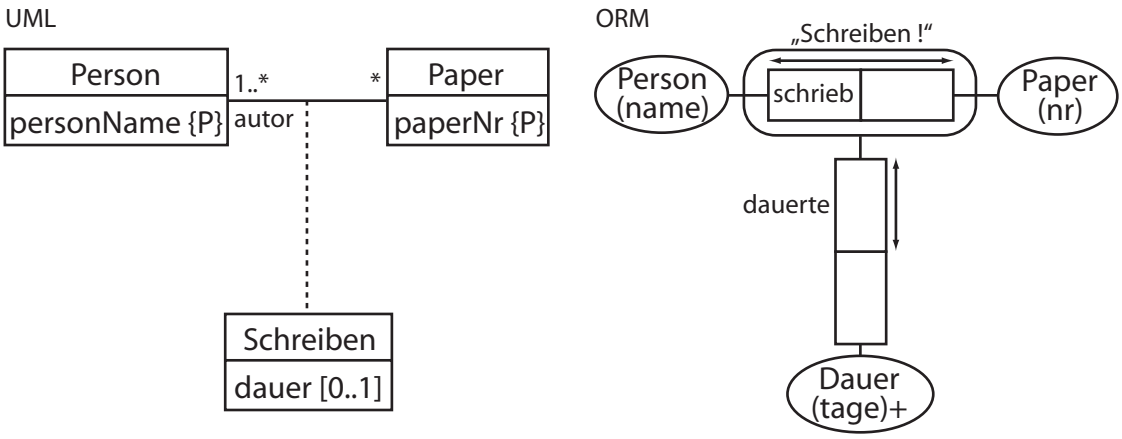


Abbildung 5.1: Unterschiede und Gemeinsamkeiten von ORM und UML nach [HB99].

in ORM übersetzt werden und umgekehrt. Als Beispiel für die strukturellen Gemeinsamkeiten zeigt Abbildung 5.1 den gleichen Sachverhalt in beiden Modellierungssprachen ausgedrückt. Jan Demey und andere haben dazu mit ORM-ML [DJM02] eine XML-Repräsentation erstellt, die ein ORM-Modell abgesehen von der Anordnung der Grafik verlustfrei übersetzen soll. Mit Hilfe von *XSLT stylesheets* sollen auch Übersetzungen in pseudo-natürliche Sprache, Prädikatenlogik oder Regelsprachen möglich sein, wobei die Autoren den Nachweis dafür allerdings schuldig bleiben. Thomas Strang und Claudia Linnhoff-Popien [SLP04] sehen im „graphischen“ Ansatz von CML einen kategorischen Unterschied zu anderen Modellierungsarten, die „Logik-basiert“ oder „Ontologie-basiert“ (gemeint ist hier: auf OWL basierend) seien. Dies ist fragwürdig, spricht doch schon die Überführbarkeit in Prädikatenlogik dafür, dass auch CML strukturell Logik-basiert ist. Im Grunde liegt auch hinter der CML-Modellierung der gleiche Gedanke wie hinter den Subjekt-Prädikat-Objekt-Tripeln von RDF. Wichtigste Eigenart der Faktum-basierten CML ist die, dass modelliert wird, auf welchem Weg eine Kontextinformation gewonnen worden ist. Das erlaubt im Umkehrschluss Aussagen über Gültigkeit und Zuverlässigkeit. So wird unterschieden zwischen statischen (unveränderlichen), profilbasierten, sensorbasierten und abgeleiteten Kontextinformationen (Beispiele für die ersten drei: Abb. 5.2).

Eine der Stärken von CML ist die Mächtigkeit bezüglich der Ausdrucksstärke von Beziehungen. So lassen sich durch Erweiterungen von ORM Qualitätsannotationen modellieren, ebenso die zeitliche Gültigkeit von Kontextinformationen (temporale Fakten), Kontextinformationen, die Primärschlüsseleigenschaften haben, und exklusive Fakten („*alternative Facts*“), die sich gegenseitig ausschließen. Ein Beispiel für Letzteres: Es kann real nur eine gültige Information über den Ort einer Entität geben, während auf Grund von ungenauen oder falschen Messungen durchaus zwei verschiedene Datenwerte existieren können, wobei durch exklusive Fakten gekennzeichnet wird, dass sich diese Daten gegenseitig ausschließen. Die Ausdrucksstärke ist gleichzeitig auch Grund für den größten Nach-

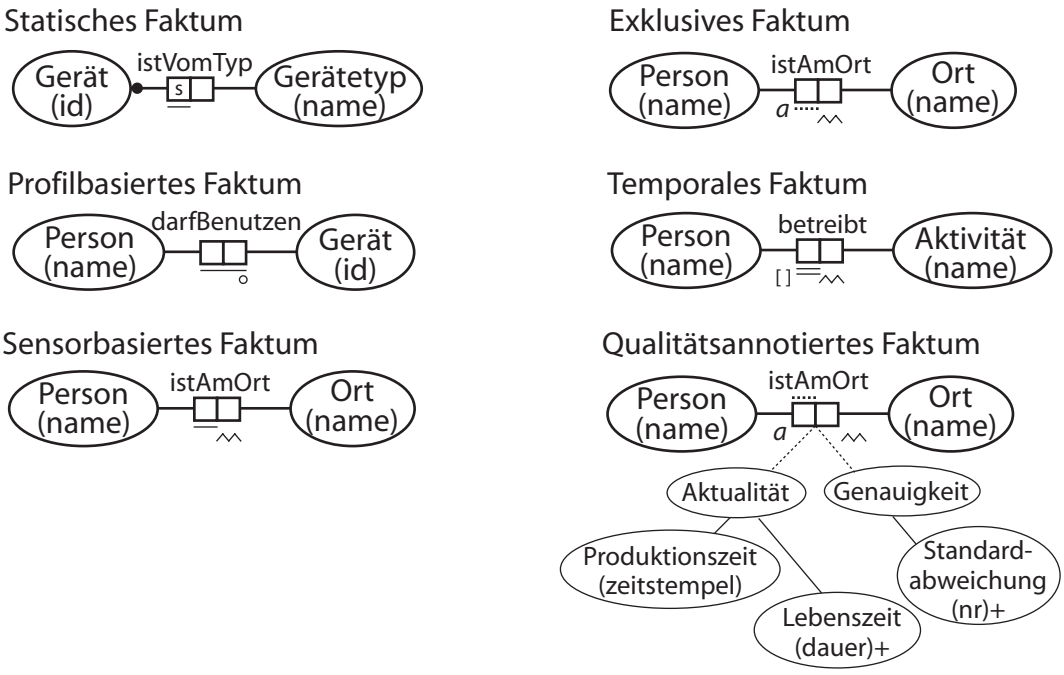


Abbildung 5.2: Beispiele der Faktum-Typen in CML nach [HIM05].

teil von CML. Während ORM nach Aussagen von Demey et al. noch in Prädikatenlogik übersetzt werden kann, scheint das für die zahlreichen ORM-Erweiterungen in CML zumindest fraglich. Aber selbst Prädikatenlogik bietet noch nicht die Entscheidbarkeit und Berechenbarkeit, die von einer Kontextmodellierung gefordert wird.

Es ist anzunehmen, dass mit Hilfe der Vererbungsmechanismen von ORM Klassenhierarchien möglich sind. Inwieweit CML verschiedene Repräsentationsformate (für die Werte von Kontextinformationen genauso wie für die Werte von Qualitätsannotationen) unterstützt und entsprechende Bereichs- und Domänenangaben erlaubt, wird aus den vorliegenden Artikeln nicht deutlich. Zum einfachen Transfer von in CML kodierten Kontextinformationen ist eine XML-Übersetzung oder eine Transformation in ein anderes serialisierbares Format notwendig. Dazu wäre etwa eine entsprechende Erweiterung von ORM-ML notwendig. Es ist nahe liegend, dass ein Kontextmodell, das ohne Übersetzungsschritt XML-Daten erlaubt, gegenüber CML im Vorteil ist.

COntext Ontology (CONON)

Anders als die Entwickler von CML haben Tao Gu et al. [GWPZ04, WGZP04] ihren Ansatz zur Modellierung von Kontext auf die *Web Ontology Language* OWL gestützt und damit direkt auf eine XML-basierte Sprache aufgesetzt. Sie schlagen eine erweiterbare *CONtext ONtology (CONON)* vor, bestehend aus einer General-Ontologie, auf der jede

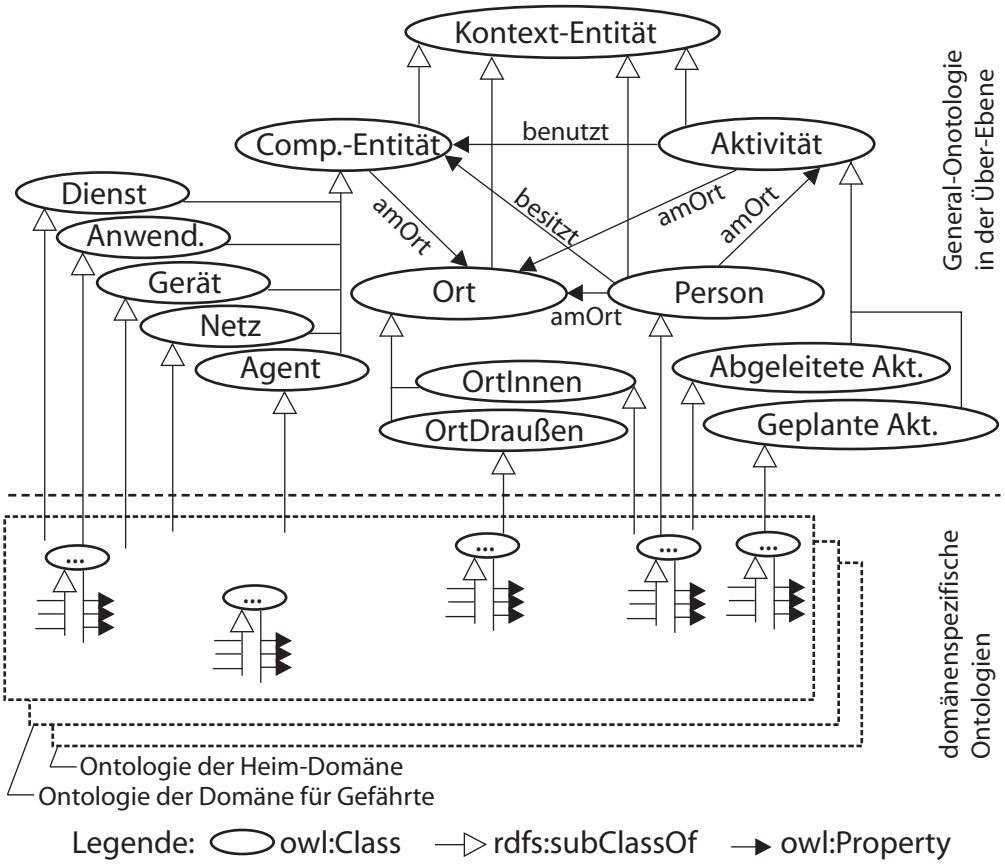


Abbildung 5.3: Die *Upper Ontology* in CONON nach [GPZ05].

domänenspezifische Ontologie (siehe Abbildung 5.3) aufbauen muss. So darf es beispielsweise nur Entitäten geben, die von den Grundentitäten „Person“, „Aktivität“, „Ort“ und „Computer-Entität“ erben. Dies widerspricht der Anforderung dieser Arbeit, generisch jede Art von Ontologie ohne thematische Einschränkung modellieren zu können.

Auch CONON erlaubt Qualitätsannotationen zu Kontextinformationen. Dabei werden sogar mehrere Repräsentationsformate (genannt „Metriken“) erlaubt (vgl. Abbildung 5.4). Dennoch sind damit nur einfache Einheit-Typ-Wert-Tripel erlaubt, und keine komplexen (zusammengesetzten oder verschachtelten) Datenstrukturen.

```
<Person rdf:ID="Wang">
 <locatedIn rdf:resource="#Bedroom" />
</Person>
<Room rdf:ID="Bedroom">
 <locatedIn rdf:resource="#Home" />
</Room>
```

Listing 5.1: OWL-DL-Spezifikation von Hilfskonstrukten für die Identität von Entitäten.

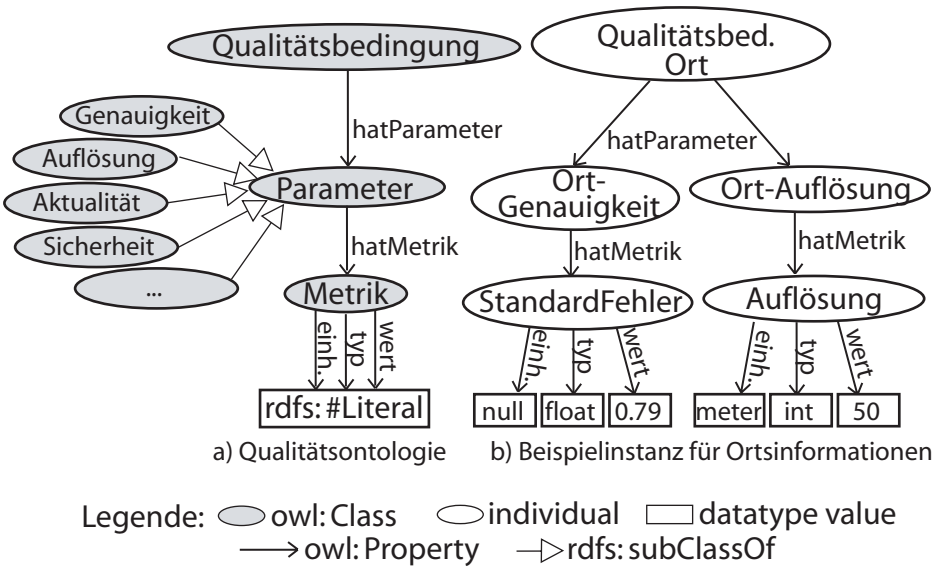


Abbildung 5.4: Qualitätsannotationen in CONON nach [GPZ05].

Listing 5.1 zeigt zwei Kontextinformationen in CONON beschrieben: Eine Person mit der ID „Wand“ befindet sich in einem Raum mit der ID „Home“, der sich wiederum in einer (nicht explizit aufgeführten) Entität mit der ID „Bedroom“ befindet. Hier offenbart sich auch eine Schwäche des Ansatzes. Die Entitäten werden lediglich über einen String für das RDF-Attribut „ID“ identifiziert. Das ist zu wenig ausdrucksstark, um Eindeutigkeit zu sichern oder komplexe Identifikatoren auszudrücken. Es erlaubt auch keinen Rückschluss auf das verwandte Repräsentationsformat.

Eine weiteres Fragezeichen steht bei CONON hinter der Entscheidbarkeit. Zwar legen die Autoren großen Wert darauf, dass auf den mit CONON modellierten Kontextinformationen automatisches Schließen möglich ist, insbesondere das Schließen auf implizites Kontextwissen. Doch der Proof-of-Concept mit einem OWL-Lite Reasoner geschah in einer Variante, die noch keine Qualitätsannotationen erlaubt hat. Solche Annotationen zu OWL-Properties erlaubt nur OWL Full, das allerdings über die Mächtigkeit der entscheidbaren Beschreibungslogiken hinaus geht.

Aspect-Scale-Context-Model (ASC)

Thomas Strang et al. [Str03, SLPF03a, SLPF03b] haben eine Kontextmodellierung entwickelt, deren Kern nach den drei Hauptkomponenten ASC benannt worden ist. Diese drei Komponenten sind „Aspekte“ (entspricht den Kontextinformationsklassen), „Skalen“ (entspricht Repräsentationsformaten) und „Kontextinformationen“. Dabei hat ASC eine stark objektorientierte Sicht. Eine Kontextinformation bei ASC wird beschrieben als Objektinstanz eines Wertes (zum Beispiel *new GaussKruegerCoordinate(„367032“, „533074“)*),

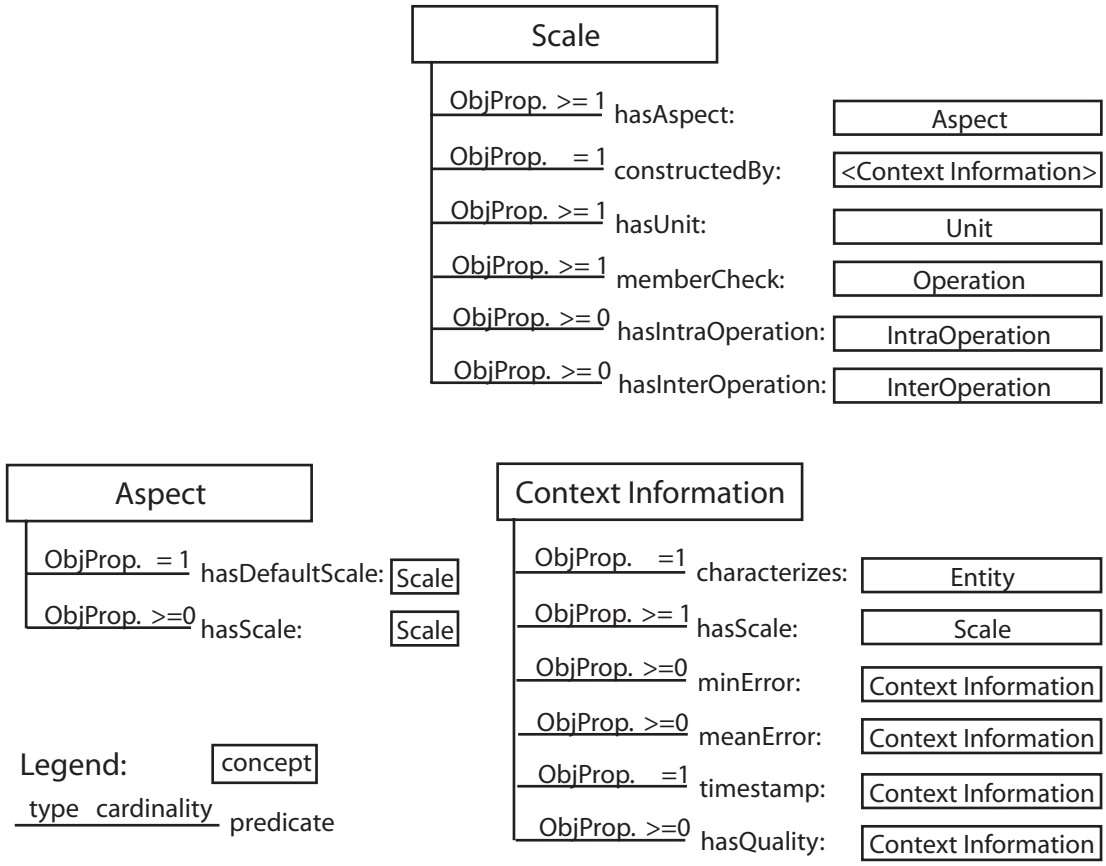


Abbildung 5.5: Die Hauptkomponenten des ASC-Modells nach [SLPF03a].

wobei das Objekt auf die Instanz der Entität zeigt, die sie beschreibt. Gleichzeitig zeigt die Instanz des Aspektes, der beschrieben wird, auf ein Skala-Objekt, das wiederum auf die Kontextinformation zeigt. Abb. 5.5 liefert dazu einen Überblick. Für diesen Kern gibt es eine Umsetzung in OWL DL mit der *Context Ontology Language* (COoL). Von der entsprechenden Darstellung gibt es nur eine skizzenhafte Beschreibung (Abb. 5.2).

```
<WGS84Scale rdf:ID="WGS84ScaleInst">
 <hasAspect rdf:resource="#GeometricPlaceAspectInst" />
 <hasUnit rdf:resource="#WGS84UnitInst" /> <!-- ... -->
</WGS84Scale>

<Unit rdf:ID="WGS84UnitInst">
 <rdfs:label>Lat Lon Alt</rdfs:label>
 <rdfs:comment>Latitude and Longitude in Degree, Altitude in
 m</rdfs:comment>
</Unit>
```

```
<WGS84ContextInformation> <!-- anonymous instance -->
 <characterizes rdf:resource="#SomePersonEntity" />
 <usedByScale rdf:resource="#WGS84ScaleInst" />
 <timestamp rdf:resource="#SomeTimeStampContextInformation" />
</WGS84ContextInformation>
```

Listing 5.2: Exemplarische Skizze einer Kontextinformation nach [Str03].

ASC sieht Verknüpfungen der Kontextinformationen mit Qualitätsinformationen vor, die wiederum (rekursiv) als Kontextinformationen ausgedrückt werden. Parameter kennt ASC nicht, und die Modellierung von Entitäten und ihrer Identitäten wird nicht näher betrachtet – abgesehen davon, dass eine Kontextinformation selbst in den Rang einer Entität gerückt wird, sobald ihr eine Qualitätsinformation beigegeben wird. Thomas Strang stellt in [Str03] auch eine Umsetzung des ASC-Modells in *F-Logic* vor und schreibt, dass es eine Übersetzung nach OWL DL gibt. In ASC modellierte Kontextinformationen genügen nicht den Anforderungen an die Ausdrucksstärke einer Kontextmodellierung.

Augmented World Modeling Language AWML

Nach einem ganz anderen Ansatz wird die Nexus Plattform [Meß01, DHN+04, BBHS03, HNSG05] entwickelt. Dabei soll ein globales Kontextmodell in großem Umfang dadurch entstehen, dass man eine Föderation über kleine „lokale Kontextmodelle“ legt. Dafür werden eine Modellierungssprache (*Augmented World Modeling Language AWML*) und eine Anfragesprache (*Augmented World Query Language*) entwickelt. Nexus soll eine Plattform für kontextsensitive Dienste sein, die auf einem Modell für erweiterte Realität (*Augmented Reality*) basieren. Die erweiterte Realität umfasst reale Objekte (statisch oder mobil) sowie virtuelle (etwa Objekte im WWW). Alle diese Objekte werden in AWML modelliert, wobei sie sich etwa in einem globalen Klassenschema unterordnen müssen, oder eben nur von erweiterten Spezialdiensten behandelt werden können. Hauptanliegen von Nexus ist vor allem, mit Mobilität und Ortsabhängigkeiten umgehen zu können, deshalb werden vor allem geographische Räume und Informationen modelliert. Diese Spezialisierung und Einschränkung genügt allerdings nicht den Anforderungen an die Generik eines allgemeinen Kontextmodells.

```
<awml>
 <scope>
  <ecs name="nexus://nexusschemas.org/ContextCube" is="CC" />
 </scope>
 <nexusobject type="CC:temperatureSensor" NOL="nexus://dvs188:80/
    c0017141-cc15-7141-0001-00603500a570/c0017141-cc157141
    -0100-00603500a570">
  <value>23.3 0C</value>
  <accuracy>+- 1 0C</accuracy>
  <dataSource>sensor</dataSource>
  <type>temperature</type>
```

```
 </nexusobject>
</awml>
```

Listing 5.3: AWML-Beispiel nach [BBHS03].

Abbildung 5.6 und Listing 5.3 zeigen Beispiele für die Darstellung einer Kontextinformation, einmal schematisch, einmal im XML-basierten AWML-Format. In beiden wird deutlich, dass die Ausdrucksstärke nicht den gestellten Anforderungen dieser Arbeit gerecht wird, da zum Beispiel Qualitätsinformationen ohne Angabe eines Repräsentationsformates angegeben werden, oder das Repräsentationsformat der Temperatur mit in den Werte-String eingebettet wird.

Standard Ontology for Ubiquitous and Pervasive Applications SOUPA

Eine verteilte Ontologie zur Unterstützung ubiquitärer Anwendungen haben Harry Chen et al. [CFJ, CPFJ04, Che04c] entwickelt: Die *Standard Ontology for Ubiquitous and Pervasive Applications (SOUPA)* setzt als strukturelle Vorgabe allein auf OWL DL, ohne weitere Spezialisierungen für Kontextinformationen vorzunehmen. Dadurch kennt sie keine allgemeinen Entitäten, Kontextinformations- oder Qualitätsklassen und wird damit zur Kontextmodellierung ohne Kontextmodell. Dahinter steckt die Meinung, dass sich allein durch den Gebrauch der Ontologie und die Weiterentwicklung von kontextsensitiven Diensten und ihren Anforderungen mit der Zeit von selbst die sinnvollen Ontologiebestandteile heraus mendeln werden.

Ganz im Sinne dieses *best-practise*-Ansatzes haben Chen und seine Kollegen für SOUPA eine Reihe bereits eingeführter Ontologien aus verschiedenen Fachgebieten integriert. Um bessere Interoperabilität zu unterstützen und den Aufwand für automatisches Schließen so gering wie möglich zu halten, sind diese Ontologien nicht direkt importiert worden. Vielmehr sind ihre Begriffe für neue OWL-DL-Ontologien verwandt worden, die wiederum durch OWL-Abbildungskonstrukte auf die Originalkonzepte in den fremden Vorbildontologien zeigen. So ist der normative SOUPA-Kern entstanden (siehe Abbildung 5.7), der je nach Anwendungsgebiet um weitere, spezielle Ontologien erweitert werden kann.

```
<owl:Class rdf:ID="LocationContextOfHarry">
 <owl:intersectionOf rdf:parseType="Collection">
  <owl:Class rdf:about="&loc;LocationContext" />
  <owl:Restriction>
   <owl:onProperty rdf:resource="&loc:locationContextOf" />
   <owl:hasValue>
    <per:Person rdf:about="http://umbc.edu/people/hchen4" />
   </owl:hasValue>
  </owl:Restriction>
 </owl:intersectionOf>
</owl:Class>
```

Listing 5.4: Ausschnitt aus einer exemplarischen SOUPA-Ontologie nach [CPFJ04].

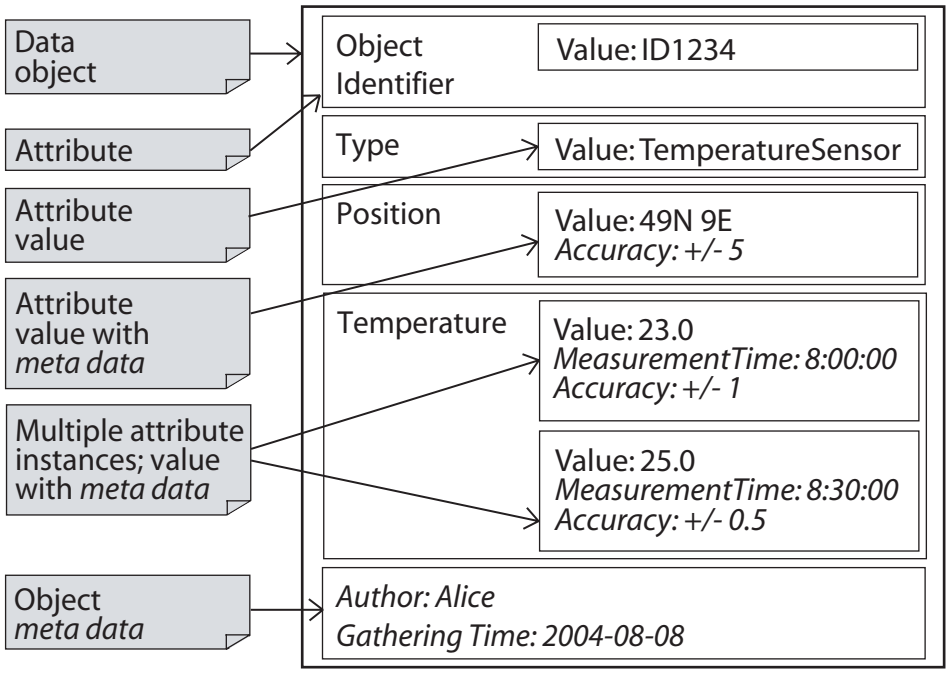


Abbildung 5.6: Schema einer NEXUS-Kontextinformation nach [HNSG05].

Listing 5.4 zeigt einen Ausschnitt aus einer Ontologie, in der eine Klasse „LocationContextOfHarry“ definiert wird, indem die Klasse „LocationContext“ direkt mit einem URI verknüpft wird, der Harry Chen identifiziert. Durch seine OWL DL-Basis ist SOUPA zwar im Prinzip für automatisches Schließen geeignet, doch durch das Fehlen eines Kontextmodells und die dadurch bedingte Strukturfreiheit (innerhalb OWL DL) stellt sich die Frage, ob daraus sinnvolle Schlüsse mit vertretbarem Aufwand gezogen werden können. Damit wird SOUPA nicht den Ansprüchen hinsichtlich der Ausdrucksstärke und der Organisation der Klassen gerecht. Auch der normative SOUPA-Kern widerspricht einer Anforderung, nämlich der Generik.

Weitere Ansätze

Abgesehen von diesen Ansätzen zur Kontextmodellierung gibt es weitere, meist ältere Ansätze, die allerdings nicht als Kandidaten für eine Kontextmodell in Frage kommen, da sie meist die nach Meinung der jeweiligen Autoren wichtigsten Kontextkategorien auflisten, ohne formale Grundlage zur Modellierung zu geben. Darunter fallen die *Context Categories* von Anind Dey et al. [DSA01], ebenso wie das *Working Model for Context* von Albrecht Schmidt et al. [SBG99] und die *Context Atoms* von Jani Mäntyjärvi [Män03]. Daneben gibt es unter anderem die XML-Sprache ConteXtML von Nick Ryan [Rya99], die allerdings, was Generik, formale Grundlage und Eignung für automatisches Schließen

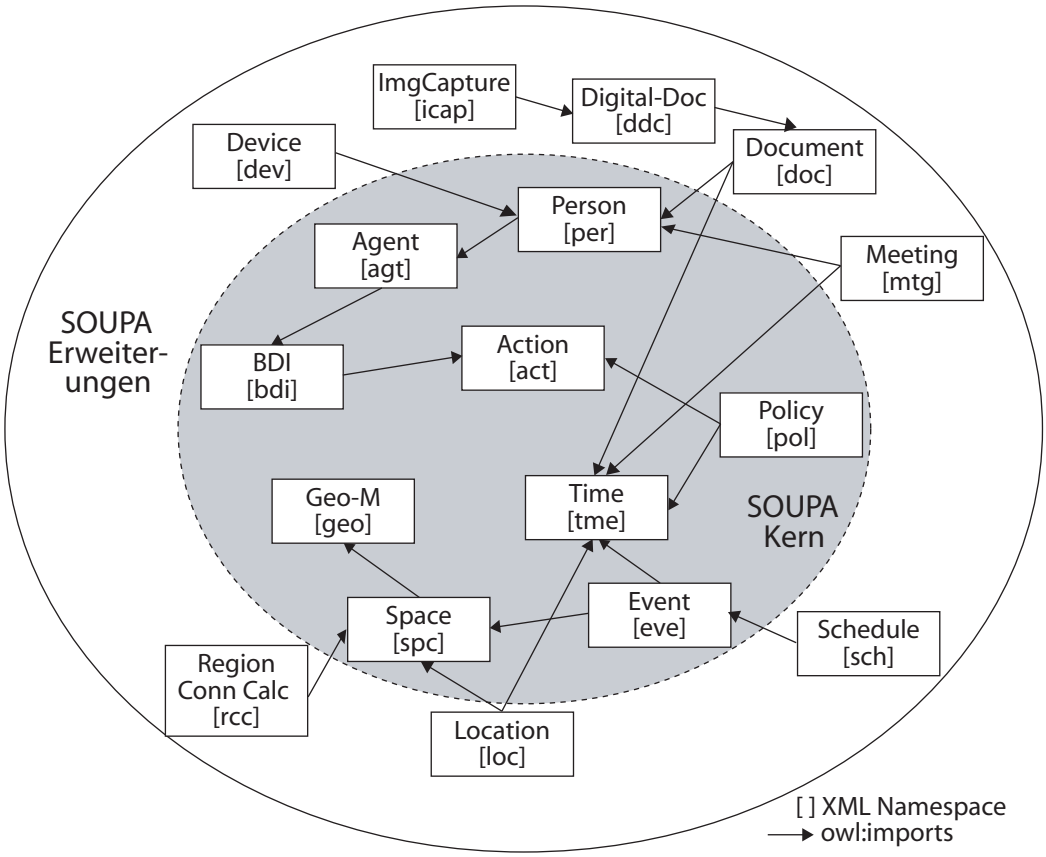


Abbildung 5.7: Der SOUPA-Kern und Erweiterungen nach [CPFJ04].

anlangt, weit weniger geeignet ist als OWL-basierte Ansätze. Des weiteren gibt es einen Ansatz von Albert Held et al. [HBS02], die die *Composite Capability/Preference Profiles Language (CC/PP)* [ccp04] zu *Comprehensive Structured Context Profiles (CSCP)* erweitert haben. Da sie den notwendigen Kontext auf die Funktionalität mobiler Geräte, die Charakteristiken des Netzzugangs sowie „nutzerspezifische“ Informationen beschränken, ist auch dieses zu eingeschränkt für einen allgemeinen Ansatz.

Bewertung

In Tabelle 5.1 sind die wichtigsten Ansätze zur Kontextmodellierung gegen die in Abschnitt 5.3 aufgestellten Anforderungen evaluiert worden. Es zeigen sich Defizite hinsichtlich der Generik bei Modellen, die eine *Upper Ontology* vorschreiben. Generell haben alle Ansätze kein Konzept, wie Entitäten so modelliert werden können, dass sie auch eindeutig und praktikabel identifiziert werden. Weitere Schwächen sind die Darstellung der Kontextwerte, die Möglichkeit zur Inferenz oder die Mittel zur Integration fremder Modelle. Kein Modell kann parametrisierte Kontextinformationen darstellen. Die Möglichkeit, die

Herkunft einer Kontextinformation zu modellieren, beschränkt sich meist auf den informationsliefernden Dienst oder gar nur auf die Art der Quelle – nicht aber auf zu Grunde liegende Kontextinformationen. Bis auf CONON sehen die Modelle keine direkte Kodierung des Grades der Unsicherheit einer Information vor.

Besser sieht es dagegen bei der Anwendbarkeit der Modelle aus: Durch ihre XML-Repräsentation eignen sie sich im Prinzip alle zur Bildung einer Wissensbasis, oder dazu, Grundlage einer Anfragesprache oder einer CIS-Beschreibung zu sein. Wobei letzteres natürlich nur eingeschränkt gelten kann, wenn die Ausdrucksstärke des Modells generell zu schwach ist – wie etwa bei SOUPA, dem (abgesehen von der OWL-DL-Kodierung) überhaupt kein spezielles Kontextmodell zu Grunde liegt.

Bei der Wahl eines Kontextinformationsmodells für den Bereitstellungs- und Beschaffungsprozess von Kontextinformationen gibt es zwei Alternativen: Entweder, sich für einen bestehenden Ansatz zu entscheiden und diesen eventuell zu erweitern, um den Anforderungen zu genügen, oder einen neuen Ansatz zu entwickeln. Diese Arbeit verfolgt die zweite Alternative, zum einen um sicherzugehen, dass das Ergebnis sich immer noch eignet, Grundlage automatischen Schließens zu sein. Zum anderen, da die Defizite der jeweiligen Kandidaten doch zu groß sind. Übernommen wird allerdings das Vorgehen, sich auf OWL DL zu stützen, das schon bei CONON, SOUPA und auch bei der Übersetzung von ASC angewandt wird. Ein *Upper-Ontology*-Ansatz wird dagegen nicht verfolgt, um das Ergebnis so allgemein und generisch wie möglich zu halten. Der eigene Modellierungsansatz namens CMPlus wird in dem folgenden Abschnitt vorgestellt.

5.5 CMPlus für Kontextontologien

Zur Modellierung von Kontextinformationen schlägt diese Arbeit ein Konzept namens CMPlus vor, das in diesem Abschnitt erläutert wird. Die Vorarbeiten dazu waren in dem *Context Meta Model (CMM)* [FHKB05, Fuc04] gemündet, das nun noch um Merkmale wie Identität von Entitäten, Ableitungsbeziehungen oder Parameter erweitert wird. Abbildung 5.8 gibt eine erste, stark vereinfachte Übersicht über die drei Abstraktionsebenen von CMPlus:

- Auf Instanzebene wird aktuelles Kontextwissen in einer Wissensbasis gespeichert. Im Beispiel sind dies zwei Kontextinformationen: Eine nicht näher bezeichnete Person benutzt ein Gerät der Klasse PDA und hat einen Puls von 2,1 Hertz (126 Herzschläge pro Minute).
- Auf Modellebene werden die Klassen definiert, die im konkreten Fall instanziiert werden können. Auch Abhängigkeiten zwischen den Klassen können hier hinterlegt sein. Ein Modell ist in einer Ontologie zusammengefasst.
- Der abstrakte Kern der Modellierung legt mit seinen Assoziations- und Aggregationsbeziehungen die Grundstruktur für alle CMPlus-Modelle und Wissensbasen fest. Die Klassen der Modellebene sind Unterklassen der Klassen des abstrakten Kerns.

Tabelle 5.1: Bewertung, wie die wichtigsten Kontextmodellierungen die in Abschnitt 5.3 aufgestellten Anforderungen erfüllen. Dort, wo keine Aussage getroffen werden konnte (meist wegen fehlender Informationen), steht ein „o“.

	CML	CONON	ASC	NEXUS	SOUPA
A1 Generik	+	–	+	–	–
A2 Eindeutigkeit	+	+	+	+	+
A3 Inferenz	-	+/–	+	o	+/–
B1 Entität	+	++	+/–	+	++
B1.1 Identifikator	–	–	–	–	–
B2 Kontextinformationsklasse	+	+	+	+	–
B2.1 Parameter	–	–	–	–	–
B3 Datenstruktur	+/–	+	+/–	+/–	–
B4 Zeitstempel	+	+	+	+	–
B5 Qualitätsattribute	+/–	+	+	+/–	–
B6 Herkunft	+/–	+/–	–	+/–	+/–
B7 Wahrscheinlichkeit	–	+	–	–	–
C1 Vererbung	+	+	+	+	+
C2 Assoziations-Domänen und -Bereiche	o	+	+	–	–
C3 Äquivalenz	+/-	o	–	o	+/–
C4 Nebeneinander von Kontextmodellen	o	+	+	+	+
C5 Erweiterbarkeit	o	+	+	+	+
D1 Wissensbasis	+	+	+	+	+/–
D2 Anfragesprache	+/–	+	+	+	+/–
D3 CIS-Beschreibung	+/–	+	+	+	+/–
D4 Verknüpfung verschiedener Modelle	o	+	+	+	+
D5 Integration fremder Modellierungen	–	–	–	–	+
E1 Toolunterstützung	+/–	+	+	+/–	+
E2 Übertragung	–	+	+	+	+

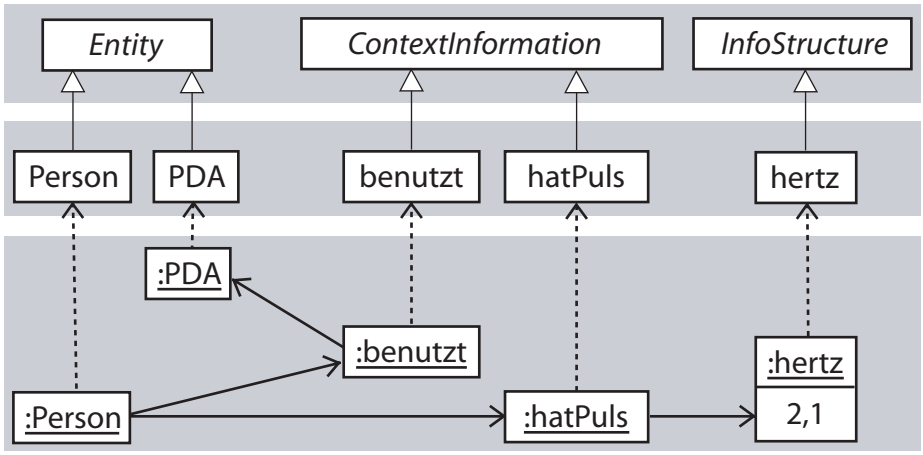


Abbildung 5.8: Stark vereinfacht: Zusammenhang zwischen Instanzebene, Modellebene und Modellierung.

Im Kern besteht die Kontextmodellierung in CMPlus aus der Idee von Subjekt, Prädikat und Objekt, wie sie auch in den RDF-Tripeln verwirklicht worden ist. Subjekt einer Kontextinformation ist immer eine Entität. Das Prädikat bezeichnet, *was* über diese Entität ausgesagt wird (im Beispiel von Abbildung 5.8 sind die Prädikate „*usesDevice*“ oder „*hasPulse*“). Das Objekt wiederum gibt den Wert des Prädikats an. Dies kann nun abhängig vom Prädikat ein echter Datenwert sein (im Beispiel 2,1 Hertz) oder eine Entität, sofern das Prädikat eine Beziehung beschreibt (im Beispiel ein bestimmtes Gerät, das benutzt wird). Möglich sind also die Aussagen-Tripel der Art Entität-Kontextinformation-Entität (im Beispiel „Person“-„hat Puls“-„2,1 hertz“) oder der Art Entität-Kontextinformation-Datenstruktur (im Beispiel „Person“-„benutzt“-„PDA“). Dies zeigt auch Abbildung 5.9.

Im Folgenden werden nun die wichtigsten Merkmale von CMPlus erläutert, bevor die Spezifikationen für die drei Ebenen dargelegt werden. Dabei wird zunächst die Modellierung von CMPlus erklärt, dann exemplarisch auf die Modellebene und die Instanzebene heruntergebrochen.

Kontextinformations-Datenstruktur

Ist das Objekt einer Kontextinformation ein Datenwert, kann dieser in verschiedenen Formaten mit eventuell wechselndem Informationsgehalt repräsentiert werden. Die Kontextinformation „Temperatur“ (etwa einer Festplatte) könnte unter anderem ausgedrückt werden in Celsius, Fahrenheit oder Kelvin. Die Kontextinformation „Position“ (etwa einer Person) könnte unter anderem ausgedrückt werden als logische Adresse mit Straße, Hausnummer, Postleitzahl und Ort oder als physische Adresse aus Längen- und Breitengraden. Während sich im ersten Beispiel die Formate sehr ähneln und auch den selben Informationsgehalt tragen, sind die Formate im zweiten sowohl in der Struktur als auch im Informationsgehalt sehr verschieden.

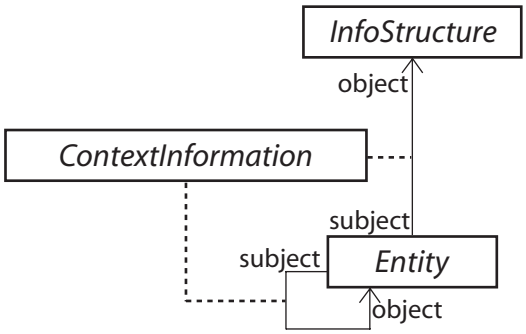


Abbildung 5.9: Grundsätzliche Zusammenhänge zwischen den Komponenten des abstrakten CMPlus-Kerns.

Identität von Entitäten

Anonyme Instanzen von Entitäten, wie die der Klasse „Person“ in Abbildung 5.8 genügen nicht als Teil einer Kontextinformation: Wie will man wissen, ob die Kontextinformation sich tatsächlich auf die Entität bezieht, zu der ein kontextsensitiver Dienst Kontextwissen benötigt? Die Darstellung von Entitäten benötigt eindeutige Identifikatoren, die diese auch außerhalb einer Wissensbasis eindeutig identifizieren.

Optimal wäre, modellübergreifend die gleiche Art von Identifikatoren für alle Arten von Entitäten verwenden zu können, etwa nach der Art der *Uniform Resource Identifier (URI)* [UPIG01] für Ressourcen im Internet. Dies scheitert allerdings daran, dass es modellübergreifend nicht möglich ist, den Begriff der Entität näher einzugrenzen. Was in einem Modell eine Entität ist, kann in einem anderen Modell aus mehreren Entitäten bestehen oder in einer Teilmengenbeziehung zu Entitäten aus weiteren Modellen bestehen. Was eine Entität ist, bestimmen letztlich nur die Anwendungsfälle eines Modells und denen sind keine Grenzen gesetzt. Strenge hierarchische Strukturen wie im Internet sind also nicht möglich. Es gilt:

- Was ein möglicher Identifikator für eine Entitätsklasse ist, hängt ab vom jeweiligen Modell und damit vom Anwendungsfall.
- Es kann mehrere Identifikatoren für eine Entitätsklasse geben.
- Identifikatoren können temporär sein (zum Beispiel eine über DHCP bezogene IP-Adresse als Identifikator für einen bestimmten Rechner).
- Identifikatoren können selbst Kontextinformationen sein.

In CMPlus sind Identifikatoren deshalb Kontextinformationsklassen, die im Modell als „identitätsstiftend“ ausgezeichnet worden sind. Durch den Zeitbezug von Kontextinformationen wird das Problem der temporären Identifikatoren umgangen. Eine Entität kann mehrere verschiedenartige Identifikatoren haben, die in verschiedenen Formaten ausgedrückt

werden können – auch das ist durch die Modellierung als Kontextinformation möglich. Es bleiben zwei Forderungen des Modells: Jede Entität muss mindestens eine identitätsstiftende Kontextinformation besitzen, um überhaupt dargestellt werden zu können. Und eine identitätsstiftende Kontextinformation kann nur auf einen Datenwert, nicht auf eine Entität zeigen.

Parameter von Kontextinformationsklassen

Eine Kontextinformationsklasse „istInDerNäheVon“, die eine Entfernungsbeziehung zwischen zwei Entitäten ausdrückt, wäre nicht formal genug, solange „Nähe“ nicht enger definiert werden würde. Eine Kontextinformationsklasse „istNäherAlsAchtMeter“ wäre zwar formal genug (wenn der Name der Klasse auch der Semantik der Definition entspricht), aber nicht praktikabel. Für jeden Anwendungsfall, der einen unterschiedlichen Nähe-Begriff hat, müssten eigene Klassen vorrätig gehalten werden: „istNäherAlsSiebenMeter“, „istNäherAlsSiebenMeterFünfzig“, „istNäherAlsEinKilometer“ oder ähnliche.

Aus diesem Grund gibt es in CMPlus die Möglichkeit, parametrisierbare Kontextinformationsklassen zu definieren: „istNäherAls(x)“, wobei x eine Entfernung bezeichnet.

Qualitätsmerkmale von Kontextinformationen

Wie in Abschnitt 2.1.3 beschrieben, sind als Basis zur Auswahl und Bewertung von Kontextinformationen Metainformationen notwendig (zum Beispiel zu Genauigkeit und Ursprung). Deshalb sieht das Modell vor, Kontextinformationen mit Metadaten zu annotieren. Damit die Informationen über den Ursprung von Kontextinformationen auch nach Transformationen und Aggregationen nicht verloren gehen, gibt es die Möglichkeit, Kontextinformationen mit denjenigen Kontextinformationen zu assoziieren, aus denen sie hervorgegangen sind.

Auch über diese Qualitätsinformationen können auf der Metaebene Aussagen getroffen werden, etwa über die Wahrscheinlichkeit von deren Gültigkeit. So wird aus der zu einer Ortsangabe gehörenden Qualitätsinformation „Die Maximalabweichung beträgt 10 Meter“ etwa „Die Wahrscheinlichkeit, dass die Maximalabweichung nicht größer als 10 Meter ist, beträgt 97 Prozent“. Theoretisch ließe sich diese Rekursion endlos fortsetzen, was die Modellierung zwar grundsätzlich zulassen würde. Tatsächlich scheint es nicht praktikabel, dies über die Rekursionsstufe „Qualitätsmerkmal eines Qualitätsmerkmals einer Kontextinformation“ hinaus zu tun.

Zeitstempel

Da viele Kontextinformationen sehr kurzlebig sind, ist eine Kontextinformation ohne Zeitstempel sinnlos. Wie in Abschnitt 3.3.1 erläutert, beinhaltet auch eine Kontextanfrage den Zeitpunkt, auf den sich die Information beziehen soll. In CMPlus ist deshalb ein Zeitstempel (als ein Qualitätsmerkmal) notwendige Annotation für jede Kontextinformation.

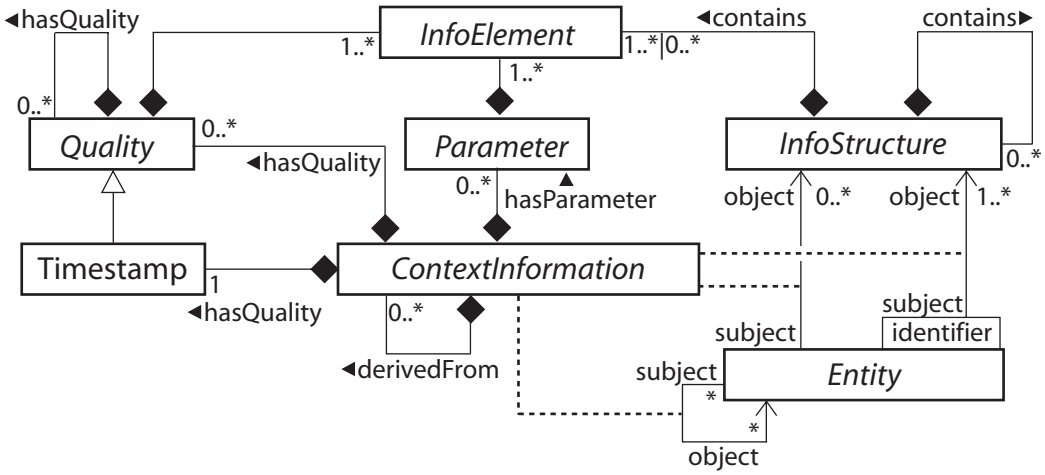


Abbildung 5.10: Diese UML-Darstellung gibt einen ausführlicheren Überblick über den Kern des CMPlus-Modells zur Kontextmodellierung. Bis auf „Timestamp“ sind alle Klassen abstrakt und daher nicht instanziierbar. Die Realisierung des Modells in OWL DL folgt allerdings nicht den UML-Konstrukten.

Datenwerte und Datentypen

CMPlus erlaubt für die eben angeführten Datenkonstrukte Container, die ein Element oder mehrere Elemente der Form Bezeichner-Datentyp-Wert enthalten und eventuell weitere Container. So können rekursiv komplexe Datenstrukturen aufgebaut werden.

Äquivalenzbeziehungen

Ontologien beinhalten ein Modell eines ganz bestimmten Ausschnitts der Welt (zum Beispiel „HealthCare“, „Transportation“ usw.). Das bedeutet nicht, dass es nicht Klassen gibt, die in verschiedenen Ontologien Äquivalentes modellieren, zum Beispiel die selbe Entitätsklasse oder Kontextinformationsklasse. Deshalb gibt es in CMPlus ein Konstrukt, Äquivalenzen zwischen Klassen auszudrücken.

Damit sind alle grundlegenden Konstrukte des abstrakten CMPlus-Modells erläutert. Abbildung 5.10 gibt einen graphischen Überblick über die logischen Zusammenhänge, die sich aber von der tatsächlichen Spezifikation in OWL DL unterscheiden (beispielsweise ist die Subjekt-Assoziation für identitätsstiftende Kontextinformationen nicht als ausgezeichnete Rolle der Assoziation sondern als Unterklasse der Assoziation realisiert). Trotzdem ist diese Darstellungsweise als UML-Diagramm gewählt worden, da sie die anschaulichste ist.

5.5.1 CMPlus Modellierungsebene in OWL DL

Das in diesem Abschnitt behandelte abstrakte Modell von CMPlus gibt es als OWL-Dokument herunter zu laden unter www.mobile.ifi.lmu.de/~krausem/owl/CMPlus.owl.

Die Kernklassen „Entity“, „ContextInformation“, „InfoStructure“ und ihre Assoziationen

Die scheinbar nahe liegende Möglichkeit, den Tripel (Entität, Kontextinformation, Datenwert) beziehungsweise den Tripel (Entität, Kontextinformation, Entität) in OWL DL zu spezifizieren, wäre, das analog zu RDF zu machen. Dann wäre die Kontextinformation eine *OWL property*, die die Entität entweder mit einer Entität oder einem Datenwert verknüpft. Das Problem dabei ist jedoch, dass die Kontextinformation bei CMPlus mit Metainformationen verknüpft ist. In der Konsequenz bedeutet das, dass eine *OWL property* mit anderen *OWL properties* assoziiert wird – dies erlaubt OWL DL allerdings nicht.

Die Lösung besteht darin, die Kontextinformation (siehe Listing 5.5) selbst als OWL-Klasse zu modellieren, genauso wie auch Entität und Datenwert. Dies sind die Klassen „ContextInformation“, „Entity“ und „InfoStructure“. Subjekt- und Objektklasse sind dann über OWL *properties* („ci_subject“ beziehungsweise „ci_object“) mit „ContextInformation“ verknüpft (siehe Listing 5.6). Invers zu der von der „ContextInformation“ ausgehenden *property* „ci_subject“ hin zur „Entity“ gibt es die *property* ”´ci“ ausgehend von der „Entity“ hin zur zur „ContextInformation“. Diese Verknüpfung ist zwar logisch redundant, erleichtert aber später das Navigieren durch die Instanzen.

```
<owl:Class rdf:ID="Entity">
 <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing" />
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#
       int">1</owl:minCardinality>
   <owl:onProperty rdf:resource="#ci_id" />
  </owl:Restriction>
 </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="ContextInformation">
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="#ci_subject" />
   <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
       >1</owl:cardinality>
  </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
  <owl:Restriction>
```

```
   <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
   <owl:onProperty rdf:resource="#ci_object" />
  </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="#q_Timestamp" />
   <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
  </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="#ci" />
   <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
  </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing" />
</owl:Class>

<owl:Class rdf:ID="InfoStructure">
 <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing" />
</owl:Class>
```

Listing 5.5: OWL DL Spezifikation der abstrakten CMPlus-Klassen 'Entity', 'ContextInformation' und 'InfoStructure'.

Die OWL *restriction* in der Klassendefinition von „Entity“ bedeutet, dass eine Entität mit mindestens einer identitätsstiftenden Kontextinformation verknüpft ist. Damit ist gewährleistet, dass jede Entität eine Identität besitzt. Außerdem gibt ein Schalter in der Kontextinformationsklasse an, ob sie identitätsstiftend ist oder nicht.

Es gibt in OWL zwei vordefinierte Identifikatoren für alle Klassen: Die Expansion von „owl:Thing“ umfasst alle Individuen, wodurch diese Oberklasse aller OWL-Klassen ist. Dagegen ist die Expansion von „owl:Nothing“ die leere Menge, wodurch diese Klasse Unterklasse jeder OWL-Klasse ist. Die explizite Angabe in CMPlus, dass alle Klassen Unterklassen von „owl:Thing“ sind, ist damit eigentlich redundant. Sie wurde beibehalten zur Verdeutlichung einerseits, und andererseits, um mit dem Ontologie-Editor Protégé [GMF+02] kompatibel zu bleiben, mit dem die Sourcen erstellt und validiert worden sind (Protégé Version 3.1).

```
<owl:ObjectProperty rdf:ID="ci">
 <owl:inverseOf rdf:resource="#ci_subject" />
 <rdfs:domain rdf:resource="#Entity" />
 <rdfs:range rdf:resource="#ContextInformation" />
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:ID="ci_subject">
 <rdfs:range rdf:resource="#Entity" />
 <rdfs:domain rdf:resource="#ContextInformation" />
 <owl:inverseOf rdf:resource="#ci" />
</owl:ObjectProperty>

<owl:FunctionalProperty rdf:ID="ci_object">
<rdfs:domain rdf:resource="#ContextInformation" />
<rdfs:range>
<owl:Class>
<owl:unionOf rdf:parseType="Collection">
<owl:Class rdf:about="#Entity" />
<owl:Class rdf:about="#InfoStructure" />
</owl:unionOf>
</owl:Class>
</rdfs:range>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty" /
    >
</owl:FunctionalProperty>
```

Listing 5.6: OWL DL Spezifikation der abstrakten CMPlus-Assoziationen 'subject' und 'object'.

Die Kardinalitätseinschränkung, dass jede Entität mindestens eine identitätsstiftende Kontextinformation besitzen muss, wird erreicht durch die Hilfsdefinitionen in Listing 5.7. Jede Instanz einer Kontextinformationsklasse, deren *DatatypeProperty* „foundsIdentity“ auf den booleschen Wert 1 zeigt, ist damit auch Instanz der Unterklasse „IDContext-Information“. Eine Subjekt-Assoziation ist damit auch eine „IDsubject“-Assoziation, die wiederum mit der gewünschten Kardinalitätseinschränkung in der „Entity“-Spezifikation eingeschränkt ist.

```
<owl:ObjectProperty rdf:ID="ci_id">
 <rdfs:range rdf:resource="#IDContextInformation" />
 <owl:inverseOf rdf:resource="#ci_id_subject" />
 <rdfs:subPropertyOf>
  <owl:ObjectProperty rdf:about="#ci" />
 </rdfs:subPropertyOf>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="ci_id_subject">
 <owl:inverseOf rdf:resource="#ci_id" />
 <rdfs:domain rdf:resource="#IDContextInformation" />
 <rdfs:subPropertyOf>
  <owl:ObjectProperty rdf:about="#ci_subject" />
 </rdfs:subPropertyOf>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:ID="ci_id_object">
 <rdfs:range rdf:resource="#InfoStructure" />
 <rdfs:domain rdf:resource="#IDContextInformation" />
 <rdfs:subPropertyOf>
  <owl:FunctionalProperty rdf:about="#ci_object" />
 </rdfs:subPropertyOf>
</owl:ObjectProperty>
```

Listing 5.7: OWL DL Spezifikation von Hilfskonstrukten für die Identität von Entitäten.

„InfoElement“ zur Repräsentation von Datenwerten

Nicht-komplexe Datenwerte werden in CMPlus als „InfoElement“ realisiert, einer Unterklasse der OWL *Datatype-Property*. „InfoElement“ selbst ist dabei abstrakt – Ontologien müssen die Spezialisierungen erst festlegen, die dann instanziiert werden können. Datenwerte werden von „InfoStructure“-, „Quality“- und „Parameter“-Klassen getragen, wie die „InfoElement“-Definition in Listing 5.8 zeigt. Der Datentyp ist entweder ein XML-Schema-Datentyp oder ein „rdfs:Literal“. Laut OWL-Spezifikation muss jedes Inferenzsystem für OWL DL mindestens mit den XML-Schema-Datentypen „xsd:integer“ und „xsd:string“ umgehen können.

Qualitäts- und Parameterklassen werden über „hasQuality“- und „hasParameter“ mit der Kontextinformationsklasse assoziiert. Die „InfoStructure“-Klasse kann als einzige auch mit sich selbst assoziiert sein und so verschachtelte Datenstrukturen bilden.

Der notwendige Zeitstempel einer Kontextinformation (siehe Listing 5.5) ist eine instanziierbare Unterklasse der Qualität. Ähnlich wie bei der Identität von Entitäten ist eine Assoziation mit einer „Timestamp“-Qualität eine Unterklasse von „hasQuality“ namens „hasTimestamp“. Für diese besteht eine Kardinalitätseinschränkung innerhalb der Kontextinformation: Es muss genau eine geben.

```
<owl:ObjectProperty rdf:ID="hasParameter">
 <rdfs:range rdf:resource="#Parameter" />
 <rdfs:domain rdf:resource="#ContextInformation" />
 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
     FunctionalProperty" />
</owl:ObjectProperty>

<owl:FunctionalProperty rdf:ID="hasQuality">
 <rdfs:domain>
  <owl:Class>
   <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#ContextInformation" />
    <owl:Class rdf:about="#Quality" />
   </owl:unionOf>
  </owl:Class>
 </rdfs:domain>
```

```
 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"
     />
 <rdfs:range rdf:resource="#Quality" />
</owl:FunctionalProperty>

<owl:FunctionalProperty rdf:ID="contains">
 <rdfs:domain rdf:resource="#InfoStructure" />
 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"
     />
 <rdfs:range rdf:resource="#InfoStructure" />
</owl:FunctionalProperty>

<owl:DatatypeProperty rdf:ID="InfoElement">
 <rdfs:domain>
  <owl:Class>
   <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#InfoStructure" />
    <owl:Class rdf:about="#Parameter" />
    <owl:Class rdf:about="#Quality" />
   </owl:unionOf>
  </owl:Class>
 </rdfs:domain>
 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
     FunctionalProperty" />
</owl:DatatypeProperty>

<owl:Class rdf:ID="Parameter">
 <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing" />
</owl:Class>

<owl:Class rdf:ID="Quality">
 <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing" />
</owl:Class>

<owl:FunctionalProperty rdf:ID="q_Timestamp">
 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"
     />
 <rdfs:subPropertyOf rdf:resource="#hasQuality" />
 <rdfs:range rdf:resource="#Timestamp" />
</owl:FunctionalProperty>

<owl:Class rdf:ID="Timestamp">
 <rdfs:subClassOf rdf:resource="#Quality" />
</owl:Class>
```

Listing 5.8: OWL-DL-Spezifikation aller CMPlus-Klassen, die Datenwerte tragen.

Historie durch „derivedFrom“

Die OWL-*ObjectProperty* „derivedFrom“ realisiert eine Assoziation von Kontextinformationen mit anderen Kontextinformationen, aus denen sie hervorgegangen sind. So ist es möglich, eine Historie einer Kontextinformation zu erstellen, wie sie zum Beispiel für die Bewertung einer Kontextinformation wichtig ist. Listing 5.9 enthält die Definition.

```
<owl:ObjectProperty rdf:ID="derivedFrom">
 <rdfs:domain rdf:resource="#ContextInformation" />
 <rdfs:range rdf:resource="#ContextInformation" />
</owl:ObjectProperty>
```

Listing 5.9: OWL-DL-Spezifikation der 'derivedFrom'-Assoziation.

Qualität „hasQuality“ von Qualitätsinformationen

Wie in Listing 5.8 zu sehen ist, kann nicht nur „ContextInformation“ sondern auch „Quality“ selbst über die *ObjectProperty* „hasQuality“ mit einer Qualitätsinformation annotiert werden.

Vererbungs- und Äquivalenzbeziehungen

RDF-S sieht mit seiner „subClassOf“-*Property* bereits die Möglichkeit vor, Vererbungsbeziehungen zu modellieren. CMPlus übernimmt diese direkt. Gleiches gilt im Prinzip auch für die „equivalentClass“-*Property* von OWL, die es ermöglicht, Äquivalenzen zwischen Klassen auszudrücken. Mit ihr können etwa Klassen aus verschiedenen Ontologien als gleichwertig bezeichnet werden. Hier ist es für CMPlus allerdings notwendig, sicherzustellen, dass nur gleichartige Klassen miteinander verbunden werden. Deshalb werden die Spezialisierungen „equivalentEntity“, „equivalentContextInformation“, „equivalentIDContextInformation“, „equivalentInfoStructure“, „equivalentQuality“ und „equivalentParameter“ in CMPlus eingeführt, die entsprechende Einschränkungen tragen (siehe exemplarisch in Listing 5.10. Sie sind nur als Beziehungen zwischen Klassen zu verwenden, nicht zwischen Individuen.

```
<owl:ObjectProperty rdf:ID="equivalentEntity">
 <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#
    equivalentClass"/>
 <rdfs:range rdf:resource="#Entity" />
 <rdfs:domain rdf:resource="#Entity" />
</owl:ObjectProperty>
```

Listing 5.10: Äquivalenzen zwischen Klassen in CMPlus am Beispiel der Entitätsklassen.

5.5.2 CMPlus Modellebene in OWL DL

Auf Basis von CMPlus lassen sich Kontext-Ontologien zu den verschiedensten Themenbereichen erstellen, die sich dank der Fähigkeiten von XML (wie etwa einheitliche Identifikatoren und Namensräume) einfach integrieren, erweitern und verknüpfen lassen. Diese Ontologien entsprechen den *TBoxes* der Beschreibungslogiken (vgl. Abschnitt 4.2.3), definieren also das Vokabular.

Im Folgenden werden Ausschnitte aus einer einfachen Beispielontologie benutzt, die es unter www.mobile.ifi.lmu.de/~krausem/owl/Example1_Ontology.owl zu laden gibt. Listing 5.11 zeigt einen Ausschnitt davon, wobei „cmp“ hier wie in den kommenden Ausschnitten für den Namensraum des CMPlus Metamodells steht. Hier werden die Entitätsklassen „Man“ und „Woman“ als Unterklasse von „Person“ und die Entitätsklasse „MobilePhone“ als Unterklasse von „Device“ definiert.

Zur Namenskonvention sei dabei angemerkt:

- Namen von Klassen beginnen mit einem Großbuchstaben.
- Assoziationen (*properties*), die auf Qualitäten zeigen, beginnen mit dem Präfix „q_“.
- Assoziationen, die auf Parameter zeigen, beginnen mit dem Präfix „p_“.
- Die Datenassoziationen („InfoElement“) von den Qualitäten, Parametern und Datenstrukturen der Kontextinformationsklassen beginnen mit dem Präfix „ie_“.
- Die Namen der Assoziationen einer Kontextinformationsklasse setzen sich zusammen aus dem Präfix (dieses wird bestimmt durch die jeweilige Oberklasse) „ci_“, „ci_id_“, „ci_object_“, „ci_id_object_“, „ci_subject_“ oder „ci_id_subject_“ und dem Namen der Kontextinformationsklasse.

```
<owl:Class rdf:ID="Person">
 <rdfs:subClassOf rdf:resource="cmp:Entity"/>
</owl:Class>

<owl:Class rdf:ID="Man">
 <rdfs:subClassOf>
  <owl:Class rdf:about="#Person"/>
 </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="Woman">
 <rdfs:subClassOf rdf:resource="#Person" />
</owl:Class>

<owl:Class rdf:ID="Device">
 <rdfs:subClassOf rdf:resource="cmp:Entity"/>
</owl:Class>
```

```
<owl:Class rdf:ID="MobilePhone">
 <rdfs:subClassOf rdf:resource="#Device" />
</owl:Class>
```

Listing 5.11: Dieser Ausschnitt einer CMPlus-Ontologie zeigt die Definition von Entitätsklassen.

Die größere Mächtigkeit, die CMPlus-Ontologien dadurch erlangen, dass sie Kontextinformationen in OWL nicht als Verknüpfung sondern als eigenständige Klasse modellieren, ist erkauft durch eine größere Komplexität der Informationskonstrukte. Dieser Mehraufwand wird deutlich, wenn Kontextinformationsklassen definiert werden wie in Listing 5.12. Für jede Kontextinformationsklasse müssen die zugehörigen OWL-*properties* ebenfalls vererbt werden, da diese dann die Informationen tragen, welche Klassen Objekt und Subjekt der Kontextinformation sein können.

Mit der Kardinalitätsbeschränkung auf der Assoziation „hasParameter“ wird angegeben, wie viele Parameter diese Kontextinformation hat. Im aufgelisteten Beispiel hat die Kontextinformation „Name“ keinen Parameter. Sie ist eine identitätsstiftende Kontextinformationen, da ihre Konstrukte von den entsprechenden „ID“-Unterklassen erben.

```
<owl:Class rdf:ID="Name">
 <rdfs:subClassOf rdf:resource="cmp:IDContextInformation"/>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="cmp:hasParameter"/>
   <owl:cardinality rdf:datatype="xsd:int">0</owl:cardinality>
  </owl:Restriction>
 </rdfs:subClassOf>
</owl:Class>

<owl:ObjectProperty rdf:ID="ci_id_Name">
 <rdfs:subPropertyOf rdf:resource="cmp:ci_id"/>
 <rdfs:domain rdf:resource="#Person"/>
 <rdfs:range rdf:resource="#Name"/>
 <owl:inverseOf rdf:resource="#ci_id_subject_Name" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="ci_id_subject_Name">
 <rdfs:subPropertyOf rdf:resource="cmp:ci_id_subject"/>
 <rdfs:domain rdf:resource="#Name"/>
 <rdfs:range rdf:resource="#Person"/>
 <owl:inverseOf rdf:resource="#ci_id_Name"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="ci_id_object_Name">
 <rdfs:subPropertyOf rdf:resource="cmp:ci_id_object"/>
```

```
 <rdfs:domain rdf:resource="#Name"/>
 <rdfs:range rdf:resource="#NameString"/>
</owl:ObjectProperty>
```

Listing 5.12: Dieser Ausschnitt einer CMPlus-Ontologie zeigt die Definition einer (in diesem Fall identitätsstiftenden) Kontextinformationsklasse mit den dazu gehörenden Assoziationen.

Wie in Listing 5.12 zu sehen ist, zeigt die Kontextinformation „Name“ nicht auf eine Entität, sondern auf eine Datenstruktur „NameString“. (Eine Entität als Objekt einer identitätsstiftenden Kontextinformation wäre auch gar nicht erlaubt.) Die Definition dieser Datenstruktur zeigt Listing 5.13. Die Datenstruktur besitzt zwei Datenassoziationen „ie_firstname“ und „ie_lastname“ und keinen weiteren Verschachtelungen (da die Eigenschaft „contains“ auf die Kardinalität „0“ beschränkt wird). Diese Datenassoziation könnte von anderen „InfoStructure“-Klassen ebenfalls verwendet werden. Im vorliegenden Beispiel ist dies jedoch nicht der Fall, da die Domäne dieser Datenassoziation auf die „NameString“-Klasse beschränkt ist.

```
<owl:Class rdf:ID="NameString">
 <rdfs:subClassOf rdf:resource="cmp:InfoStructure"/>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="cmp:InfoElement"/>
   <owl:cardinality rdf:datatype="xsd:int">2</owl:cardinality>
  </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="#ie_lastname" />
   <owl:cardinality rdf:datatype="xsd:int">1</owl:cardinality>
  </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:cardinality rdf:datatype="xsd:int">0</owl:cardinality>
   <owl:onProperty rdf:resource="cmp:contains"/>
  </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:cardinality rdf:datatype="xsd:int">1</owl:cardinality>
   <owl:onProperty rdf:resource="#ie_firstname" />
  </owl:Restriction>
 </rdfs:subClassOf>
</owl:Class>

<owl:DatatypeProperty rdf:ID="ie_lastname">
```

```
 <rdfs:subPropertyOf rdf:resource="cmp:InfoElement"/>
 <rdfs:domain rdf:resource="#NameString"/>
 <rdfs:range rdf:resource="xsd:string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="ie_firstname">
 <rdfs:subPropertyOf rdf:resource="cmp:InfoElement"/>
 <rdfs:domain rdf:resource="#NameString"/>
 <rdfs:range rdf:resource="xsd:string"/>
</owl:DatatypeProperty>
```

Listing 5.13: Dieser Ausschnitt einer CMPlus-Ontologie zeigt die Definition einer Kontextinformations-Datenstruktur („InfoStructure“) zur Kodierung eines Namens mit Vor- und Nachname.

Andere Kontextinformationsklassen in dieser Beispielontologie sind dagegen durchaus parametrisiert, etwa die Kontextinformationsklasse „IsNearby“, die Nähebeziehungen zwischen Entitäten ausdrückt. Welche Entfernung dabei noch als „nah“ verstanden wird, bestimmt der Parameter „WithinDistance“, der in dem Ontologieausschnitt in Listing 5.14 definiert wird. Er wird über „p_WithinDistance“ mit „IsNearby“ assoziiert und besitzt genau eine Datenassoziation „InfoElement“, nämlich „ie_meter“. Letzteres wird durch die beiden Kardinalitätseinschränkungen so festgelegt: Dass es genau einen Parameter „ie_meter“ gibt, bewirkt die Kardinalität von „ie_meter“ selbst. Dass es keine weitere Datenassoziation geben kann, bewirkt die Kardinalität „1“ auf „InfoElement“. Die Datenassoziation „ie_meter“ wird übrigens auch von der Qualitätsklasse „PositionTolerance“ benutzt, wie ihrer Domäne zu entnehmen ist.

```
<owl:Class rdf:ID="WithinDistance">
 <rdfs:subClassOf rdf:resource="cmp:Parameter"/>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="cmp:InfoElement"/>
   <owl:cardinality rdf:datatype="xsd:int">1</owl:cardinality>
  </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
  <owl:Restriction>
  <owl:onProperty rdf:resource="#ie_meter" />
  <owl:cardinality rdf:datatype="xsd:int">1</owl:cardinality>
 </owl:Restriction>
 </rdfs:subClassOf>
</owl:Class>

<owl:ObjectProperty rdf:ID="p_WithinDistance">
 <rdfs:subPropertyOf rdf:resource="cmp:hasParameter"/>
 <rdfs:domain rdf:resource="#IsNearby"/>
 <rdfs:range rdf:resource="#WithinDistance"/>
```

```
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:about="#ie_meter">
 <rdfs:subPropertyOf rdf:resource="cmp:InfoElement"/>
 <rdfs:range rdf:resource="xsd:float"/>
 <rdfs:domain>
  <owl:Class>
   <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#WithinDistance"/>
    <owl:Class rdf:about="#PositionTolerance"/>
   </owl:unionOf>
  </owl:Class>
 </rdfs:domain>
</owl:DatatypeProperty>
```

Listing 5.14: Dieser Ausschnitt einer CMPlus-Ontologie zeigt Parameter von Kontextinformationen.

Beispiel für einen Qualitätsparameter einer Kontextinformation ist „PositionTolerance“, der angibt, wie stark die tatsächliche Position von der angegebenen maximal abweichen kann. Wie in der Domäne der entsprechenden Assoziation in Listing 5.15 aufgeführt kann dieser Qualitätsparameter sowohl auf die Kontextinformationen der Klasse „IsNearby“ als auch auf die der Klasse „Position“ angewandt werden.

```
<owl:Class rdf:ID="PositionTolerance">
 <rdfs:subClassOf rdf:resource="cmp:Quality"/>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:cardinality rdf:datatype="xsd:int">0</owl:cardinality>
   <owl:onProperty rdf:resource="cmp:hasQuality"/>
  </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:cardinality rdf:datatype="xsd:int">1</owl:cardinality>
   <owl:onProperty rdf:resource="cmp:InfoElement"/>
  </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="#ie_meter" />
   <owl:cardinality rdf:datatype="xsd:int">1</owl:cardinality>
  </owl:Restriction>
 </rdfs:subClassOf>
</owl:Class>

<owl:ObjectProperty rdf:ID="q_PositionTolerance">
 <rdfs:subPropertyOf rdf:resource="cmp:hasQuality"/>
```

```
 <rdfs:domain>
  <owl:Class>
   <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Position"/>
    <owl:Class rdf:about="#IsNearby"/>
   </owl:unionOf>
  </owl:Class>
 </rdfs:domain>
 <rdfs:range rdf:resource="#PositionTolerance"/>
</owl:ObjectProperty>
```

Listing 5.15: Dieser Ausschnitt einer CMPlus-Ontologie zeigt Qualitätsannotationen.

5.5.3 CMPlus Instanzebene in OWL DL

Nachdem die Kontextontologie das Vokabular definiert, umfasst die CMPlus-Instanzebene das aktuelle Kontextwissen, was der *assertional box* der Beschreibungslogik entspricht. Passend zur der in Abschnitt 5.5.2 vorgestellten Beispielontologie zeigt dieser Abschnitt nun exemplarisch, wie das entsprechende Kontextwissen selbst dargestellt wird. Abbildung 5.11 gibt einen Überblick über das Wissen, das die Instanzebene enthält. Es gibt drei Entitäten, einen Mann, eine Frau und ein Mobiltelefon, wobei Mann und Frau durch ihren Namen und das Mobiltelefon durch einen ID-String identifiziert werden, jeweils ausgedrückt als identitätsstiftende Kontextinformation. In Listing 5.16 ist exemplarisch davon nur der Name des Mannes aufgeführt: „Bob Jones“. Das gesamte Dokument der Beispiel-Wissensbasis gibt es herunter zu laden unter www.mobile.ifi.lmu.de/~krausem/owl/Example1_Knowledgebase1.owl.

```
<Man rdf:ID="Man_38">
 <ci_id_Name rdf:resource="#Name_39" />
 <ci_IsNearby rdf:resource="#IsNearby_53" />
 <ci_Knows rdf:resource="#Knows_51" />
 <ci_Position rdf:resource="#Position_56" />
</Man>

<Woman rdf:ID="Woman_43">
 <ci_id_Name rdf:resource="#Name_44" />
</Woman>

<MobilePhone rdf:ID="MobilePhone_47">
 <ci_id_DeviceID rdf:resource="#DeviceID_48" />
 <ci_Position rdf:resource="#Position_62" />
</MobilePhone>

<Name rdf:ID="Name_39">
 <ci_id_subject_Namerdf:resource="#Man_38"/>
 <ci_id_object_Name>
  <NameString rdf:ID="NameString_41">
```

```
   <ie_lastname rdf:datatype="xsd:string">Bob</ie_lastname>
   <ie_firstname rdf:datatype="xsd:string">Jones</ie_firstname>
  </NameString>
 </ci_id_object_Name>
 <cmp:q_Timestamp>
  <cmp:Timestamp rdf:ID="Timestamp_40">
   <ie_time rdf:datatype="xsd:dateTime">2005-10-27T18:31:00</ie_time>
  </cmp:Timestamp>
 </cmp:q_Timestamp>
</Name>
```

Listing 5.16: Dieser Ausschnitt einer Wissensbasis zeigt in CMPlus kodierte Entitäten.

Exemplarisch zeigt Listing 5.17 eine Kontextinformation die besagt: Das Mobiltelefon ist in der Nähe von Bob. Diese Information wird charakterisiert durch einen Parameter „p_WithinDistance“, der angibt, dass eine Entfernung von maximal 50,0 Metern noch als „Nähe“ verstanden werden soll. Dazu gibt die Qualitätsinformation „q_PositionTolerance“ eine Toleranzgrenze von 5,0 Metern an. Diese Kontextinformation ist abgeleitet aus zwei Kontextinformationen über die Position (des Mobiltelefons und von Bob), die nicht in dem Listing aufgeführt sind.

```
<IsNearby rdf:ID="IsNearby_53">
 <ci_subject_IsNearby rdf:resource="#Man_38" />
 <ci_object_IsNearby rdf:resource="#MobilePhone_47" />
 <p_WithinDistance>
  <WithinDistance rdf:ID="WithinDistance_54">
   <ie_meter rdf:datatype="xsd:float">50.0</ie_meter>
  </WithinDistance>
 </p_WithinDistance>
 <q_PositionTolerance>
  <PositionTolerance rdf:ID="PositionTolerance_60">
   <ie_meter rdf:datatype="xsd:float">5.0</ie_meter>
  </PositionTolerance>
 </q_PositionTolerance>
 <cmp:q_Timestamp>
  <cmp:Timestamp rdf:ID="Timestamp_55">
   <ie_time rdf:datatype="xsd:dateTime">2005-10-27T19:43:00</ie_time>
  </cmp:Timestamp>
 </cmp:q_Timestamp>
 <cmp:derivedFrom rdf:resource="#Position_56" />
 <cmp:derivedFrom rdf:resource="#Position_62" />
</IsNearby>
```

Listing 5.17: Dieser Ausschnitt einer Wissensbasis zeigt eine Kontextinformation in CMPlus.

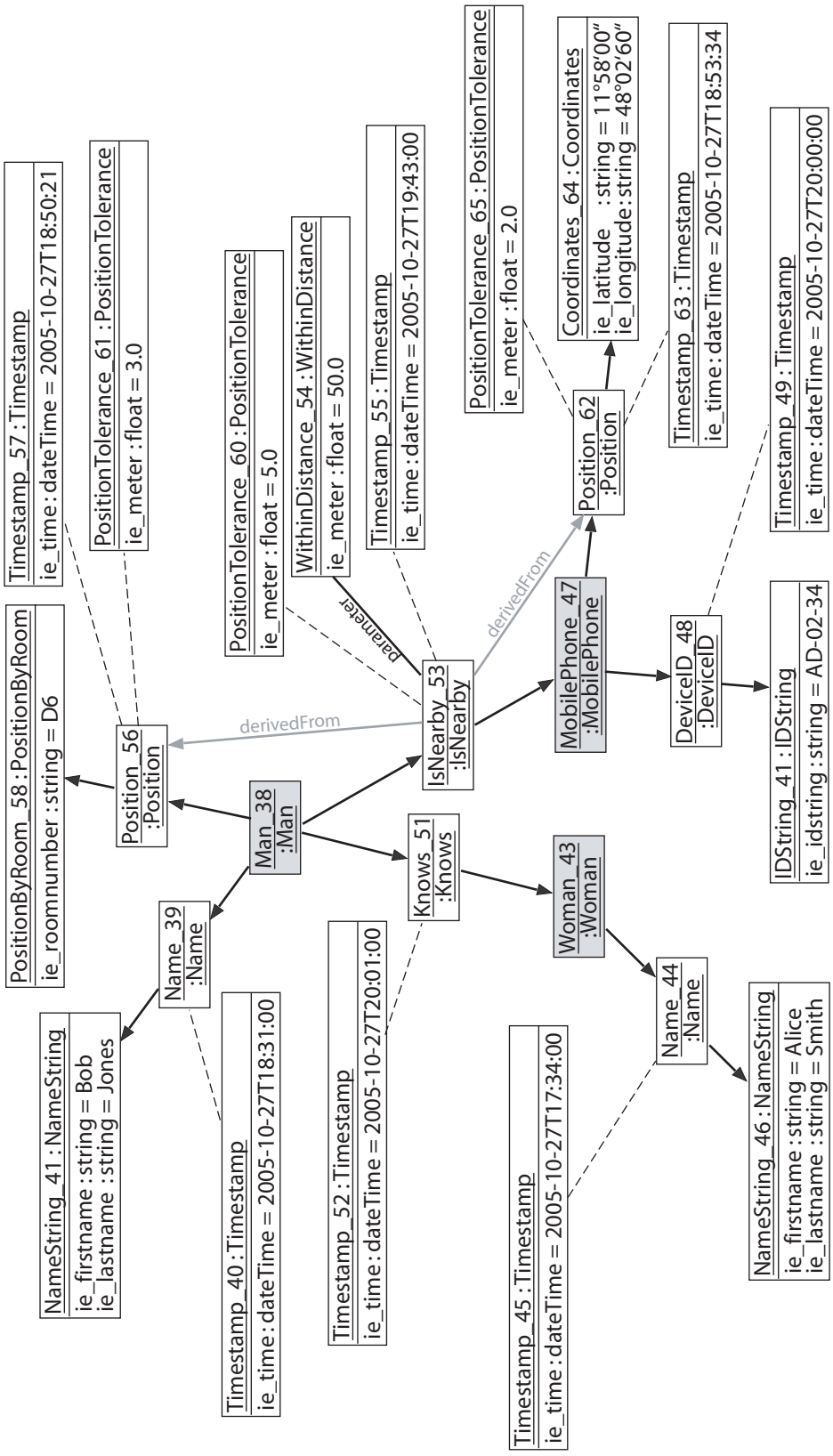


Abbildung 5.11: Grafische Übersicht des Inhalts einer Wissensbasis, die CMPlus-kodierte Kontextinformationen speichert.

5.6 Bewertung

In den vorangegangenen Abschnitten ist die Kontextmodellierung mit CMPlus erläutert worden, nämlich wie der abstrakte Kern von CMPlus aussieht, wie damit Ontologien modelliert werden können, und wie eine dazu gehörende Wissensbasis aussieht. Im Folgenden soll nun der Ansatz gegenüber den in Abschnitt 5.3 aufgestellten Anforderungen evaluiert werden, soweit dies an dieser Stelle bereits möglich ist. Soweit es um die Anwendung von CMPlus und CMPlus-Ontologien geht, wird auf die entsprechenden folgenden Kapitel verwiesen.

A1 Generik Dies wird voll erfüllt, da der abstrakte Kern von CMPlus keinerlei Einschränkungen bezüglich der mit ihm zu modellierenden Klassen gibt. Einzige Ausnahme ist die Qualitätsklasse „Timestamp“, die aus Gründen der einheitlichen Verarbeitung vorgegeben wird, aber weitere Modellierungen von Zeitangaben daneben erlaubt.

A2 Eindeutigkeit Wird durch Formalität und Ausdrucksstärke von CMPlus möglich, es liegt aber zum Teil in der Verantwortung der Ontologieentwickler, die Klassen entsprechend zu entwerfen, um Zweideutigkeiten zu vermeiden.

A3 Inferenz Wird dadurch erfüllt, dass die Modellierung sich allein auf OWL-DL-Konstrukte stützt. Auch für die Regeln ist eine OWL-DL-sichere Untermenge von SWRL gewählt worden. Was die Formulierung von Regeln angeht, sei auf das Kapitel 7 verwiesen.

B1 Entität Die Modellierung von Entitäten ist möglich.

B1.1 Identifikator Eine identitätsstiftende Kontextinformation als Identifikator zu benutzen (einer der größten Vorteile gegenüber bestehenden Modellierungen) verspricht, in zweierlei Hinsicht günstig zu sein: Zum einen steht damit die gesamte Ausdrucksstärke von Kontextinformations-Datenkonstrukten zur Verfügung, zum anderen müssen können für deren Verarbeitung bereits bestehende Mechanismen genutzt werden, was die Handhabbarkeit des Modells unterstützt.

B2 Kontextinformationsklasse Die Modellierung von Kontextinformationsklassen wird ermöglicht.

B2.1 Parameter Kontextinformationsklassen sind parametrisierbar. Dies gibt es nicht in den anderen Ansätzen zur Kontextmodellierung.

B3 Datenstruktur Die Modellierung der Datenstrukturen entspricht den Anforderungen.

B4 Zeitstempel Wird voll erfüllt durch die entsprechende Spezialisierung eines Qualitätsattributs.

B5 Qualitätsattribute Wird erfüllt.

B6 Herkunft Wird durch Qualitätsklassen erfüllt, die Aussagen über den liefernden Kontextinformationsdienst geben und in den Ontologien näher zu spezifizieren sind. Außerdem verweist die „*derivedFrom*"-Beziehung zu den zu Grunde liegenden Kontextinformationen.

B7 Wahrscheinlichkeit Dies wird ebenfalls durch zu spezifizierende Qualitätsannotationen erreicht. Für eingehendere Betrachtungen dazu sei auch auf Abschnitt 7.1 verwiesen.

C1 Vererbung Wird durch die Anwendung der Unterklassenbeziehung von OWL erfüllt.

C2 Äquivalenz Äquivalenzbeziehungen werden durch das entsprechende OWL-Konstrukt modellierbar. Für Ähnlichkeitsbeziehungen gibt es noch kein geeignetes, ausgereiftes Konstrukt. Hier lässt sich eventuell einiges über das Instrument der Regeln ausdrücken. Dies ist ein Ansatzpunkt für weitere Arbeit: Wie lässt sich Ähnlichkeit ausdrücken, beschreiben und wie lassen sich Bedingungen dafür ausdrücken. Spannend ist auch die Frage hinsichtlich welcher Eigenschaften zwei Klassen ähnlich sind.

C3 Assoziations-Domänen- und Bereiche Wird durch die jeweiligen Spezialisierungen der OWL-Konstrukte „domain" und „range" erfüllt.

C4 Nebeneinander von Kontextmodellen Wird durch die Generik einerseits und die XML-Eigenschaften von OWL (unter anderem der Einsatz von Namensräumen) erfüllt.

C5 Erweiterbarkeit Auch diese Anforderung erfüllt CMPlus bereits durch die entsprechenden XML-Eigenschaften.

D1 CIS-Beschreibung Wird auch voll erfüllt. Hier sei auf Kapitel 6 verwiesen.

D2 Grundlage für Wissensbasis Wird voll ermöglicht. Hier sei verwiesen auf die Wissensbasis-Ausschnitte in diesem Kapitel und Erläuterungen zu deren Verwendung in den folgenden.

D3 Anfragesprache Wird erfüllt. Hier sei insbesondere auf Kapitel 6 verwiesen.

D4 Verknüpfung verschiedener Modelle Wird einfach möglich, indem die entsprechenden Klassen aus verschiedenen CMPlus-Ontologien verwandt werden. Die Eindeutigkeit bleibt dabei durch die XML-Namensräume gewahrt.

D5 Integration fremder Modellierungen Ist – abhängig von der fremden Modellierungsart – in Grenzen möglich. Hier sei ebenfalls auf Kapitel 6 verwiesen.

E11 Toolunterstützung Hier profitiert CMPlus den vielen XML-Tools und den sich immer weiter entwickelnden OWL-Werkzeuge. Letztere werden gerade im Zuge der Arbeit zum *Semantic Web* verstärkt entwickelt, darunter auch Inferenzsysteme, die auf OWL DL basieren.

E2 Übertragung Hier profitiert CMPlus von der XML-Kodierung, die genau für diesen Anwendungszweck entwickelt worden ist und einfachen und schnellen Transfer über Systemgrenzen hinweg erlaubt.

Um dies zusammenzufassen: CMPlus ist ein geeigneter Ansatz zur Kontextmodellierung, der konzeptionell den gestellten Anforderungen in vollem Umfang genügt. An dieser Stelle sei auch noch auf Kapitel 8 verwiesen, in dem auf die prototypische Implementierung eingegangen wird.

6 Suche nach Kontextinformationen

> Es zeichnet einen gebildeten Geist aus, sich mit jenem Grad an Genauigkeit zufrieden zugeben, den die Natur der Dinge zulässt, und nicht dort Exaktheit zu suchen, wo nur Annäherung möglich ist.
>
> *(Aristoteles)*

Auf Basis der CMPlus-Modellierung werden in diesem Kapitel Mechanismen zur Vermittlung von Kontextinformationen vorgestellt. Zunächst wird dazu eine Modellierung von CMPlus-Kontextanfragen für die Kommunikation zwischen Kontextanfrager und Kontextvermittler entwickelt. Anschließend werden eine Dienstmodellierung und eine Dienstvermittlung vorgestellt, die den Besonderheiten von Kontextinformationsdiensten und der Suche nach ihnen Rechnung tragen.

6.1 Die CMPlus-Kontextanfrage

In diesem Abschnitt wird die Modellierung für die Kontextanfrage entwickelt und damit ein Teil des Protokolls zwischen Kontextanfrager und Kontextvermittler.

6.1.1 Grundlegender Aufbau der Kontextanfrage

Ein offenes, ubiquitäres System ist geprägt von der möglichen Mobilität der Benutzer und der Geräte. Die daraus resultierende Konfiguration kann erst zur Laufzeit erfolgen. Deshalb kann ein kontextsensitiver Dienst nicht zu jedem Zeitpunkt die aktuell erforderlichen Kontextinformationsdienste kennen. Und deshalb muss eine Kontextanfrage rein deklarativ erfolgen. Sie kann wie in Abschnitt 3.3 beschrieben durch einen Kontextvermittler ausgewertet werden, der dann die erforderlichen Schritte unternimmt, um die Kontextinformation bereit zu stellen.

Nachdem mit CMPlus eine eigene, neuartige Modellierung für Kontextinformationen entwickelt worden ist, ist auch eine eigene, CMPlus-basierte Modellierung für Kontextanfragen notwendig. Die in diesem Abschnitt vorgestellte Modellierung für Kontextanfragen basiert auf den Erfahrungen mit prototypischen Implementierungen, in deren Rahmen die Modellierung inkrementell bis zum jetzigen Stand erweitert und verbessert worden ist.

Das allgemeine Prinzip, nach dem eine CMPlus-Kontextanfrage aufgebaut ist, ist sehr simpel: Im Kontexttripel aus Subjekt-Prädikat-Objekt wird das Objekt weggelassen, das

durch eine Entität oder eine Datenstruktur repräsentiert werden kann. Übrig bleiben eine Entität als Subjekt und die Kontextinformationsklasse als Prädikat. Damit wird ausgesagt, *welche Art von Kontextinformation* über *welche Entität* gesucht wird. Diese beiden Informationen können präzise angegeben werden. Alle weiteren charakterisierenden Angaben wie Qualitätskriterien dagegen müssen als Bereich oder Schranke angegeben werden. Den grundlegenden Aufbau einer Kontextanfrage gibt das folgende Listing 6.1 wieder:

```
<CONTEXT_INFORMATION_FACTORY>
 <ENTITY>
  <CONTEXT_INFORMATION isIdentity="isIdentity" />
 </ENTITY>
 <QUALITY_CONDITION />
 <PROCESSING_INFORMATION />
</CONTEXT_INFORMATION_FACTORY>
```

Listing 6.1: Schematischer Aufbau einer Kontextanfrage.

Das hier allgemein als „Context_Information_Factory“ bezeichnete Element beschreibt die gesuchte Kontextinformationsklasse. Die „Entity“ gibt diejenige Entität an, welche als Subjekt Bezugspunkt der Kontextinformation ist. Sie wird über die „Context_Information“ mit dem Attribut „isIdentity“ eindeutig identifiziert. Darüber hinaus gibt es Qualitätsbedingungen, die die gesuchte Kontextinformation näher charakterisieren und Verarbeitungsinformationen, die Vorgaben für den Beschaffungsprozess machen.

Im folgenden Listing 6.2 wird zum Beispiel der Ort einer Person gesucht, die mit einer „ID-Nummer“ identifiziert wird. Alle Elemente entstammen dabei einer CMPlus-Ontologie, deren Namensraum „o1“ abgekürzt wird. Das Element „hasLocationFactory“ steht dabei für die Anfrage der Kontextinformationsklasse „hasLocation“. Das Attribut „isDerivedDepth“ dieses Elements gibt an, bis zu welcher Tiefe die Historie der Kontextinformation mitgeliefert werden soll. Eine Historie gibt es nur, wenn Kontextinformationen aus anderen Kontextinformationen abgeleitet werden, nicht aber, wenn sie direkt von logischen oder physikalischen Sensoren stammen. Die „0“ besagt natürlich, dass auf die Historie ganz verzichtet wird.

```
 <hasLocationFactory isDerivedDepth="0">
  <person>
   <hasIDNumber isIdentity="isIdentity">
    <idNumber>
     <longType>6543542342</o1:longType>
    </idNumber>
   </hasIDNumber>
  </person>
</hasLocationFactory>
```

Listing 6.2: Kontextanfrage nach einer Ortsinformation.

6.1.2 Modellierung der Qualitätsbedingungen

Für die Modellierung der Qualitätsbedingungen in einer Kontextanfrage gibt es die folgenden Anforderungen:

- Es müssen optionale und zwingende Qualitätsbedingungen gestellt werden können.
- Bedingungen müssen sowohl für Qualitätsparameter formuliert werden können, auf denen eine Ordnung definiert ist, als auch für solche ohne Ordnung.
- Es muss möglich sein, Qualitätsbedingungen zu verknüpfen.
- Bedingungen müssen nicht nur für Qualitätsparameter von Kontextinformationen sondern auch für Qualitätsparameter von Qualitätsparametern (zur Rekursivität von Qualitätsparametern siehe auch Abschnitt 5.5) darstellbar sein.

Abbildung 6.1 zeigt im Überblick, wie die Qualitätsbedingungen in Kontextanfragen aufgebaut sind [Alb05]. Das abstrakte Basiskonstrukt dafür ist die „QualityCondition“, die zwei Attribute besitzt: „mandatory“ ist ein boolesches Attribut, das standardmäßig nicht gesetzt ist. Erst wenn dieses Attribut mit dem Wert „true“ angegeben ist, wird die jeweilige Bedingung zwingend. Das andere Attribut ist „not“, das – falls gesetzt – die Qualitätsbedingung logisch invertiert. Von dieser Basisklasse gibt es fünf konkrete Spezialisierungen:

- Mit dem Element „boundaryQualityCondition“ werden Grenzwerte angegeben. Das Element beinhaltet genau einen Wert eines Qualitätsparameters („Quality“). Mit der Darstellung der Spezialisierung „OrderedQuality“ in Abbildung 6.1 soll angedeutet werden, dass auf der entsprechenden Qualitätsklasse eine Ordnung definiert sein muss. Mit den Attributen „type“ und „include“ wird dabei definiert, ob die Bedingung einen Wert verlangt, der größer oder kleiner als der angegebene Grenzwert ist, und ob der Grenzwert selbst auch erlaubt ist.
- Das Element „elementOfQualityCondition“ ist das entsprechende Gegenstück für Qualitätsklassen, auf denen keine lineare Ordnung definiert ist. Hier werden direkt alle Qualitätswerte angegeben, die erlaubt sind (oder ausgeschlossen, falls das Attribut „not“ gesetzt wird).
- Die drei Elemente „andQualityCondition“, „orQualityCondition“, „xorQualityCondition“ beinhalten im Gegensatz zu den vorgenannten Elementen keine Qualitätswerte, sondern selbst Qualitätsbedingungen, die dadurch entsprechend des Elementnamens logisch verknüpft werden. Da auch ein Ineinanderschachteln erlaubt ist, können auf diese Weise komplexe Bedingungen formuliert werden.

Zur Veranschaulichung wird im folgenden Listing 6.3 das Beispiel aus Listing 6.2 um komplexe Qualitätsbedingungen erweitert:

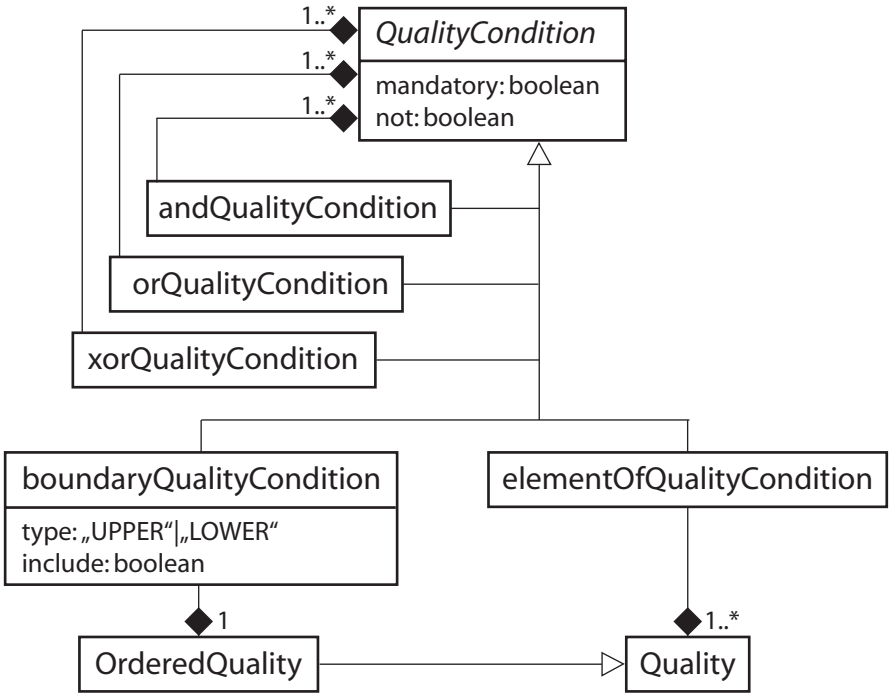


Abbildung 6.1: Modellierung von Qualitätsbedingungen in einer Kontextanfrage.

```
<hasLocationFactory>
 <person>
  <hasIDNumber isIdentity="isIdentity">
   <idNumber>
    <number><longType>6543542342</longType></number>
   </idNumber>
  </hasIDNumber>
 </person>
 <orQualityCondition>
  <boundaryQualityCondition type="UPPER" include="false">
   <resolution>
    <meter><floatType>1</floatType></meter>
   </resolution>
  </boundaryQualityCondition>
  <boundaryQualityCondition type="UPPER" include="false">
   <maximalDeviation>
    <meter><floatType>2.3</floatType></meter>
   </maximalDeviation>
  </boundaryQualityCondition>
  <elementOfQualityCondition>
   <contextSource>
    <name><stringType>Vertrauensvoll & Co. KG </stringType></name>
   </contextSource>
```

```
  </elementOfQualityCondition>
 </orQualityCondition>
</hasLocationFactory>
```

Listing 6.3: Beispiel für Kontextanfrage aus [Alb05].

In dieser Kontextanfrage wird also verlangt: Die Auflösung der Kontextinformation muss kleiner sein als 1 Meter, außer die Abweichung beträgt maximal 2,3 Meter oder die Kontextinformation stammt von der Kontextquelle „Vertrauensvoll & Co. KG“.

Die in Listing 6.1 ebenfalls vorgesehenen Verarbeitungsinformationen („Processing_Information“) sind bislang noch nicht ausgestaltet worden. Es ist vorgesehen, darin Vorgaben etwa zu Kosten, zu maximalen Wartezeiten oder zu Parametern der Privatsphäre zu machen.

6.1.3 Komposition von Kontextanfragen

Beginnend mit einer Arbeit des Autors [Kra03] ist für die (damals noch rudimentären) CMPlus-Kontextanfragen eine Kompositionssprache namens CoCo entwickelt worden, mit der vom Kontextanfrager zum Kontextvermittler nicht nur einzelne Kontextanfragen abgesetzt werden können, sondern auch komplette Kompositionspläne, die Anleitung geben, wie höherwertige Kontextinformationen, die mutmaßlich nicht direkt verfügbar sind, aus niederwertigeren erstellt werden können. Diese Kompositionsvorschriften sind Graphen („CoCoGraphen“), wobei einzelne Kontextanfragen, wie sie im vorangegangenen Abschnitt beschrieben worden sind, darin in Kontextanfrageknoten gekapselt werden. CoCo wird in Abschnitt 7.1.1 ausführlicher vorgestellt.

6.2 Vermittlung von Kontextinformationsdiensten

Mit der im vorangegangenen Abschnitt entwickelten Kontextanfragesprache ist es möglich, Anfragen für Kontextinformationen auszudrücken. Dies geschieht in der Kommunikation zwischen Kontextanfrager und Kontextvermittler. Nun muss auch noch der Kontextvermittler in die Lage versetzt werden, anhand der Kontextanfragen passende Kontextinformationsdienste zu finden, um von diesen die Kontextinformationen zu besorgen und dem Kontextanfrager bereitzustellen. Gelingt ihm dies trotzdem nicht, so bleiben ihm noch diverse Ausweichstrategien, die in Kapitel 7 vorgestellt werden.

6.2.1 Problemstellung

Hat ein Kontextvermittler eine Kontextanfrage erhalten, benötigt er einen Kontextinformationsdienst, der ihm die angefragte Kontextinformation liefern kann. Der dazu notwendige Vorgang der Suche und Vermittlung eines Kontextinformationsdienstes unterscheidet sich von der klassischen Dienstsuche und Dienstvermittlung, denn die grundlegende Aufgabe

lautet jedes Mal: Suche einen Dienst, der für eine konkrete Entität E den Kontext (die Kontextinformationsklasse) K zum Zeitpunkt t angeben kann.

Während klassische Dienstsuchen primär nach Diensten eines bestimmten Diensttyps suchen, reicht das hier nicht mehr aus: Der Diensttyp „Kontextinformationsdienst“ ist bei allen Anfragen gleich. Selbst wenn man die jeweilige Kontextinformationsklasse, die ein Kontextinformationsdienst liefert, noch als Spezialisierung des Diensttyps betrachtet, ist dies als primäres Suchkriterium nicht ausreichend. Schließlich sagt dies noch nichts darüber aus, ob der Dienst die Kontextinformation auch für die gewünschte Entität liefern kann.

Eine weitere Unzulänglichkeit klassischer Dienstvermittlungssysteme bei der Vermittlung von Kontextinformationsdiensten ist, dass sie Anfrage und Angebot nur syntaktisch abgleichen, in der Regel mit einem Stringvergleich. Kontextinformationen, die wie in Kapitel 5 beschrieben auf Ontologien basieren, können durch Spezialisierungen und Äquivalenzen semantisch gleich, aber syntaktisch höchst unterschiedlich sein. Notwendig ist also ein semantischer Abgleich zwischen Dienstanforderung und Dienstangebot.

Im Folgenden sollen nun zunächst die klassischen Dienstvermittlungsmechanismen näher beleuchtet werden, bevor neue Ansätze vorgestellt werden. Anschließend wird das eigene Konzept zur Beschreibung und Vermittlung von Kontextinformationsdiensten erläutert.

6.2.2 Klassische Dienstvermittlungsprotokolle

Die grundlegende Idee bei der Dienstvermittlung kennt drei Parteien ([PKM94], vgl. auch [PG03]): Der Anbieter publiziert die Beschreibungen angebotener Dienste, der Vermittler sammelt diese Dienstangebote und der Klient sucht einen Dienst, indem er eine Dienstanforderung an den Vermittler sendet. Dabei gibt es zwei Arten von Dienstvermittlungssystemen: Die einen halten sich selbst aus dem Vorgang der Dienstbeauftragung und -nutzung komplett heraus und teilen dem Klienten lediglich den Ort von Diensten mit, die seiner Dienstanforderung entsprechen. Andere unterstützen die Beauftragung, indem sie neben den Dienstbeschreibungen auch die Schnittstellen der Dienste vorrätig halten, oder sie fungieren gleich als Stellvertreter für den beauftragten Dienst, wobei alle Kommunikation nur über den Vermittler geschieht. Stellvertretend sollen nun vier Vertreter von Vermittlungsprotokollen und Vermittlungsarchitekturen betrachtet werden: Jini-LUS, UPnP, SLP und UDDI.

Jini-LUS: Lookup-Service

Jini [jin03a, jin03b] ist eine offene Softwarearchitektur von Sun Microsystems und umfasst mehrere Spezifikationen zur Programmierung von verteilten Anwendungen (in Java), die dynamisch in lokalen Netzen miteinander kommunizieren. Ein Teil dieser Spezifikationen beschreibt einen Mechanismus zur Dienstvermittlung zur Laufzeit. Jini[1] ist unabhängig

[1] Der Begriff Jini ist dabei kein Akronym. Expansionen wie „*Java Intelligent Network Interface*“ oder „*Java Intelligent Network Infrastructure*“ sind erst später hinein interpretiert worden. Ken Arnold, einer der

von der verwandten Netzwerktechnologie und kann auf Java *Remote Method Invocation* (RMI) [jrm03] ebenso wie auf dem *Simple Object Access Protocol* (SOAP) [BEK+] oder anderen Protokollen aufgebaut werden. Die Unabhängigkeit von der Plattform ergibt sich durch die Verwendung der Java *Virtual Machine*.

Bei Jini heißt die Vermittlungskomponente *Lookup Service* (LUS). Dienstanbieter (*Service Provider*) hinterlegen dort so genannte *Service Objects*, die zum Aufruf des Dienstes benutzt werden können, und gegebenenfalls weitere, charakterisierende Attribute. Klienten (*Clients*) formulieren in der Dienstanfrage Schnittstelle und Attribute des gesuchten Dienstes und erhalten serialisiert das erste *Service Object* oder alle *Service Objects*, die der Anfrage entsprechen (je nach Einstellung).

Einträge, Anfragen und dynamische Attribute haben eine bestimmte Lebensdauer. Diese so genannten *Leases* müssen regelmäßig erneuert werden, wenn sie über die mit dem LUS ausgehandelte Zeitspanne hinaus gültig sein sollen, andernfalls werden die Ressourcen wieder freigegeben.

Da das Ausführen einer Java-Umgebung rechenschwache Klienten überfordern kann, ist im *Surrogate*-Projekt ein Standard [jsu03] entwickelt worden, der beschreibt, wie diese mit einem Proxy-Mechanismus entlastet werden können. Choonhwa Lee and Sumi Lehal haben außerdem einen Ansatz [LL03] entwickelt, wie dynamische Attribute nicht beim LUS hinterlegt werden, sondern erst zur Laufzeit eingeholt werden können.

UPnP: Universal Plug and Play

Bei *Universal Plug and Play* (UPnP) [upn00b, upn00a] liegt wie bei Jini der Fokus auf lokalen Netzen und einfacher Konfigurierbarkeit. Microsoft ist zwar die treibende Kraft hinter der Initiative, doch die Standardisierung erfolgt im UPnP-Forum, dem zur Zeit knapp 800 Unternehmen angehören. UPnP kennt drei Arten von Komponenten: Geräte (*Devices*), Dienste (*Services*) und Steuerungseinheiten (*Control Points*). Geräte sind Container, die Dienste und wiederum Geräte enthalten können. Sie halten eine XML-Gerätebeschreibung vor, in der unter anderem die enthaltenen Dienste aufgeführt sind und Referenzen auf die XML-Dienstbeschreibungen. Ein Dienst hält Dienstprimitive und Zustandsvariablen vor. Über die Änderung von letzteren kann man sich über Ereignisse benachrichtigen lassen. Steuerungseinheiten können Gerätebeschreibungen und die damit verbundenen Dienstbeschreibungen abrufen, gezielt die Beschreibungen interessanter Dienste suchen, Dienstprimitive der Dienste aufrufen und sich bei Diensten registrieren, um bei Zustandsänderungen benachrichtigt zu werden.

Um den Netzverkehr gering zu halten, gelangen die Information über verfügbare Geräte und Dienste auf zwei Wegen zu den Steuereinheiten: EInmal, indem sie von den Geräten in regelmäßigen Abständen von ihrer Anwesenheit benachrichtigt werden, zum anderen, indem die Steuereinheiten selbst gezielt nach Geräten oder Diensten suchen. Da diese Verfügbarkeitsinformationen mit einer Gültigkeitsdauer versehen sind, arbeitet auch UP-

Urheber, überspitzt dies mit einer GNU-ähnlichen Expansion: „*Jini Is Not Initials*“.

nP wie Jini mit einem *Leasing*-Konzept. Um die Semantik von Diensttyp, Dienstprimitiven und Zuständen einheitlich zu gestalten, werden diese im UPnP-Forum je nach Geräte- und Diensttyp spezifiziert. Diese Grundspezifikation kann durch einen Hersteller um eigene Befehle erweitert werden. Bei der Suche nach einem Dienst ist es nicht möglich, nach bestimmten Eigenschaften eines Dienstes zu suchen – man kann lediglich direkt nach bestimmten Gerätetypen oder Diensten suchen mittels eines *Uniform Resource Name* [Moa97] – zum Beispiel in der Form „urn:schemas-upnp-org:device:printer:1“ (vgl. [Mat05]). Die Kommunikation erfolgt über TCP/IP, wobei die Geräte dynamisch mittels DHCP oder Auto-IP (im Gegensatz zu DHCP ohne zentrale Vergabeinstanz, dafür mit Kollisionserkennenung) mit einer IP-Adresse ausgestattet werden.

Es gibt zahlreiche Ansätze, das der Erkennung von (in der Regel mobilen) Geräten dienende UPnP in die allgemeinere, aber auch komplexere Dienstvermittlungsarchitektur von JINI zu integrieren [New05, ACGI03, SLL05].

SLP: Service Location Protocol

Das *Service Location Protocol* (SLP) [GPD99, BR00] ist von der „*Service-Location*“-Arbeitsgruppe der *Internet Engineering Task Force* entwickelt worden. Im Gegensatz zu UPnP und Jini (zumindest dem Jini ohne Erweiterungen), die für lokale Netze gedacht sind, hat SLP zum Ziel, soweit zu skalieren, dass Dienstvermittlung auch in großen Firmennetzen möglich ist. SLP kennt drei klassischen Rollen der Dienstvermittlung. Hier heißen sie Benutzeragenten („*User Agents*“), Dienstagenten („*Service Agents*“) und Verzeichnisagenten („*Directory Agents*“), wobei die meisten Geräte sowohl Benutzeragent als auch Dienstagent sein können, und oft auch die Rolle des Verzeichnisagenten übernehmen können.

Dienste werden beschrieben durch eine URL mit dem Dienstzugriffspunkt und einem Dienst *Template*, das die chrakteristischen Attribute eines Dienstes und deren Standardwerte beschreibt. Für einen Drucker könnte dies wie folgt aussehen:

```
service:printer://lj4050.tum.de:1020/queue1
scopes = tum, bmw, administrator
printer-name = lj4050
printer-model = HP LJ4050 N
printer-location = Room 0409
color-supported = false
pages-per-minute = 9
sides-supported = one-sided, two-sided
```

Listing 6.4: Beispiel für ein SLP-Dienst-*Template* nach [BR00].

Die Angabe „service:printer“ beschreibt dabei den Diensttyp. Dienste können nach ihrem Typ und nach ihren (Attribut,Wert)-Paaren gesucht werden. Dies funktioniert auch in einem Netz ohne Verzeichnisagent, dann sendet der Benutzeragent die Dienstsuche mittels Multicast-Nachricht ab. Dienstagenten, die sich selbst in der Suchanfrage erkennen,

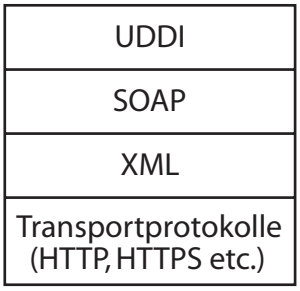


Abbildung 6.2: UDDI-Stack nach [Wut02a].

antworten dann direkt. Im Fall einer Dienstsuche bei einem Verzeichnisagenten antwortet dieser an Stelle des Dienstagenten. Da die Antwort die URL des betreffenden Dienstes enthält, kann der Benutzeragent den Dienst dann direkt aufrufen.

Dienstagenten, die sich bei einem Verzeichnisagenten registrieren, müssen diese Registrierung periodisch erneuern.

UDDI: Universal Description, Discovery, and Integration Protocol

Das *Universal Description, Discovery, and Integration Protocol* (UDDI) [udd04, Wut02a, Wut02b] beschreibt einen Mechanismus zum Auffinden von *Web Services*. *Web Services* sind im Prinzip Anwendungen, die über das Internet miteinander kommunizieren können. [WG02] Doch wer von *Web Services* spricht, meint selten die Anwendungen selbst, sondern die Techniken und Standards, die diese Kommunikation ermöglichen. Dabei spricht man in der Regel vom *Web-Services*-Dreieck aus SOAP, WSDL und UDDI [CJ02] als den Kernstandards. Das *Simple Object Access Protocol* (SOAP) [BEK+, Mar02a, Mar02b] definiert ein Kommunikationsprotokoll für die Verbindung zu einem *Web Service*, das unabhängig von der jeweiligen Programmiersprache, dem Objektmodell oder dem Betriebssystem ist und meist, aber nicht notwendigerweise, auf HTTP als Transportmedium aufsetzt. Und die *Web Service Description Language* (WSDL) [CCMW01, PM] ist geeignet, die Schnittstellen-Definition eines *Web Service* zu erstellen. Dazu gehört das Format der Anforderungs- und Antwort-Nachrichtenströme, mit denen Funktionsaufrufe an andere Programm-Module abgesetzt werden.

UDDI selbst baut auf SOAP auf, wie Abbildung 6.2 zeigt. Dabei ist UDDI weniger verbreitet als die anderen *Web-Services*-Kernstandards. Weniger als zehn Prozent der Unternehmen benutzen UDDI als Verzeichnis für ihre SOA-Anwendungen, so eine Gartner-Studie [AVS05], die in ihrer Analyse mehr Bemühungen bei der Standardisierung von Namenskonventionen für die Dienste anmahnen. Äußerst wichtig dabei sei die Semantik der Dienste. In UDDI lassen sich Dienste nach Namen oder Attributen suchen. Ist ein passender Eintrag gefunden, verläuft die eigentliche Interaktion der Dienstnutzung ohne

UDDI ab. Grundlegender Datentyp ist dabei das „tModel“, ein generischer Container, der detaillierte Dienstinformationen zusammenfasst und dabei selbst wieder „tModels“ enthalten kann. Seine Bedeutung ist jeweils in einem Spezifikationsdokument (zum Beispiel formuliert in WSDL) beschrieben.

Bewertung

Die Schwachstellen der Techniken zur allgemeinen Dienstvermittlung sind dabei vor allem:

- Die Lösungen bieten nicht ausreichend Dynamik. Die jeweiligen Leasing-Konzepte gelten nur für komplette Dienstbeschreibungen und sind eher dafür gedacht zu verhindern, dass Dienste noch geführt werden, die gar nicht mehr zur Verfügung stehen. Die Beschreibungen von Kontextinformationsdiensten können dagegen Aktualisierungen notwendig machen, die in unterschiedlichen Teilen der Beschreibung unterschiedliche Dynamik brauchen, zum Beispiel ereignis- oder zeitgesteuert. Dabei kann die Aktualisierungsrate so groß sein, dass es sich nicht lohnt, die Dienstbeschreibung beim Vermittler ständig auf dem Laufenden zu halten, sondern nur im konkreten Vermittlungsfall die aktuellen Werte einzuholen.

- Die Semantik der Dienstbeschreibung wird von den vorgestellten Mechanismen nur unzureichend verarbeitet. Die objektorientierten Dienstbeschreibungen können wenigstens noch eingeschränkt Spezialisierungen bei Diensttyp und Attributklassen erkennen, sind aber bei Mehrfachvererbung und Äquivalenzen überfordert, wie sie typisch für Ontologien sind. Für die sehr spezialisierte Aufgabe, Kontextinformationsdienste zu beschreiben, sind die Techniken zu generisch und bieten keine Modellierungshilfe (zum Beispiel zur Typisierung der Dienstattribute) an. Hier ist eine eigene Modellierung für Dienstbeschreibungen notwendig, die idealerweise ein bestehendes Dienstvermittlungssystem ergänzt und auf der CMPlus-Modellierung von Kontextinformationen aufsetzt.

- Teilweise (UPnP und Jini ohne Erweiterungen) sind die vorgestellten Techniken nur für lokale Netze gedacht und deshalb per se nicht in offenen Systemen einsetzbar, die potenziell einen globalen Wirkungskreis haben.

Keine der vorgestellten Techniken ist somit ohne größere Erweiterung zur Suche nach Kontextinformationsdiensten geeignet.

6.2.3 Semantische Ansätze

Im Folgenden sollen einige, teilweise neuere Ansätze betrachtet werden, die die Semantik von Dienstbeschreibungen stärker in den Dienstvermittlungsprozess miteinbeziehen.

Liquid

Liquid [HNBH03] ist ein Ansatz, der auf *Context Fabric* [Hon02] aufbaut und vorschlägt, für jede Entität einen Container im Netz zu schaffen, an dem alle Kontextinformationen für diese Entität gespeichert sind. Diese Informationsräume („Infospaces“) halten für jede relevante Kontextinformationsklasse („type“) entweder lokal einen Datenwert vor oder eine Referenz auf den Informationsraum einer anderen Entität. Die Referenzen sind URLs und der Datenaustausch wird über HTTP realisiert.

Mit *Liquid* wird die Suche nach Kontextinformationen sehr einfach und deterministisch. Eine Anfrage muss lediglich den existierenden Pfaden folgen, um die richtige Kontextinformation zu erhalten. Die Probleme verschieben sich jedoch: Die Suche nach passenden Kontextinformationen und nach Diensten, die sie liefern können, ist nun nötig, um die Informationsräume zu befüllen und aktuell zu halten. Liquid eignet sich deshalb nur zur Realisierung einzelner, konkreter Anwendungen in festgelegten Domänen. Denn in offenen, ubiquitären Systemen ließen sich weder die Art und Anzahl der Entitäten noch die Menge der Kontextinformationen a priori festlegen, was zum Aufbau der *Liquid*-Informationsräume aber zwingend notwendig ist.

Context Shadow

Ähnlich wie *Liquid/ContextFabric* hat auch *ContextShadow* [Jon02] das Ziel, die logischen Strukturen der realen Welt nachzubilden. Jeder Entität entspricht ein Dienst, der die eigenen Daten in Form von Tupeln in einem gemeinsamen Datenraum vorrätig hält. Wie bei *Liquid/ContextFabric* gibt es Referenzen zwischen diesen Diensten, so genannte *CrossLinks*. Diese entsprechen Kontextinformationen, bei denen sowohl Subjekt als auch Objekt jeweils eine Entität sind.

Da hinter *ContextShadow* die gleiche Idee wie im zuvor vorgestellten Ansatz steckt, sind auch hier die Probleme entsprechend. In einem offenen System, das generisch für jede denkbare Art von kontextsensitiven Anwendungen und Kontextinformationen funktionieren soll, ist es praktisch nicht möglich, a priori feste, logische Strukturen aufzubauen. Dies funktioniert nur für konkrete Anwendungsbereiche.

OWL-S

Die *OWL Services Coalition* ist ein Zusammenschluss zahlreicher universitärer Forscher, die die *Web Ontology Language for Services* (OWL-S) [MM03, MPM+04, MBH+04, MBL+04, Mat05] standardisiert haben. Dies geschah im Rahmen des *DARPA-MarkUp-Language*-Programms, aus welchem auch DAML hervorgegangen ist – ein Vorgänger von OWL (siehe Abschnitt 4.3.2). OWL-S hieß in früheren Versionen DAML-S.

Zunächst geht es lediglich darum, mit OWL-S Dienste und deren Semantik als Teil des *Semantic Web* beschreiben zu können, der Vermittlungsaspekt ist dabei noch außen vor. Allerdings nennen die Autoren als Motivation für OWL-S, dass sie damit ermöglichen

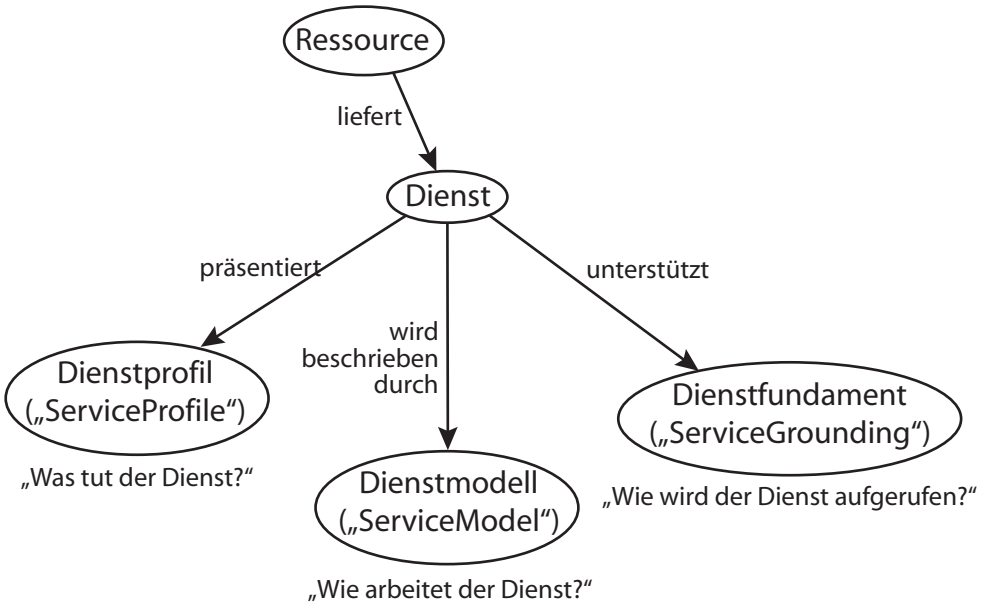


Abbildung 6.3: OWL-S Generalontologie für Dienste nach [MBH+04].

wollen, Dienste automatisch zu suchen, automatisch aufzurufen, automatisch zusammenzuschließen und automatisch zu überwachen.

OWL-S ist auch die Grundlage für die Arbeit des *Semantics Web Services Language*-(SWSL)-Komitees der *Semantic Web Services Initiative* (SWSI). Diese Initiative entwickelt OWL-S weiter und will außerdem ein Forum bieten für alle Anstrengungen, OWL-S mit den Ergebnissen der *Web-Service-Modeling-Ontology*-(WSMO)-Projektes [wsm04] zu verschmelzen (vgl. Abschnitt 6.2.3).

OWL-S teilt eine Dienstbeschreibung in drei Teile: Das Dienstmodell („ServiceModel“), das den Dienst detailliert beschreibt, das Dienstprofil („ServiceProfile“), das dem Abgleich mit einer Dienstsuche dient, und dem Dienstfundament („ServiceGrounding“), das der Beschreibung und Erstellung von Schnittstellen dient. Die OWL-DL-Spezifikation dieser Generalontologie in der Version 1.0 gibt es an dieser Stelle herunterzuladen: www.daml.org/services/owl-s/1.0DL/Service.owl. Für die aktuelle Version 1.2 gibt es bislang nur eine OWL-Full-Spezifikation.

In OWL-S ist das Dienstprofil (siehe Abbildung 6.4) die Möglichkeit für jemanden, der einen Dienst sucht, zu beschreiben, wie dieser Dienst aussehen soll. Auf der anderen Seite können Dienstanbieter damit die Dienste beschreiben, die sie anbieten. Bei der Dienstsuche müssen also die Dienstprofile der Anbieterseite und der Anfragerseite abgeglichen werden. Für die eigentliche Dienstbenutzung dient anschließend das Prozessmodell (eine Unterklasse des Dienstmodells) als detaillierte Dienstbeschreibung.

Das hat zur Folge, dass sowohl Dienstprofil als auch Dienstmodell zwar separate Informationen darstellen, obwohl das Dienstprofil genau genommen eine Untermenge oder

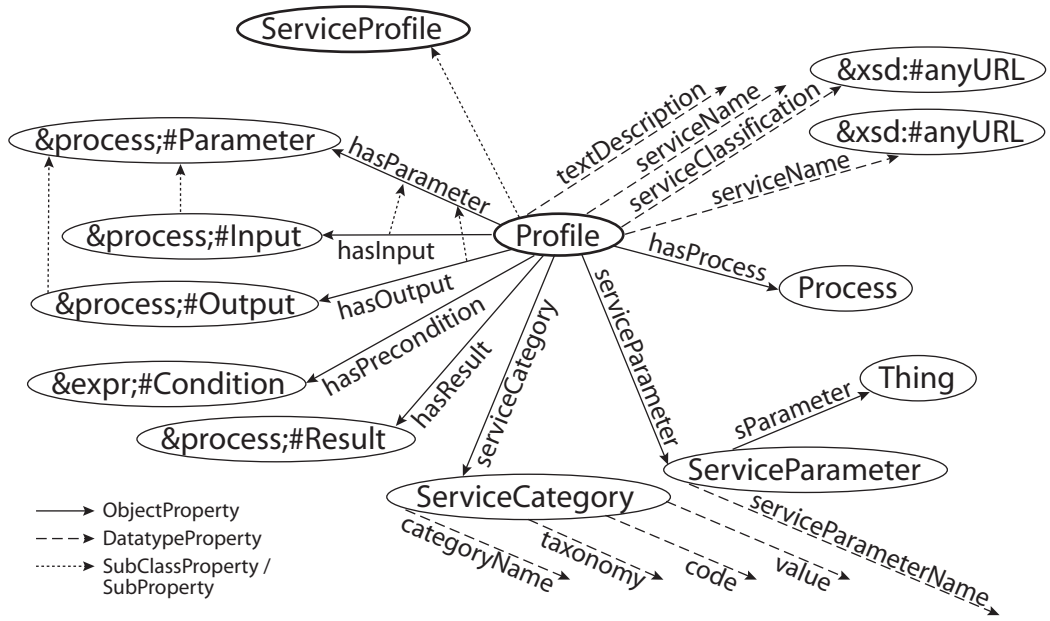


Abbildung 6.4: OWL-S Dienstprofil nach [MBH+04].

eine Abstraktion des Dienstmodells ist. Somit wären auch Inkonsistenzen möglich. Sowohl im Dienstprofil als auch im Dienstmodell ist das IOPE-Modell Grundlage für die Dienstbeschreibung. „IOPE“ steht für *input*, *output*, *preconditions* und *effects*. Diese vier Teile lassen sich in zwei Kategorien fassen:

- Die datenorientierten Teile *input* und *output* beschreiben die Eingabeparameter und die Rückgabewerte des Dienstes. Und zwar nicht nur syntaktisch (als „String“ und „Integer“), wie das zum Beispiel in WSDL der Fall wäre, sondern auch semantisch. Zum Beispiel werden „Geokoordinaten“ als Eingabe erwartet, und „Temperaturangaben“ ausgegeben.
- Der logische Teil beschreibt zum einen die Vorbedingungen („preconditions“), die gelten müssen, damit der Dienst überhaupt arbeiten kann (Zum Beispiel: „Verbindung zum Internet“). Zum anderen werden die Nachbedingungen formuliert, die nach Ablauf des Dienstes gelten und die Veränderung beschreiben, die der Dienst bewirkt hat.

Diese Art, Dienste zu beschreiben, basiert auf folgendem Dienstverständnis: Demnach kann ein Dienst nur zwei Arten von Zielen haben, zum einen Informationen auszugeben, die aufgrund der Eingabewerte und der Zustands der Welt erstellt worden sind, zum anderen kann er eine Veränderung des Zustands der Welt erwirken. Die ausgegebene Information wird durch den datenorientierten Teil, die bedingten Zustandsübergänge der Welt werden durch den logischen Teil der Dienstbeschreibung modelliert.

OWL-S und UDDI Während OWL-S zwar Dienstprofile für das Suchen nach Diensten spezifiziert, beschreibt es aber nicht die eigentliche Suche. UDDI, das das Auffinden von Diensten unterstützt, fehlt es dafür an Aussagekraft bei der Beschreibung von Diensten. OWL-S und UDDI ergänzen sich also. Folgerichtig gibt es auch mehrere Vorhaben, OWL-S und UDDI zu verbinden. So zum Beispiel Stanislav Pokraev, Johan Koolwaaij und Martin Wibbels [PKW03] mit ihrem Ansatz „UDDI+“, wobei eine OWL-S Beschreibung einem „tModel“ als Ganzes zugeordnet wird. Anders Massimo Paolucci et al. [PKPS02], die in ihrem *Semantic-Service-Matchmaker* die OWL-S-Beschreibung aufsplitten und für jedes Konstrukt im Dienstprofil ein eigenes „tModel“ anlegen. Das Abgleichen der Dienstbeschreibungen des Anfragers mit den OWL-S-Beschreibungen in dem erweiterten UDDI-Verzeichnis erfolgte zunächst iterativ: Es wird für jede Angebotsbeschreibung überprüft, in welchem Verhältnis die Klassen der Eingaben und Ausgaben stehen, und es wird ein entsprechendes Ranking erstellt: Ob die des angebotenen Dienstes exakt übereinstimmen, oder eine Ober- oder eine Unterklasse des Geforderten darstellten, oder ob sie überhaupt nicht zusammenpassen. In einer überarbeiteten Version [KdBH+03, SuKPS04] wird der Abgleichvorgang flexibler und mächtiger: Zunächst wird überprüft, ob Anforderung und Beschreibung überhaupt aus gemeinsamen Dienstontologien entstammen. Neben den Ein- und Ausgabefiltern werden unter anderem logische Einschränkungen und die UDDI-Beschreibungstexte untersucht. Zudem wird die Abgleichzeit durch eine aufwändige Vorverarbeitung der Vererbungskonstellationen aller Dienstverzeichnis-Einträge innerhalb ihrer Dienstontologie reduziert.

Der Vorteil von OWL-S als Grundlage für eine Dienstbeschreibung im Vermittlungsprozess ist seine Ausdrucksmächtigkeit. Nicht nur, dass die OWL-basierten Beschreibungen gut mit dem ebenfalls OWL-basierten CMPlus (Kapitel 5) harmonieren. Im Gegensatz zu üblichen Dienstbeschreibungen, die sich auf (Attribut,Wert)-Paare stützen, sind damit auch die für Kontextinformationsklassen so wichtigen Vererbungs- und Äquivalenzbeziehungen abbildbar.

WSMF

Dietrich Fensel und Christoph Bussler [uCB02] haben einen zu OWL-S ähnlichen Ansatz vorgestellt, das *Web Service Modelling Framework* (WSMF). Dieses kennt vier Hauptkomponenten, die im *Web-Services-Modeling-Ontology*-Projekt [FAC+05] implementiert werden:

Ontologien Darin werden alle Bezeichnungen, die von den übrigen WSMF-Komponenten benutzt werden können, semantisch definiert. OWL-Ontologien sind zwar dafür grundsätzlich auch verwendbar, als native Sprache ist dafür jedoch die *Web Service Modeling Language* [dBLK+05] vorgesehen.

Verzeichnis von Zweckbeschreibungen Es ist die Philosophie von WSMF, die Beschreibung des Zwecks („*goal*“) eines Dienstes von der eigentlichen, der techni-

schen Dienstbeschreibung zu trennen. Die Argumentation: Es herrsche eine n:m-Beziehung zwischen diesen. Der gleiche Dienst (zum Beispiel amazon.de) kann genauso verschiedenen Zwecken dienen (zum Beispiel Kauf eines Buches versus Beschaffung von bibliographischen Informationen) wie es verschiedene Dienste für den selben Zweck gibt. Ausgedrückt wird der Zweck eines Dienstes mit logischen Vor- und Nachbedingungen (ähnlich wie der logische Beschreibungsteil von UDDI).

Web-Service-Beschreibungen Darunter sind die verschiedensten Beschreibungselemente zu verstehen, darunter Eingabe und Ausgabe, wiederum Vor- und Nachbedingungen, Zugriffspunkt, Schnittstellen und so weiter. Die Unterteilung in Zweckbeschreibung und *Web-Service*-Beschreibung, die sich trotzdem teilweise entsprechen, ähnelt sehr der Unterteilung von OWL-S in Dienstprofil und Dienstmodell.

Vermittler Im WSMF wird nicht von einem homogenen Datenmodell, oder einer homogenen Anfragesprache ausgegangen. Stattdessen wird bereits von Beginn an mit verschiedensten Vermittlern gearbeitet, die die Heterogenität überbrücken sollen. Dabei gibt es vier Arten von Vermittlern („*mediators*“): Solche, die entweder zwischen Ontologien, zwischen Zweckbeschreibungen oder zwischen *Web Services* übersetzen und solche, die Zweckbeschreibungen auf *Web-Service*-Beschreibungen abbilden können.

Schon allein wegen seiner Ähnlichkeit zu OWL-S, aber auch wegen des mächtigen Anspruchs, verschiedenste Dienstbeschreibungen und Anfragesprachen integrieren zu wollen, ist WSMF ein sehr interessanter Ansatz. In der daraus resultierenden Komplexität liegt aber auch seine Schwäche. Insbesondere das Verlassen der OWL-Welt ist zu bedauern, da OWL dabei ist, sich zum vorherrschenden Standard aufzuschwingen und bereits eine Vielzahl von Werkzeugen und Implementierungen existiert. Insofern ist leicht nachvollziehbar, dass die WSMI OWL-S und das WSMF verschmelzen möchte.

CoBrA und FIPA

In der *Context Broker Architecture* (CoBrA) [Che04c] (siehe auch Abschnitt 3.4) wird die Welt in meist räumlich eng begrenzte Domänen unterteilt, denen Broker zugeordnet werden, um Kontext zu aggregieren und auszutauschen. Zur Inter-Domänen-Kommunikation wird ein Java-Framework benutzt, das auf der *Foundation for Intelligent Physical Agents* (FIPA) [fip04, fip] beruht, einem wichtigen Standard auf dem Gebiet der Intelligenten Autonomen Systeme. Näheres zur Implementierung wird nicht erwähnt und auch die Vermittlung von Kontextinformationsdiensten wird nicht näher adressiert, sondern es wird lediglich auf die FIPA-Plattform verwiesen. Die Möglichkeit, den Austausch zwischen Agenten und Kontextquellen über stärker akzeptierte Standards wie *Web Services* und SOAP zu erledigen, wird zwar angedacht, aber nicht näher spezifiziert.

SLM

Tao Gu et al. haben in ihrem *Service Location Manager* (SLM) [GQYP03] einen Broker-Ansatz vorgestellt, der mit verschiedenen Arten von Dienstbeschreibungen umgehen kann: Einerseits sind DAML-S-Beschreibungen (die frühe Version von OWL-S) möglich, andererseits sind auch Java/Jini-Objekte oder Listen mit (Attribut,Wert)-Paaren erlaubt. Entsprechend wird jeweils entweder ein semantischer Abgleich unter Zuhilfenahme einer DAML-Inferenzmaschine erledigt, oder der normale Matching-Mechanismus aus der Jini-Architektur eingesetzt. Die Kommunikation läuft über Java und RMI. Die SLM-Server selbst sind untereinander baumartig organisiert.

ContextWeaver

Norman H. Cohen, Paul Castro and Archan Misra von IBM entwickeln mit *ContextWeaver* [CCM04, CCM05] ein System, das ziemlich genau den Anforderungen der Suche nach Kontextinformationsdiensten entspricht: Deklarativ soll aus der Art der gesuchten Kontextinformation direkt auf passende Datenlieferanten geschlossen werden können, wobei zusätzliche Einschränkungen (etwa Aktualität oder Granularität der Information) außerdem als Filter dienen. *ContextWeaver* stützt sich dabei in erster Linie auf XML und XQuery [xqu05]. Dem Ansatz fehlt ein präzises Informationsmodell, wobei die Autoren angeben, OWL als Grundlage für Ontologien zu misstrauen, da die Ontologieerstellung sehr arbeitsintensiv ist. Ihre Hierarchie von Kontextanbieterarten sehen sie als „Primitivontologie“.

Solar und INS

Dem Paradigma, Ressourcen und Dienste nur nach ihrem Typ zu benennen, folgt auch das *Intentional Naming System* (INS) von William Adjie-Winoto et al. [AWSBL99]. Statt physikalische Adressen beschreiben Namen jetzt Intentionen in Form von hierarchischen Attribut-Wert-Paaren. Mit Hilfe eines selbstorganisierenden Overlay-Netzwerkes von Namensauflösern werden auf diese Art adressierte Nachrichten an die richtige Stelle geleitet. Die Solar-Architektur (siehe Abschnitt 3.4) baut auf dem INS auf. Beschreibungen und Ereignisse werden dabei nicht in XML sondern in einer eigenen Syntax kodiert.

6.2.4 Analyse

Ansätze wie *Liquid* oder *ContextShadow* sind wenig geeignet für offene Systeme, die höhere Anforderungen an Dynamik und Generik stellen. Von den übrigen sind OWL-S und die Dienstmodellierung innerhalb des WSMF diejenigen, mit denen ohne Weiteres die ausdrucksstärkste Semantik für Dienstbeschreibungen erreicht werden kann. Beide sind zudem die beiden Forschungsprojekte mit den meisten Anstrengungen zur Weiterentwicklung.

Auf Grund der Nähe zur Kontextmodellierung CMPlus ist im Rahmen dieser Arbeit OWL-S der Vorzug zur Verwendung als Modellierung einer Dienstbeschreibung gegeben worden. Da OWL-S selbst keine eigene Vermittlungslösung enthält, wird dazu wie vorgestellt das UDDI-Protokoll verwandt. Die folgende, neue Modellierung ist geeignet für die Beschreibung von Kontextinformationsdiensten.

6.2.5 CISP-Beschreibung eines Kontextinformationsdienstes

In diesem Abschnitt wird CISP beschrieben, das steht für *Context Information Service Profile* und ist eine semantische Dienstbeschreibung für Kontextinformationsdienste, die auf OWL-S aufbaut. Mit einem Dienstverzeichnis, das CISP-Beschreibungen enthält, ist ein Kontextvermittler in der Lage, für eine Kontextanforderung geeignete Kontextinformationsdienste zu finden. Diese Art der Dienstsuche ist deklarativ, da keine Informationen über den physischen Dienst zur Suche verwandt werden, sondern lediglich das von ihm zu erbringende Ergebnis. Näheres ist auch in einer der Vorarbeiten von Johannes Mathes [Mat05] nachzulesen.

Abbildung 6.5 zeigt den Aufbau von CISP graphisch: Kern des Profils ist die Klasse „ContextInfoServiceProfile“ als Spezialisierung der OWL-S-Klasse „Profile“. Die statischen Merkmale eines Kontextinformationsdienstes sind (mit CMPlus modelliert) in Spezialisierungen der OWL-S-Klassen „Input“ und „Output“ verpackt. Sie werden bei einer Kontextsuche zum Abgleich herangezogen und sind mit CMPlus modelliert. Im Einzelnen:

- „EntityClassInput“ beinhaltet die CMPlus-Entitätsklasse oder Entitätsklassen, zu denen dieser Dienst Kontextinformationen liefert.
- „ContextInfoClassOutput“ gibt an, welche CMPlus-Kontextinformationsklasse der Dienst liefern kann. Dies ist die zentrale Information, die gewissermaßen den Typ des Kontextinformationsdienstes bezeichnet.
- „InfoStructureClassOutput“ zeigt, in welcher CMPlus-Datenstruktur die Kontextinformation angegeben wird.

Alle Informationen, die für den Abgleich mit einer Kontextanfrage notwendig sind, aber dynamisch sein können (aber auch statisch sein dürfen), sind in Parametern als Spezialisierung der OWL-S-Klasse „ServiceParameter“ angegeben. Dazu gehören:

- Der Parameter „CoveredEntitiesParameter“ gibt an, zu welchen Entitäten (Instanzen) der Dienst aktuell Kontextinformationen liefert.
- Alle Qualitätskriterien, die der Dienst bekannt gibt, sind als „QoCParameter“ angegeben. Dabei ist nicht sicher, dass der Dienst zu allen in der Kontextanfrage spezifizierten Qualitätsparametern Angaben macht.

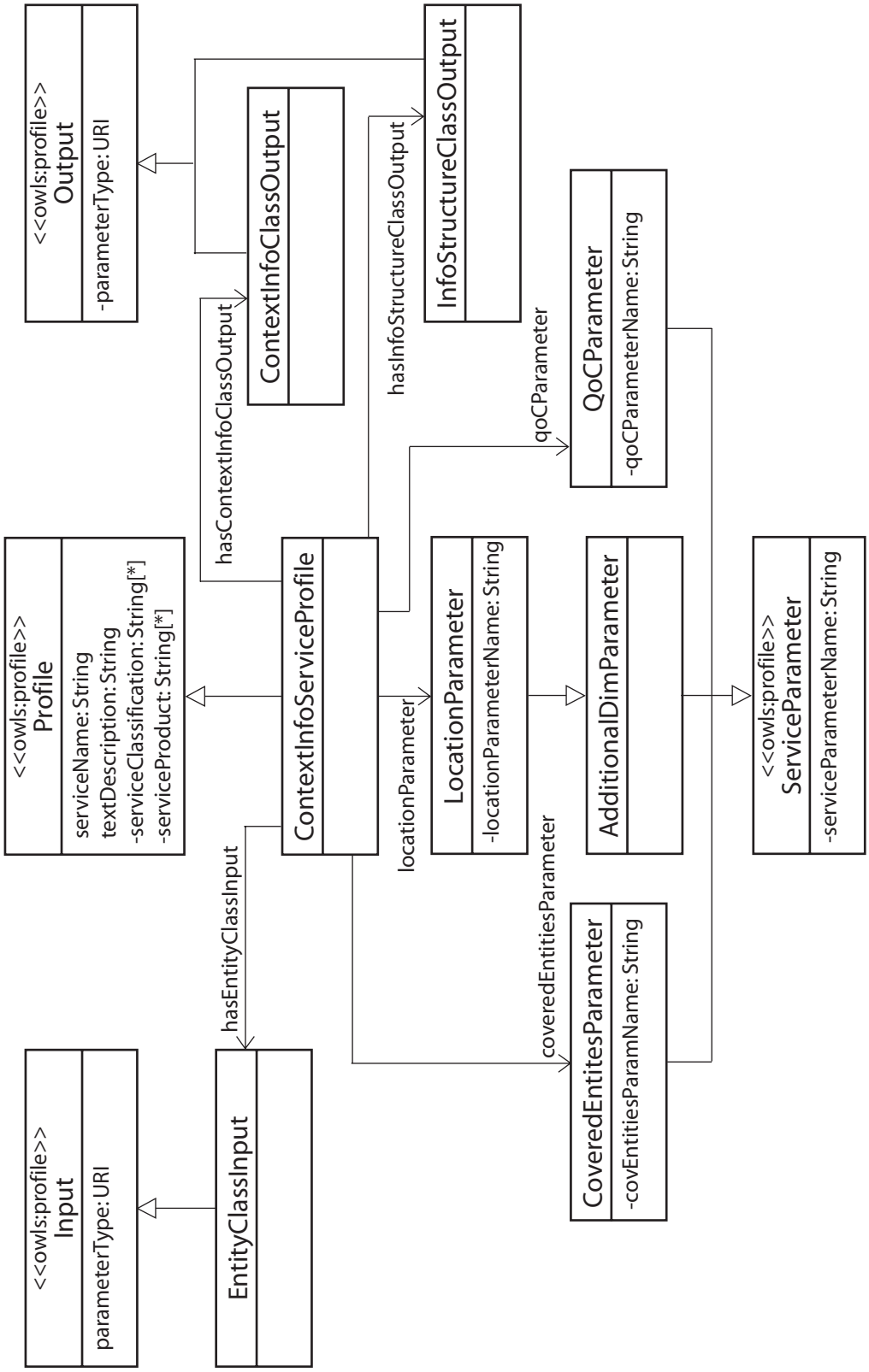


Abbildung 6.5: CISP in der Übersicht.

- Je nach Anwendungsgebiet kann es sinnvoll werden, weitere Suchdimensionen als eigene Parameter zu definieren. Um CISP so generisch wie möglich zu halten, gibt es eigens dafür die Oberklasse „AdditionalDimParameter“. In Abbildung 6.5 ist beispielsweise für den Ort als zusätzliche Suchdimension die Klasse „LocationParameter“ definiert worden. Der Ort kann auch dann wichtig zum Auffinden eines geeigneten Kontextinformationsdienstes sein, wenn weder die Eingabewerte (Entitäten) noch die Ausgabewerte (Kontextinformationen) Ortsinformationen sind.

Die Parameterwerte des selben Dienstes können von ganz unterschiedlicher Dynamik sein. Deswegen wird dafür ein flexibles Stub-Konzept eingeführt, das jeden Parameter individuell behandelt. Dabei kann für jeden einzelnen Parameter entschieden werden, ob ein konkreter Wert hinterlegt wird (bei Parameterwerten, die statisch sind oder langlebiger als die Registrierung im Dienstverzeichnis) oder ein Stub-Objekt, das auf einen Hilfsdienst zeigt, welcher den tatsächlichen Wert liefert. Die Stubs verweisen dabei entweder auf eine OWL-S-Beschreibung oder auf einen eindeutigen Identifikator einer UDDI-Beschreibung (siehe Abbildung 6.7; zum Einsatz von UDDI siehe auch Kapitel 8). Dabei gibt es zwei Kategorien von solchen Hilfsdiensten:

- Pull-basierte Parameter-Lieferdienste geben auf Anfrage direkt die entsprechenden Werte zurück.

- Bei push-basierten Parameter-Lieferdiensten können sich interessierte Dienste anmelden, um die entsprechenden Daten zu abonnieren. Dadurch können sich zum Beispiel Kontextvermittler, die über einen bestimmten Zeitraum immer wieder gleichartige Kontextinformationen anfordern, von einer Menge von Kontextinformationsdiensten auf dem Laufenden halten lassen.

Durch das Stub-Konzept ist es außerdem möglich, entitätsabhängige Parameter zu gestalten, während konkrete Werte nur entitätsunabhängig für den gesamten Dienst gelten können. Ist ein durch einen Stub gekennzeichneter Parameter entitätsabhängig, verlangt er als Eingabe die entsprechende CMPlus-kodierte Entität. Eine Besonderheit stellt dabei der Parameter dar, der angibt, welche Entitäten durch den Dienst abgedeckt werden. Hier gibt es immer zwei Möglichkeiten: Entweder kann (natürlich entitätsabhängig) über den „EntityStub“ abgefragt werden, ob eine bestimmte Entität zu den abgedeckten gehört. Oder es kann (entitätsunabhängig) über den „EntityEnumerationStub“ eine Liste aller abgedeckten Entitäten angefordert werden.

6.2.6 Ablauf der Vermittlung

Im Prozess der Suche nach einem geeigneten Kontextinformationsdienst nimmt der Kontextvermittler selbst die Rolle des Anfragers ein. Mit den eben beschriebenen Techniken sieht ein beispielhafter Ablauf der Vermittlung wie in Abbildung 6.8 aus:

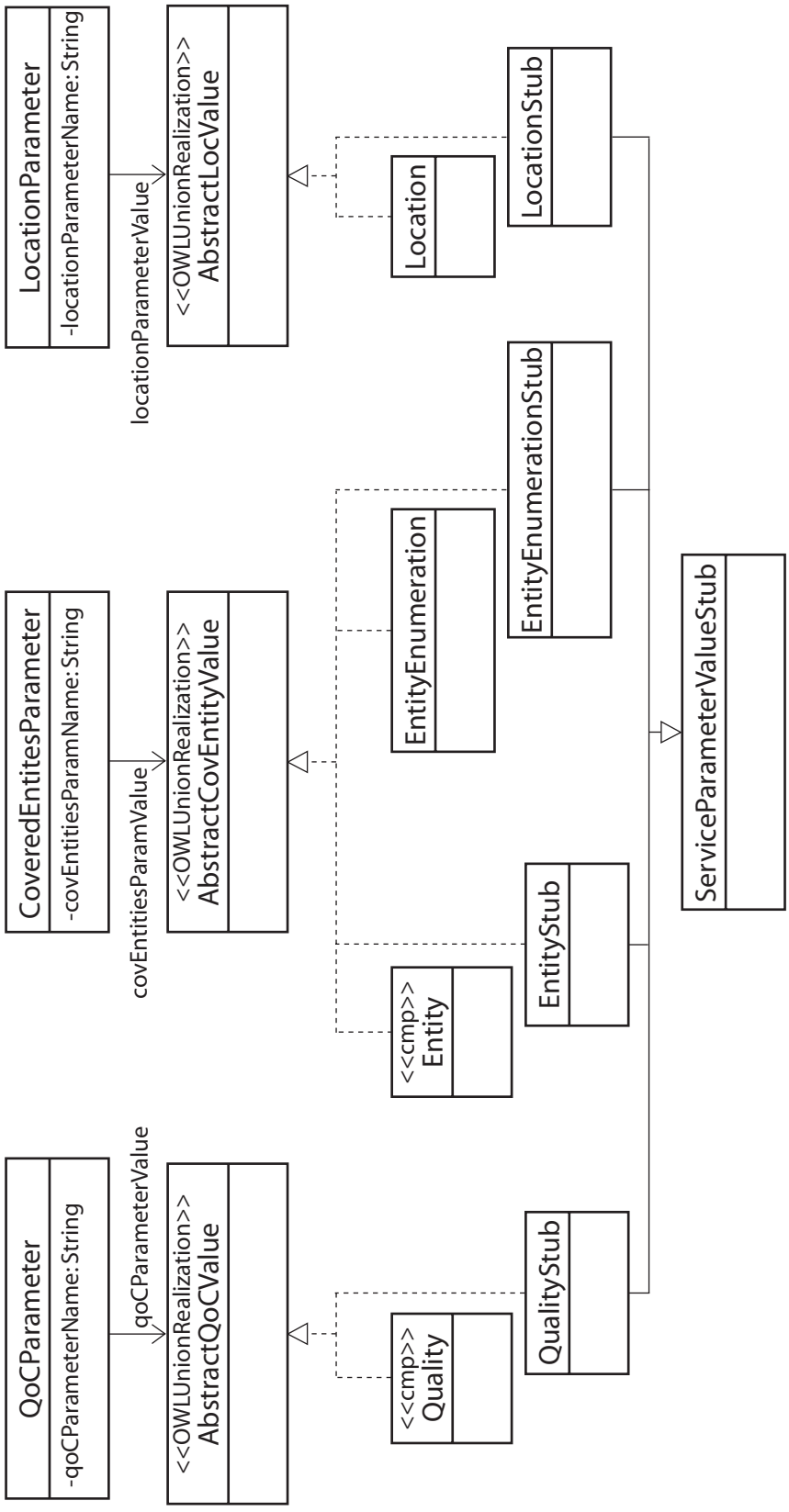


Abbildung 6.6: Ausgestaltung der CISP-Parameter mit Werten oder Stubs.

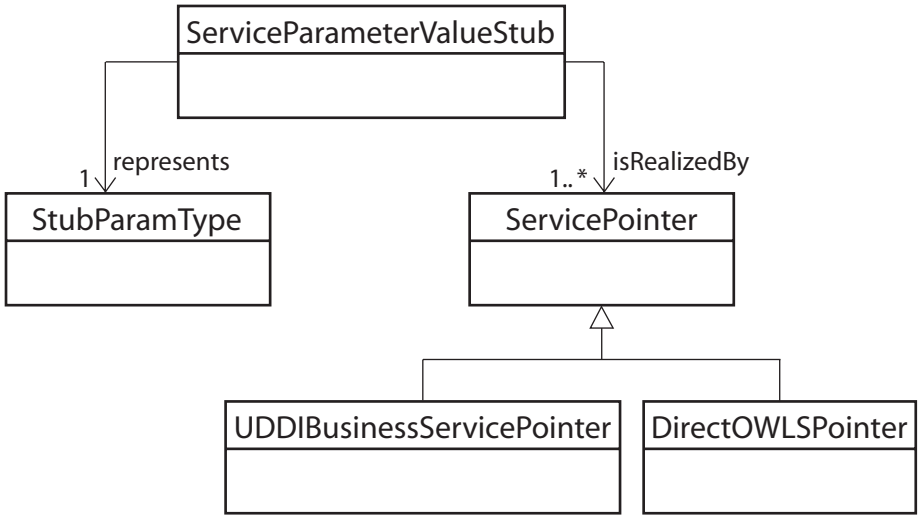


Abbildung 6.7: Ein Stub verweist auf einen Lieferdienst.

Ein Kontextinformationsdienst instanziiert einen pull-basierten und einen push-basierten Parameter-Lieferdienst. Diese registrieren sich zuerst im Dienstverzeichnis, da der Kontextinformationsdienst die Referenzen für die eigene Dienstbeschreibung benötigt, die er anschließend selbst dort hinterlegt. Danach aktualisiert der push-basierte Lieferdienst bei Bedarf seinen Parameterwert des Kontextinformationsdienstes. Auf eine Anfrage des Kontextvermittlers an das Dienstverzeichnis erhält dieser eine Menge von Dienstbeschreibungen von Kandidaten, die seine Anforderungen erfüllen könnten. Um dies zu verifizieren, fragt der Kontextvermittler die entsprechenden pull-basierten Lieferdienste an, um die aktuellen Werte der dynamischen Parameter in Erfahrung zu bringen. Anschließend fragt er bei einem geeigneten Kontextinformationsdienst direkt die von ihm gesuchte Kontextinformation an, die dieser (im Erfolgsfall) zurück sendet. Da der Kontextvermittler in Zukunft viele ähnliche Kontextanfragen erwartet, merkt er sich diesen samt seiner Dienstbeschreibung und meldet sich bei dem push-basierten Lieferdienst des Kontextinformationsdienstes an, um auch die dynamischen Parameter der Dienstbeschreibung aktuell zu halten. In Zukunft erhält er ebenso push-basierte Aktualisierungen.

Bislang ist der vorgestellte Mechanismus zur Vermittlung von Kontextinformationsdiensten ganz auf die Modellierung von Kontextinformationen mit CMPlus abgestellt. Im folgenden Abschnitt wird nun erörtert, wie auf Kontextquellen für Informationen anderer Modellierungsarten in diesen Prozess mit eingebunden werden können.

6.3 Integration von fremden Kontextinformationsdiensten

Wie schon in Abschnitt 5.3 angeführt, ist es wichtig, auch mit anderen Kontextmodellierungen kompatibel zu sein, um die Zahl verfügbarer Kontextinformationsdienste zu ma-

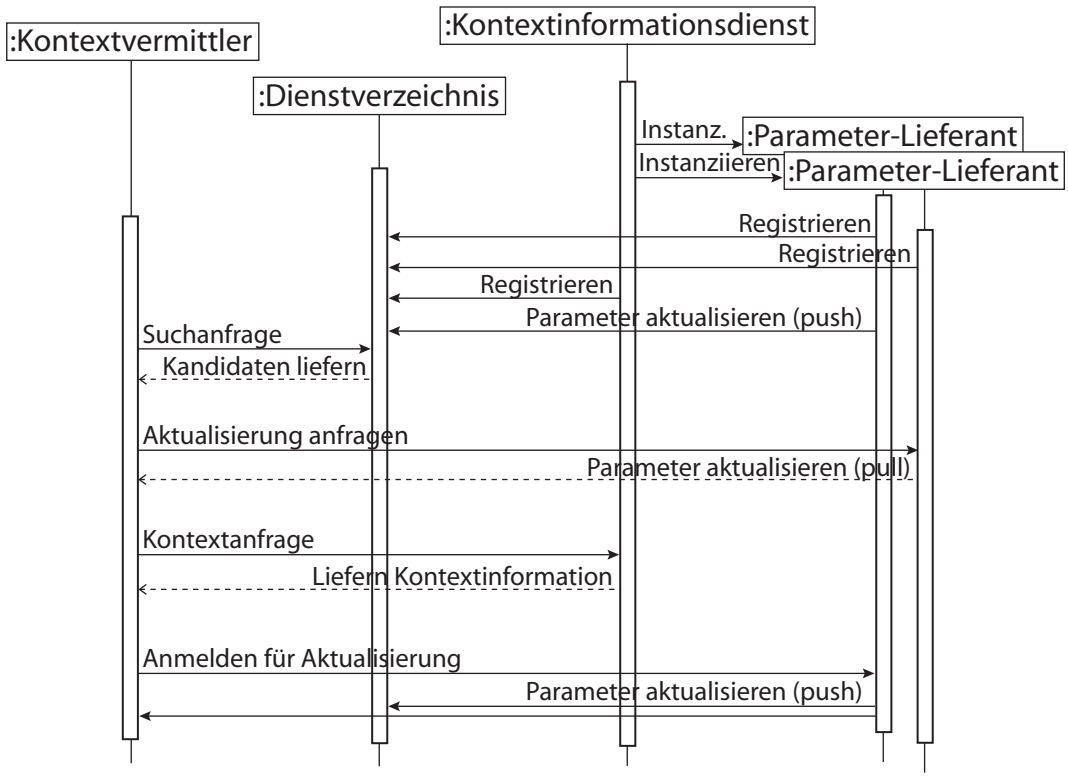


Abbildung 6.8: Suche nach einem geeigneten Kontextinformationsdienst.

ximieren. Zunächst wird nun die Frage erörtert, an welcher Position diese Integration geschehen soll (siehe dazu auch [Fra05]).

Für diese Betrachtungen wird von n verschiedenen Kontextinformationsdiensten und m Modellierungsarten ausgegangen, wobei $n \gg m$. Abbildung 6.9 illustriert dieses Szenario mit drei Modellierungsarten und sechs Kontextinformationsdiensten, wobei natürlich für die Integrationsüberlegungen lediglich der Kontextvermittler aus der CMPlus/CISP-Modellierungswelt von Belang ist. Möglich ist eine Integration auf Seiten des Kontextvermittlers, auf Seiten des Dienstverzeichnisses oder auf Seiten des Kontextinformationsdienstes. Eine Integration auf Seiten des anfragenden kontextsensitiven Dienstes scheidet als Nicht-Integration selbstverständlich aus. Die Möglichkeiten im Einzelnen:

Positionierung beim Kontextinformationsdienst In diesem Fall entscheidet der Anbieter des Kontextinformationsdienstes selbst, in welchen Modellierungen er seine Kontextinformationen anbietet. Die Implementierung ist sehr einfach: Es wird jeweils eine eigene Schnittstelle für jede Modellierung zur Verfügung gestellt. Damit sind keine weiteren Integrationsmaßnahmen im bisherigen Bereitstellungsprozess notwendig. Da der Anbieter die Semantik seiner Kontextinformationen am besten kennt, fällt ihm die korrekte

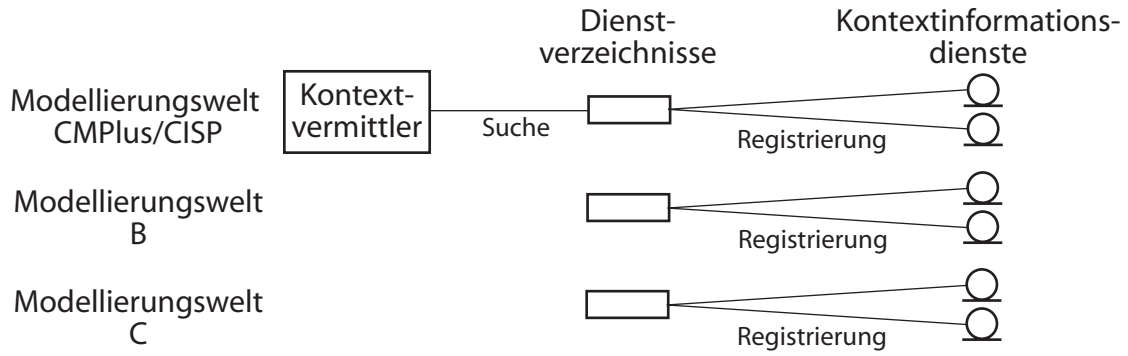


Abbildung 6.9: Integration von Kontextinformationsdiensten fremder Modellierung.

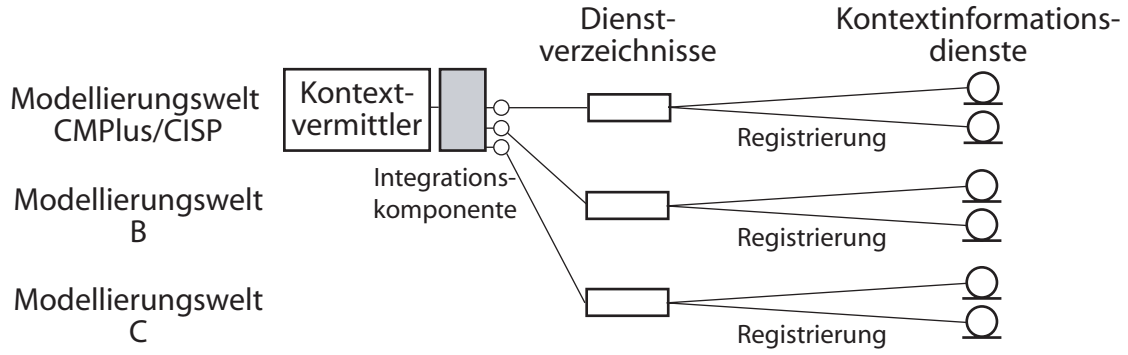


Abbildung 6.10: Integrationskomponente beim Kontextvermittler.

Adaption in die jeweilige Modellierung am leichtesten. Dennoch ist der Gesamtaufwand immens, da die Adaption in alle Modellierungen einzeln für jeden Kontextinformationsdienst erledigt werden muss. Zudem bleiben Kontextinformationsdienste, deren Anbieter diese Arbeit nicht erledigen, für den Kontextvermittler unerreichbar. Integration auf Seiten des Kontextinformationsdienstes ist also nur als zusätzlicher Aufwand sinnvoll, nie als Grundlage einer generellen Integrationslösung.

Positionierung beim Kontextvermittler Geschieht die Integration beim Kontextvermittler selbst, so muss der Kontextvermittler im Anfragefall mehrere Kontextanfragen absetzen, jede davon in dem Format der jeweiligen zu integrierenden Modellierung. Die Abbildung der Resultate auf die eigene Modellierung, die notwendig ist, um sie vergleichbar zu machen, muss ebenfalls zur Laufzeit geschehen. Diese Möglichkeit der Integration erfordert allerdings keine Änderungen auf Seiten der Anbieter von Kontextinformationsdiensten, sie ist transparent für sie.

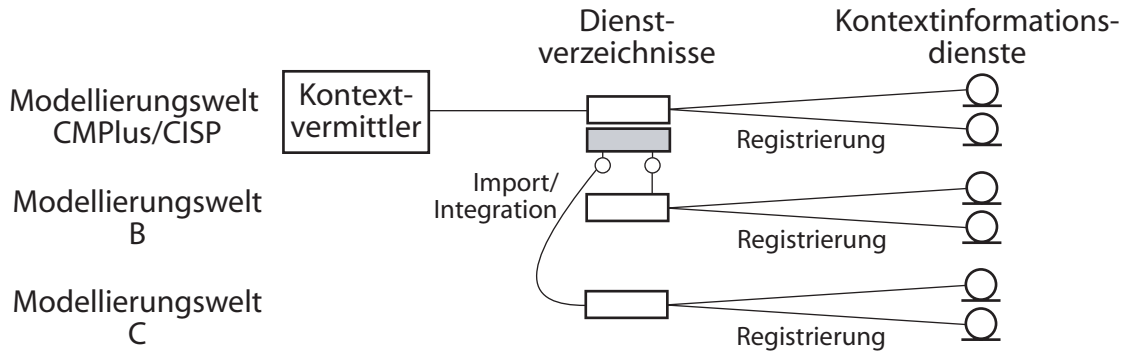


Abbildung 6.11: Integrationskomponente zwischen Dienstverzeichnissen.

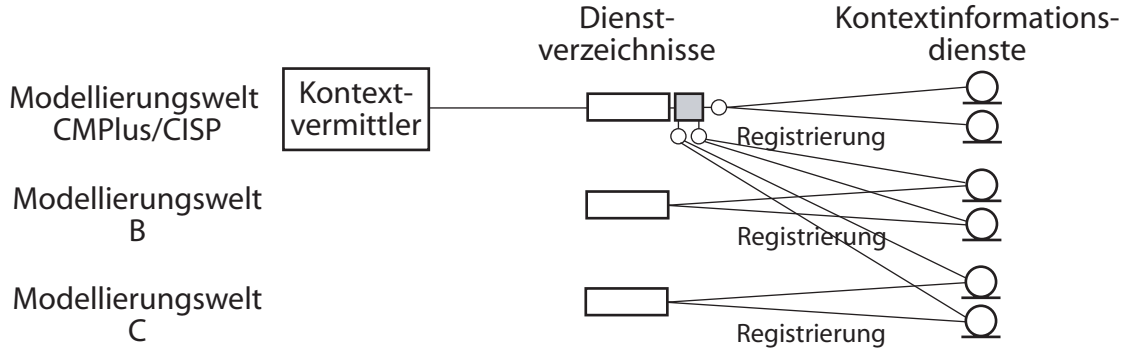


Abbildung 6.12: Integrationskomponente zwischen Dienstverzeichnis und Kontextinformationsdiensten.

Positionierung beim Dienstverzeichnis Wird die Integration auf Seiten des Dienstverzeichnisses vorgenommen, so eröffnen sich zwei Alternativen: Entweder das Dienstverzeichnis der eigenen Modellierung bietet den Kontextinformationsdiensten unterschiedliche Schnittstellen für die jeweils zu integrierenden Modellierungswelten an, damit diese sich auch dort registrieren können und die Dienstbeschreibungen intern auf eine einheitliche Beschreibung abgebildet werden kann (siehe Abbildung 6.12). Oder das Dienstverzeichnis importiert die Daten direkt aus den Dienstverzeichnissen der anderen Modellierungswelten, was den Vorteil hat, dass diese Lösung für die Kontextinformationsdienste (im englischen Sinne) transparent ist und diese sich nicht mehrfach registrieren müssen (siehe Abbildung 6.12).

Während die Positionierung bei den Kontextinformationsdiensten aus den beschriebenen Gründen ausscheidet, sind Integrationskomponenten sowohl beim Kontextvermittler als auch beim Dienstverzeichnis grundsätzlich möglich. Sinnvoller ist es jedoch, die Integration, sprich die Umwandlung der Beschreibungen von Kontextinformationsdiensten in die eigene Modellierung, direkt beim Dienstverzeichnis anzusiedeln. So kann die Um-

wandlung nämlich bereits bei oder nach der Registrierung erfolgen („*on registration*“) und nicht erst zur Laufzeit, also während der Kontextanfrage („*on demand*“). Außerdem ist so das Dienstverzeichnis die einzige Komponente im Dienstsuchprozess, die mit der Integration befasst ist, während der Integrationsschritt für den Kontextvermittler und die Kontextinformationsdienst transparent ist. Um diesen Vorteil auch wirklich zu nutzen, sollte es nur dann ein Dienstverzeichnis geben, das mehrere Modellierungen gleichzeitig erfasst (Abb. 6.12), wenn sichergestellt ist, dass die Kontextinformationsdienste ohne Weiteres ihre Dienstbeschreibungen dort einstellen. Ansonsten ist die in Abbildung 6.11 dargestellte Variante vorzuziehen.

6.3.1 Schritte der strukturellen, ontologischen und tatsächlichen Integration

Der erste Integrationsschritt, der strukturelle, ist einer, der nicht automatisiert werden kann. Er muss händisch von einem Entwickler erledigt werden. Dazu gehört zunächst die Prüfung, ob eine bestimmte Modellierungsart überhaupt integrationsfähig ist. Das hängt davon ab, ob sie einerseits formal genug ist, und andererseits ob sie die grundlegenden Strukturelemente enthält, die in der eigenen (CMPlus- und CISP-)Darstellung minimal verlangt werden. Ist dies der Fall, können die strukturelle Abbildungsvorschriften aufgestellt werden. Etwa: Mit welchen Modellierungselementen wird eine Entität dargestellt und wie wird ihre Identität ausgedrückt?

Hat man die Abbildung der Strukturelemente aufeinander, ist der nächste Schritt die ontologische Integration (Wache et al. [WVV01] nennen diese auch “semantisch“). Das bedeutet, dass die Beschreibungsklassen der fremden Modellierung auf die Ontologieklassen der eigenen Modellierung abgebildet werden, sofern die entsprechenden bereits existieren. Andernfalls sind neue, eigenen Ontologieklassen entsprechend zu erstellen. Für einen derartigen Schritt gibt es in der allgemeinen Forschung zur Abbildung von Ontologien aufeinander viele Ansätze zur Automatisierung, die unter anderem auf linguistischen, strukturellen oder statistischen Ähnlichkeiten beruhen (siehe dazu auch [WVV01, CFLGG+04, OL03, KS05]). Für den Spezialfall der Kontextontologien hält der Autor jedoch wenig von solch stark fehlerbehafteten Verfahren. Der Aufwand der einmaligen händischen (und damit semantisch korrekten) Abbildung einer Ontologie wird durch die potenziell immens hohe Zahl der (nunmehr automatisierten) Anwendungsfälle mehr als gerechtfertigt.

Sind diese beiden Integrationsschritte wie beschrieben händisch erledigt worden, kann die tatsächliche Integration automatisch erfolgen. Im Fall der Registrierung von Kontextinformationsdiensten bedeutet das, dass es möglich ist, aufgrund der zuvor festgeschriebenen Abbildungsvorschriften „*on registration*“ die jeweilige Dienstbeschreibung in die eigene Modellierung (CISP) zu übersetzen. Dabei geschieht zweierlei. Zum einen wird in der Integrationsschicht ein CISP-Stub angelegt (nicht zu verwechseln mit den Parameter-Stubs in den vorangegangenen Abschnitten), der die Schnittstelle eines CMPlus-Kontextinformationsdienstes beinhaltet und intern über die entsprechenden ontologischen Abbildungsfunktionen den tatsächlichen Kontextinformationsdienst der fremden Modellierung aufruft. Zum anderen wird eine CISP-Beschreibung dieses Stubs in das Dienstverzeichnis

aufgenommen. Für den Kontextvermittler ist der Dienst nun nicht mehr von einem normalen CMPlus-Kontextinformationsdienst zu unterscheiden.

Hier bewährt sich auch das Stub-Konzept von CISP für dynamische Parameter: Es ist einfacher, die unterschiedlichsten Konzepte zur Registrierung dynamischer Dienstbeschreibungen der jeweiligen Modellierungswelten zu integrieren, da die Beschreibung von Kontextinformationsdiensten in Teile unterschiedlicher Dynamik modularisiert werden. Hilfreich ist auch der Umweg über Parameter-Stubs, die im Normalfall auf Hilfslieferdienste verweisen.

6.4 Zusammenfassung

In diesem Kapitel ist dargelegt worden, wie ein Kontextvermittler eine Kontextinformation beschaffen kann, die ein externer Kontextinformationsdienst bereitstellt. Dafür ist zunächst herausgearbeitet worden, wie auf Basis von CMPlus eine Anfrage nach einer Kontextinformation ausgedrückt werden kann. Diese Modellierung geht an Ausdrucksstärke und Formalität über bestehende Ansätze hinaus. Insbesondere ist sie dafür geeignet, die unterschiedlichsten Qualitätsbedingungen auszudrücken. Dem wurde eine neue Möglichkeit gegenübergestellt, einen Kontextinformationsdienst geeignet zu beschreiben, um diese Beschreibung mit der Anfrage abgleichen zu können. Dabei ist die Frage der Skalierung und Verteilung des Dienstverzeichnisses ausgespart worden, da dies ein klassisches Problem ist, das nicht nur die Verzeichnisse von Kontextinformationsdiensten betrifft und entsprechende Ansätze bereits existieren und weiter entwickelt werden. Schließlich ist noch ein Konzept zur Integration von Kontextinformationsdiensten aus fremden Modellierungswelten erläutert worden. Dies ist wichtig, um die Menge der verfügbaren Kontextinformationsdienste zu maximieren. Nur so sind die Netzeffekte zu erreichen, die die Kooperation in offenen, ubiquitären Systemen verspricht.

7 Ausweichstrategien zur Bereitstellung von Kontextinformationen

> Insofern sich die Gesetze der Mathematik auf die Wirklichkeit beziehen, sind sie nicht sicher. Und insofern sie sicher sind, beziehen sie sich nicht auf die Wirklichkeit.
>
> *(Albert Einstein)*

In dynamischen Systemen wie offenen *Ubiquitous-Computing*-Systemen geschieht die Vermittlung von Kontextinformationen nach dem *best-effort*-Prinzip. Das heißt, die Gefahr des Scheiterns ist groß (bei der Suche, der Vermittlung oder der Übertragung der Informationen); etwa weil es keinen Kontextinformationsdienst für die gewünschte Kontextinformation gibt, oder weil ein solcher Dienst nicht erreichbar ist, oder weil es Probleme bei der Übermittlung gibt.

Wie ein Kontextvermittler unter gewissen Umständen eine angeforderte Kontextinformation bereitstellen kann, obwohl die direkte Beschaffung von einem Kontextinformationsdienst missglückt ist, beschreibt dieses Kapitel. Zunächst werden verwandte Arbeiten und Grundlagen erläutert, zu denen unter anderem Fuzzy-Logik in Abgrenzung zur probabilistischen Logik gehört und deren Anwendung in Bayesschen Netzen. Ein neu entwickelter Algorithmus zur Bildung von Konstruktionsbäumen beschreibt, wie diese Strategien kombiniert werden können.

7.1 Grundlagen und verwandte Arbeiten

7.1.1 Ersetzungen in Kontextkompositionsgraphen

Zu der bereits in Abschnitt 6.1.1 erläuterten Anfragesprache von Kontextinformationen, die ihrerseits auf die in Kapitel 5 entwickelte Kontextmodellierung zurückgreift, wurde eine Kompositionssprache konzipiert [Kra03, BKLPS04, Buc05] und weiterentwickelt, die Grundlage einiger der in diesem Kapitel erläuterten Strategien sein wird. Sie soll deshalb an dieser Stelle vorgestellt werden.

Abbildung 7.1 zeigt das Beispiel eines Kontextkompositionsgraphen, mit dem es möglich ist, verschiedene Kontextanfragen hintereinander zu schalten und zu parallelisieren. Dabei wird als Eingabe ein Identifikator für den Benutzer mit übergeben. Für ihn wird im ersten Knoten eine Kontextanfrage nach seinem Ort formuliert. Der Ort ist gleichzeitig

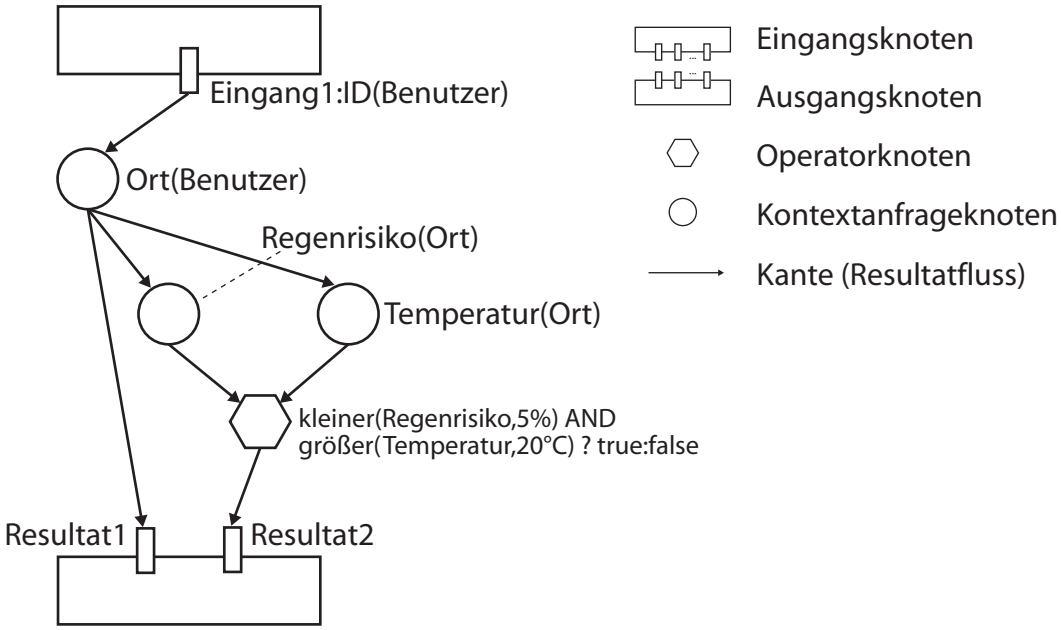


Abbildung 7.1: Ein Kontextkompositionsgraph nach [Kra03].

das erste Resultat des Graphen, das nach der Abarbeitung zurückgegeben wird. Zu diesem Ort werden zwei weitere Kontextanfragen zur Temperatur und zum Regenrisiko gestellt. Diese werden nach Anweisungen im Operatorknoten daraufhin untersucht, ob gutes Wetter herrscht. Die Antwort ist das zweite Rückgaberesultat. Formuliert wird der Graph in XML.

Statt einzelner können somit komplexe Kontextanfragen an einen Kontextvermittler gestellt werden, der dann selbsttätig aus niederwertigeren immer höherwertigere Kontextinformationen komponiert. Zwischenergebnisse, die den Abnehmer nicht interessieren, müssen nicht mehr übermittelt werden – was insbesondere von Vorteil sein kann, wenn diese Zwischenergebnisse aus Privatheitsgründen gar nicht übermittelt werden dürfen, die aus ihnen abgeleiteten Kontextinformationen dagegen schon. Aufgaben der Umwandlung der Repräsentationsformate und einfache Verarbeitungsaufgaben können somit ebenfalls an den darauf spezialisierten Vermittler ausgelagert werden. Während die Umwandlung der Repräsentationsformate (etwa von Grad Celsius nach Grad Kelvin) implizit bei Bedarf geschieht, werden die Verarbeitungen in so genannten Operatorknoten vorgeschrieben. Dazu gehören Tests auf die Ordnung, Aggregationen, Komplementbildungen und andere, einfache *built-in*-Funktionen. Dazu werden die Anweisungen der Operatorknoten nach dem Parsen in SWRL-Regeln überführt und von einem entsprechenden Inferenzsystem ausgewertet.

Sollten einzelne Kontextinformationen, die in Kontextanfrageknoten spezifiziert sind, nicht zu beschaffen sein, muss deswegen nicht die Abarbeitung des gesamten Graphen scheitern. Gerade weil für den Kontextanfrager nur das Resultat interessant ist, bietet

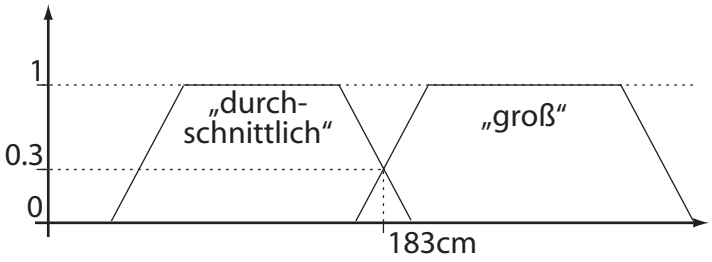


Abbildung 7.2: Im Diagramm gehört die Körpergröße „183cm“ sowohl der Menge der „großen“ als auch der Menge der „durchschnittlichen“ Größen an - jeweils mit einem Grad von 0.3.

sich die Möglichkeit, andere Kompositionspfade zu finden, die zu diesem Resultat führen können. Der teilweise Umbau von Kontextkompositionsgraphen ist deshalb einer der Strategien, die im Folgenden beschrieben werden.

Einen ähnlichen Ansatz gibt es auch von Guanling Chen und David Kotz [CK04], die in Solar (siehe Abschnitt 3.4) ein so genanntes *Dependency Management* einführen. Dabei werden die Komponenten namens „Planeten“ überwacht, die als Relaisstationen in festen Pfaden für die Verteilung von Sensordaten dienen. Fällt einer der Planeten aus, wird gemäß hinterlegter Ausfallprozedur zum Beispiel der Administrator benachrichtigt und wenn möglich werden die Sensorinformationen über einen anderen Planeten geleitet. Auch hier können sich also die Pfade automatisch restrukturieren. Allerdings betrifft dies hier nur Pfade, die für die Verteilung von Sensorinformationen fest etabliert worden sind.

7.1.2 Unscharfe Mengen und Fuzzy-Logik

Logische Aussagen, wie sie in Abschnitt 4.2 eingeführt worden sind, sind binär zu entscheiden: Entweder eine Aussage gilt oder sie gilt nicht. Dies entspricht so gar nicht den natürlichsprachlichen Aussagen der Menschen. Einen Mann empfindet man heutzutage als „groß“, wenn er „deutlich“ über 180 cm Körperhöhe misst. Mit den traditionellen logischen Mitteln, ließe sich „deutlich über 180 cm“ höchstens streng formalisieren als „$\geq$ 184 cm“. Ein Mann, der 183,9 cm groß ist, wäre demnach nur noch „durchschnittlich groß“. Der Mensch ist es aber nicht gewohnt, so scharfe Unterschiede zu ziehen, unter anderem, weil er dazu oft nicht in der Lage ist. Viel tauglicher ist dafür die Theorie der unscharfen Mengen („*Fuzzy Sets*“) [Got93, Lug01, Zim93, Rei00], die Lotfi A. Zadeh 1965 vorgestellt hat. Damit kann die Aussage über die Zugehörigkeit eines Elements zu einer Menge nicht nur binär, sondern mit dem Grad der Zugehörigkeit beantwortet werden. (Dies ist selbstverständlich nicht zu verwechseln mit dem Grad der Wahrscheinlichkeit einer Aussage.)

Definition (Unscharfe Menge, Mitgliedsgradfunktion)
Sei U eine Menge („Universum“) mit den Elementen u und $\mu_A : U \rightarrow [0,1]$ *eine totale Abbildung des Universums auf das abgeschlossene Intervall* $[0,1]$. *Dann ist* $A := \{(u, \mu_A(U))|$

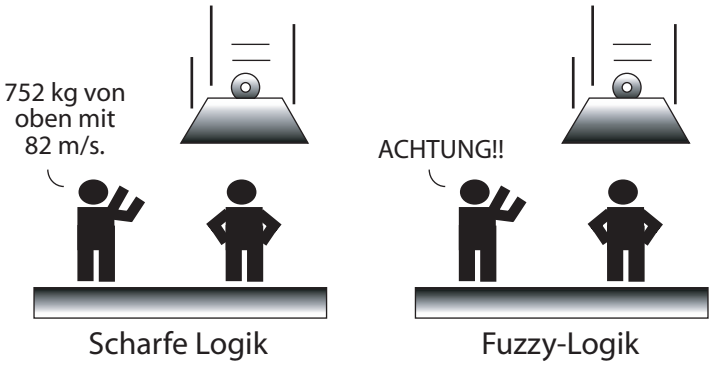


Abbildung 7.3: Manchmal sind Abschätzungen hilfreicher als exakte Informationen (nach einem Comic der Fachhochschule beider Basel).

$u \in U, \mu_A(u) \in [0,1]\}$ *eine unscharfe Menge („Fuzzy Set"). Die Funktion* $\mu_A(u)$ *gibt den Zugehörigkeitsgrad von* $u \in U$ *zu A an und heißt Mitgliedsgradfunktion („Membership Function"), Zugehörigkeitsfunktion oder charakteristische Funktion.*

Das Beispiel mit der Körpergröße ließe sich wie folgt damit ausdrücken: Sei U die Menge aller Menschen, A_g die Menge aller großen Menschen, A_d die Menge aller durchschnittlich großen Menschen und die Funktionen μ_{A_g}, μ_{A_d} entsprechend der Abbildung 7.2 definiert. Dann kann für einen Menschen u mit der Körpergröße 1,83 cm geschlossen werden, dass seine Zugehörigkeit zu der Menge der großen Menschen 0,3 beträgt und damit genauso groß ist wie die Zugehörigkeit zur Menge der durchschnittlich großen Menschen, da $\mu_{A_g}(u) = \mu_{A_d}(u) = 0,3$.

Anzumerken ist, dass statt des Intervalls [0,1] auch eine beliebige teilweise geordnete Menge verwandt werden kann. Besteht diese nur aus den Elementen 0 und 1, dann degeneriert A von der unscharfen Menge zu einer Booleschen Menge. μ muss natürlich nicht immer eine Trapezfunktion sein wie in Abb. 7.2. Möglich sind beliebige Kurven, zum Beispiel auch Gaußkurven oder Dreiecksfunktionen. Mit Hilfe geeignet definierter Operationen, die hier nicht näher betrachtet werden sollen, lassen sich auf unscharfen Mengen genauso die Konzepte der Vereinigung, des Durchschnitts und des Komplements realisieren, die es auch auf den herkömmlichen Mengen gibt.

Der Gebrauch des Begriffs „Fuzzy-Logik" hat sich kontinuierlich verändert. Inzwischen umfasst er die Theorie des abschätzenden Schließens und schließt manchmal auch die klassische Theorie unscharfer Mengen ein. Mit Hilfe unscharfer Mengen und Fuzzy-Logik lassen sich vage Konzepte wie „ungefähr" modellieren und verarbeiten. Doch die Unschärfe steckt nicht nur in der Semantik von Begriffen, auch unvollständiges Wissen kann als vage oder „unscharf" bezeichnet werden. Der darauf basierende Prozess des Schließens ist daher ebenfalls mit Unsicherheit belastet.

Die Analogie zu Kontextwissen ist offensichtlich: Oft sind Kontextinformationen nicht verfügbar, aufgrund der Messmethoden ungenau oder aufgrund der Übermittlungswege

nicht mehr ganz aktuell. Auf der anderen Seite geht es beim Schließen auf der Basis von Kontextwissen nicht um exakte Schlussfolgerungen – Ergebnisse mit einer geringen Unschärfe oder Abschätzungen reichen in der Regel als Ergebnisse aus. Fuzzy-Logik ist also zur Verwendung auf Kontextwissen geeignet.

Probabilistisches Wissen im Semantic Web

Fuzzy-Logik und probabilistische Aussagen ähneln sich in der Syntax: Beide benutzen Koeffizienten mit Werten zwischen 0 und 1, um statt binären Werten ein ganzes Werteintervall zur Verfügung zu haben. Die Semantik ist aber völlig unterschiedlich: Wo Fuzzy-Logik damit die Zugehörigkeit zu einer Menge modelliert, wird bei probabilistischen Aussagen damit deren Wahrscheinlichkeit beschrieben. Die Notwendigkeit, innerhalb des *Semantic Web* auch probabilistisches Wissen darstellen zu können, ist erkannt worden. Es gibt mehrere, ähnliche Ansätze dafür, von denen auch die Darstellung des Kontextwissens profitieren kann, wenn diese Ansätze einmal ausgereift sind. In der Regel handelt es sich um Erweiterungen von OWL, wie bei der *Knowledge Elicitation Environment for Probabilistic Event and Entity Relation (KEEPER)* [PA04, PFCA05] oder PR-OWL [CLL05, Cos05]

Auch eine Ebene unterhalb von OWL gibt es Bestrebungen, probabilistische Aussagen zu modellieren. So hat Yoshio Fukushige über seine Arbeit berichtet [Fuk05], Wahrscheinlichkeiten in RDF-Graphen darzustellen. Auch diese sollen in Bayessche Netze überführt werden, um Inferenz zu ermöglichen. Dazu führt er die RDF-Klassen „prob:Partition“, „prob:ProbabilisticStatement“, „prob:Clause“ und „prob:Probability“ ein und außerdem die RDF-Prädikate „prob:predicate“, „prob:hasProbability“, „prob:condition“, „prob:case“ und „prob:about“.

Kontextinformationen, die mit einer kleinen Unsicherheit behaftet sind, genügen in den meisten Anwendungsfällen. Es ist meist sinnvoller, wenn ein Nutzer in 20 Prozent der Anwendungsfälle die automatischen Voreinstellungen eines Dienstes korrigieren muss, als dass er ganz auf eine automatische Adaption verzichtet, wenn keine hundertprozentig sicheren Kontextinformationen als Grundlage für die Adaptionsentscheidung zu beschaffen sind. Selbstverständlich gibt es daneben auch kritische Dienste, bei denen eine einzige falsche Adaption fatale Auswirkungen haben kann, etwa bei der Bremssteuerung in Autos und Zügen. Diese können dann allerdings eigene, weit strengere Anforderungen an die Wahrscheinlichkeit des Zutreffens der dafür grundlegenden Kontextinformationen stellen.

7.1.3 Bayessche Netze

Bayessche Netze [Jen01, Bay63, Joy03] sind Grundlage einer Methodik zum Schließen unter unsicheren Umständen. Zunächst soll die allgemeinere Form, das Kausale Netz betrachtet werden.

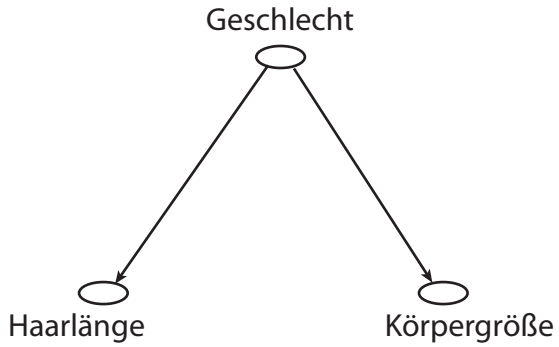


Abbildung 7.4: Beispiel für ein Kausales Netz.

Kausale Netze

Kausale Netze [Jen01] sind gerichtete Graphen mit Variablen als Knoten. Diese Variablen stehen für verschiedene Ereignisse. Eine Variable kann beispielsweise die Farbe eines Autos (Zustände: rot, gelb, blau, silber), die Zahl der Kinder in einer Familie (Zustände: 1, 2, 3, 4, 5, 6, >6) oder eine Krankheit (Zustände Bronchitis, Tuberkulose, Lungenkrebs) repräsentieren. Dabei können die Werte der Variablen wie im Beispiel diskret (abzählbar) oder kontinuierlich sein. Eine Variable kann zu einem Zeitpunkt nur genau einen Wert (einen „Zustand“) annehmen. Welcher das ist, kann unbekannt sein.

Die gerichteten Kanten repräsentieren Abhängigkeiten zwischen den Variablen. Die Wahrscheinlichkeit, mit der eine Variable einen bestimmten Zustand annimmt, beeinflusst die Wahrscheinlichkeit für eine anderen Variable.

Abbildung 7.4 zeigt zum Beispiel, dass die Haarlänge einer Person abhängig ist von ihrem Geschlecht, ebenso die Körpergröße. Dafür ist in diesem Beispiel die Haarlänge unabhängig von der Körpergröße.

Bislang ist das kausale Netz nur in der Lage, diese Abhängigkeit qualitativ darzustellen. Eine Möglichkeit, auch quantitative Aussagen zu erlauben, eröffnen der Satz von Bayes und die auf ihm basierenden Bayesschen Netze.

Satz von Bayes

Voraussetzung dabei sind n paarweise disjunkte Ereignisse $A_1, A_2, A_3, \ldots, A_n$, die den Ereignisraum Ω aufspannen. Dabei gilt nach dem Multiplikationssatz für ein beliebiges Ereignis B:

$$\mathrm{P}(B \cap A_i) = \mathrm{P}(A_i) \cdot \mathrm{P}(B|A_i) = \mathrm{P}(B) \cdot \mathrm{P}(A_i|B)$$

Dies lässt sich umformen zu:

$$\mathrm{P}(A_i|B) = \frac{\mathrm{P}(A_i) \cdot \mathrm{P}(B|A_i)}{\mathrm{P}(B)}$$

Nach dem Theorem der totalen Wahrscheinlichkeit gilt:

$$\mathrm{P}(B) = \sum_{j=1}^{n} \mathrm{P}(A_j) \cdot \mathrm{P}(B|A_j)$$

Eingesetzt ergibt das den *Satz von Bayes (Bayessche Regel)*:

$$\mathrm{P}(A_i|B) = \frac{\mathrm{P}(A_i)\cdot\mathrm{P}(B|A_i)}{\mathrm{P}(A_1)\cdot\mathrm{P}(B|A_1)+\mathrm{P}(A_2)\cdot\mathrm{P}(B|A_2)+\ldots+\mathrm{P}(A_n)\cdot\mathrm{P}(B|A_n)}$$

Auch wenn die mathematische Leistung hinter diesem Satz recht trivial ist, so ist die Bedeutung für die Berechnung bedingter Wahrscheinlichkeiten enorm. Man kann die Ereignisse A_i als Ursachen eines beobachteten Ereignis B auffassen. Der Satz von Bayes ermöglicht nun, im Fall einer beobachteten Wirkung B die Wahrscheinlichkeit dafür auszurechnen, dass eine der Ursachen A_i zu Grunde liegt.

Ein einfaches Beispiel dazu: Sei $\mathrm{P}(M) = 0,5$ die Wahrscheinlichkeit dafür, dass eine Person männlich ist und entsprechend $\mathrm{P}(W) = 0,5$ die Wahrscheinlichkeit, dass eine Person weiblich ist. Sei zudem $\mathrm{P}(L|M) = 0,1)$ die Wahrscheinlichkeit dafür, dass eine Person lange Haare hat unter der Bedingung, dass die Person männlich ist, und $\mathrm{P}(L|M) = 0,7$ die Wahrscheinlichkeit dafür, dass eine weibliche Person lange Haare hat. Weiß man nun von einer Person, dass sie lange Haare hat, ohne ihr Geschlecht zu kennen, so lässt sich nun mit dem Satz von Bayes die Wahrscheinlichkeit dafür ausrechnen, dass sie weiblich ist:

$$\mathrm{P}(W|L) = \frac{\mathrm{P}(W)\cdot\mathrm{P}(L|W)}{\mathrm{P}(M)\cdot\mathrm{P}(L|M)+\mathrm{P}(W)\cdot\mathrm{P}(L|W)} = \frac{0.5\cdot0.7}{0.5\cdot0.1+0.5\cdot0.7} = 0.875$$

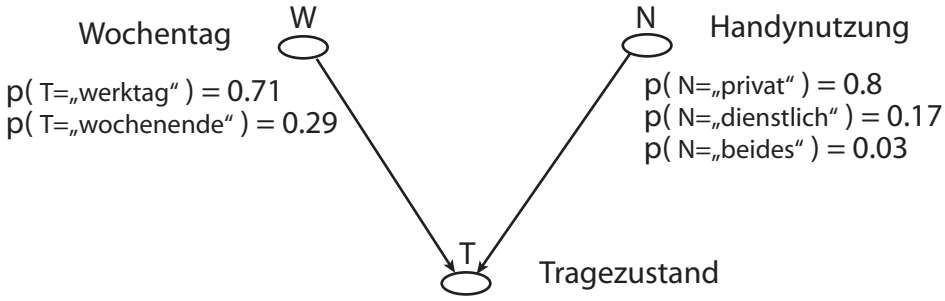
Angewandt wird diese Methodik unter anderem in Bayesschen Netzen, die aus einem Graphen und bedingten Wahrscheinlichkeitsverteilungen bestehen.

Definition (Bayessches Netz)
Ein Bayessches Netz ist ein azyklischer, gerichteter Graph, dessen Knoten Zufallsvariablen repräsentieren, und dessen Kanten die direkten kausalen Abhängigkeiten zwischen diesen Zufallsvariablen beschreiben. Die Eltern $\mathrm{parents(k)}$ *eines Knoten* k *sind diejenigen Knoten, von denen eine Kante zu* k *führt. Für jeden Knoten* k *existiert eine bedingte Wahrscheinlichkeitsverteilung* $\mathrm{P(k|parents(k))}$ *in Abhängigkeit der jeweiligen Eltern.*

Diese Wahrscheinlichkeitsverteilung ist eine mehrdimensionale Tabelle, sofern die Zufallsvariablen nur diskrete Werte annehmen. Für Variablen mit kontinuierlicher Zustandsmenge ist dagegen eine Dichteverteilung notwendig. Ist $k_1, k_2, \ldots, k_n$ eine Teilmenge der Knotenmenge des Graphen, so berechnet sich deren gemeinsame Verteilung wie folgt:

$$\mathrm{P}(k_1, \ldots, k_n) = \prod_{i=1}^{n} \mathrm{P}(k_i|\mathrm{parents}(k_i))$$

Aus dieser Formel wird auch der Vorteil von Bayesschen Netzen zur Verringerung des Berechnungsaufwandes deutlich. Statt alle Möglichkeiten für die abhängigen Belegungen



	p(T=„Benutzer trägt Handy“)	p(T=„Handy ist weggelegt“)
T=„werktag“ ∧ N=„priv.“	0.6	0.4
T=„werktag“ ∧ N=„dienstl.“	0.95	0.05
T=„werktag“ ∧ N=„beides“	0.99	0.01
T=„we.“ ∧ N=„priv.“	0.8	0.2
T=„we.“ ∧ N=„dienstl.“	0.01	0.99
T=„we.“ ∧ N=„beides“	0.95	0.05

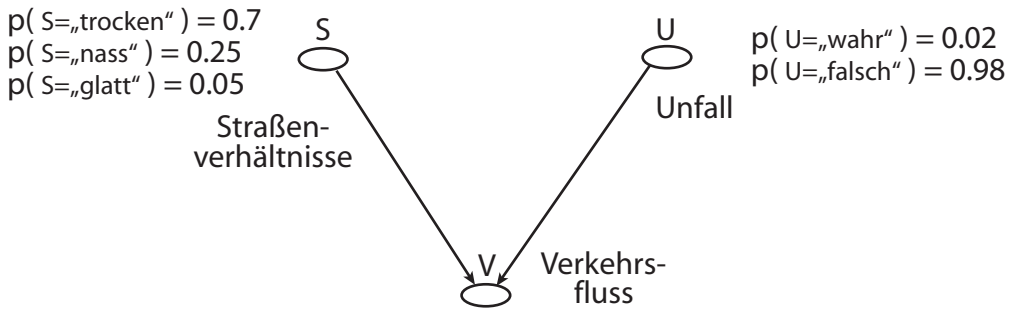
Abbildung 7.5: Ein einfaches Beispiel für ein Bayessches Netz.

der Zufallsvariablen durchzurechnen, gibt diese Formel eine wesentlich effizientere Berechnungsmethode an – gestützt darauf, dass alle Abhängigkeiten bereits explizit modelliert sind. Dabei soll allerdings nicht verschwiegen werden, dass auch diese Berechnung – obwohl effizienter – im schlimmsten Fall immer noch NP-hart sein kann.

Im Beispiel von Abbildung 7.5 geht es darum, ob ein Benutzer sein Handy bei sich trägt oder nicht, in Abhängigkeit davon, ob er es privat oder dienstlich nutzt, und in Abhängigkeit vom Wochentag. Hat man keine Kenntnis vom Zustand der Variablen, so lässt sich mit der eben aufgeführten Formel zur Verteilung die Wahrscheinlichkeit dafür ausrechnen, dass der aktuelle Tag ein Wochentag ist und der Benutzer sein Handy bei sich trägt, das er dienstlich nutzt:

$$\begin{aligned}&\mathrm{P}(W = \text{„werktag“}, N = \text{„dienstlich“}, T = \text{„Benutzer trägt Handy“}) = \\ &= \mathrm{P}(T = \text{„Benutzer trägt Handy“} \mid W = \text{„werktag“}, N = \text{„dienstlich“}) \cdot \\ &\quad \cdot \mathrm{P}(W = \text{„werktag“}) \cdot \mathrm{P}(N = \text{„dienstlich“}) = \\ &= 0.95 \cdot 0.71 \cdot 0.17\end{aligned}$$

Ein weiteres, einfaches Beispiel für ein Bayessches Netz zeigt Abbildung 7.6. Hier hängt die Wahrscheinlichkeit, ob der Verkehr in einem Streckenabschnitt normal fließt, stockend fließt oder sich staut, ab von den Straßenverhältnissen und der Tatsache, ob es eine Unfallstelle gibt. Die Zufallsvariablen S, U, V nehmen nur diskrete Werte an, weshalb die dazu gehörenden bedingten Wahrscheinlichkeitsverteilungen direkt angegeben werden. Dass in der realen Welt auch die Wahrscheinlichkeit einer Unfallstelle von den Straßenverhältnissen abhängt, fehlt in diesem Modell. Da es keine Kante zwischen S und U gibt, sind die Ereignisse als unabhängig modelliert.



	p(V=„normal“)	p(V=„stockend“)	p(V=„stau“)
S=„trocken“ ∧ U=„wahr“	0.1	0.2	0.7
S=„trocken“ ∧ U=„falsch“	0.8	0.15	0.05
S=„nass“ ∧ U=„wahr“	0.05	0.1	0.85
S=„nass“ ∧ U=„falsch“	0.7	0.25	0.05
S=„glatt“ ∧ U=„wahr“	0.01	0.49	0.5
S=„glatt“ ∧ U=„falsch“	0.1	0.7	0.2

Abbildung 7.6: Ein weiteres Beispiel für ein Bayessches Netz.

Die angegebenen a-priori-Wahrscheinlichkeiten können durch statistische Beobachtungen oder Experten festgelegt werden. Sobald der Wert einer Zufallsvariable tatsächlich beobachtet werden kann, ändern sich die Wahrscheinlichkeiten für die Werte der davon abhängenden Variablen entsprechend. Sie lassen sich induktiv und deduktiv berechnen. Exakte Inferenz in einem Bayesschen Netz ist ein NP-hartes Komplexitätsproblem. Zur Anwendung kommen meist Schätzverfahren, die in der Regel ausreichend genaue Ergebnisse liefern. Dies soll an dieser Stelle jedoch nicht vertieft werden.

Festzuhalten bleibt, dass vor allem bei kleineren Netzen kein Aufwandsproblem bei der Berechnung entsteht. Das war auch Motivation für den Ansatz zum Einsatz von Bayesschen Netzen in Kontextwissensbasen: Hier sollen möglichst kleine Vorlagen schnell instanziiert werden können, um der Dynamik einer sich ständig ändernden Wissensbasis und der Optimierung der Antwortzeit bei einer Kontextanfrage gerecht zu werden.

Anwendung von Bayesschen Netzen auf Kontextwissensbasen

Zhongli Ding und Yun Peng haben 2004 in [DP04] eine OWL-Erweiterung vorgeschlagen, mit der Wahrscheinlichkeiten modelliert werden können. Dafür gibt es drei neue Klassen „PriorProbObj“, „CondProbObjT“ und „CondProbObjF“.

Wenn A und B Klassen sind, dann interpretieren sie $\mathrm{P}(A)$ als Wahrscheinlichkeit, dass ein beliebiges Individuum zur Klasse A gehört und $\mathrm{P}(A|B)$ als Wahrscheinlichkeit, dass ein Individuum der Klasse B ebenfalls zur Klasse A gehört. Mit den drei neuen OWL-Klassen sollen jeweils die drei Aussageformen $\mathrm{P}(A)$, $\mathrm{P}(A|B)$ und $\mathrm{P}(A|\overline{B})$ entsprechend ausgedrückt werden. Im Folgenden soll das OWL-Beispiel für $\mathrm{P}(\text{Männlich}|\overline{\text{Tier}}) = 0.5$

(die Wahrscheinlichkeit, dass ein Individuum, das kein Tier ist, männlich ist) beschrieben werden:

```
<prob:CondProbObjF rdf:ID="P(Männlich|(not)Tier)">
 <prob:hasCondition><rdf:Value>&ont;Männlich</rdf:Value></
    prob:hasCondition>
 <prob:hasVariable>
  <rdf:Value>&ont;Tier</rdf:Value>
 </prob:hasVariable>
 <prob:hasProbValue>0.32</prob:hasProbValue>
</prob:CondProbObjF>
```

Listing 7.1: Bedingte Wahrscheinlichkeit nach [DP04].

Aussagen der Form $P(A|B_1,\ldots,B_n)$ lassen sich damit noch nicht direkt modellieren. Die Autoren versprechen entsprechende Methoden aber für die Zukunft. Abgesehen von diesen OWL-Erweiterungen liefern Ding und Peng einen Algorithmus, wie aus einer Ontologiedatei, die diese Erweiterungen enthält, ein Bayessches Netz generiert werden kann, indem sie Transformationsregeln für die jeweiligen Strukturelemente und Konstruktionsmöglichkeiten für die Wahrscheinlichkeitstabellen definieren. Wie die Wahrscheinlichkeitsaussagen auf eine Kontextwissensbasis, also auf Instanzen der Ontologiedateien angewandt werden können, wird nicht beschrieben.

Tao Gu und andere haben in [GPZ04] einen eigenen Ansatz zum Einsatz von Bayesschen Netzen für Kontextwissen vorgestellt. Sie haben sich dabei nach eigener Aussage von der Arbeit von Ding und Peng inspirieren lassen, wollen aber einen allgemeineren Ansatz liefern. Deshalb definieren sie nur zwei neue OWL-Klassen „PriorProb“ und „CondProb“. Eine a-priori-Wahrscheinlichkeit der Form $P(A) = 0.7$ würde damit wie folgt modelliert werden:

```
<prob:PriorProb rdf:ID="P(A)">
 <prob:hasVariable><rdf:value>A</rdf:value></prob:hasVariable>
 <prob:hasValue>0.7</prob:hasValue>
</prob:PriorProb>
```

Listing 7.2: A-priori-Wahrscheinlichkeit nach [GPZ04].

Mit der Klasse „CondProb“ können bedingte Wahrscheinlichkeiten modelliert werden, wie das Beispiel $P(A|B,\overline{C}) = 0.5$ zeigt. Ausgedrückt in OWL würde dies dann wie in dem folgenden Listing aussehen:

```
<prob:CondProb rdf:ID="P(A|B,notC)">
 <prob:hasCond><rdf:value>B</rdf:value></prob:hasCond>
 <prob:hasCond><rdf:value>notC</rdf:value></prob:hasCond>
 <prob:hasVariable><rdf:value>A</rdf:value></prob:hasVariable>
 <prob:hasProbValue>0.5</prob:hasProbValue>
</prob:CondProb>
```

Listing 7.3: Bedingte Wahrscheinlichkeit nach [GPZ04].

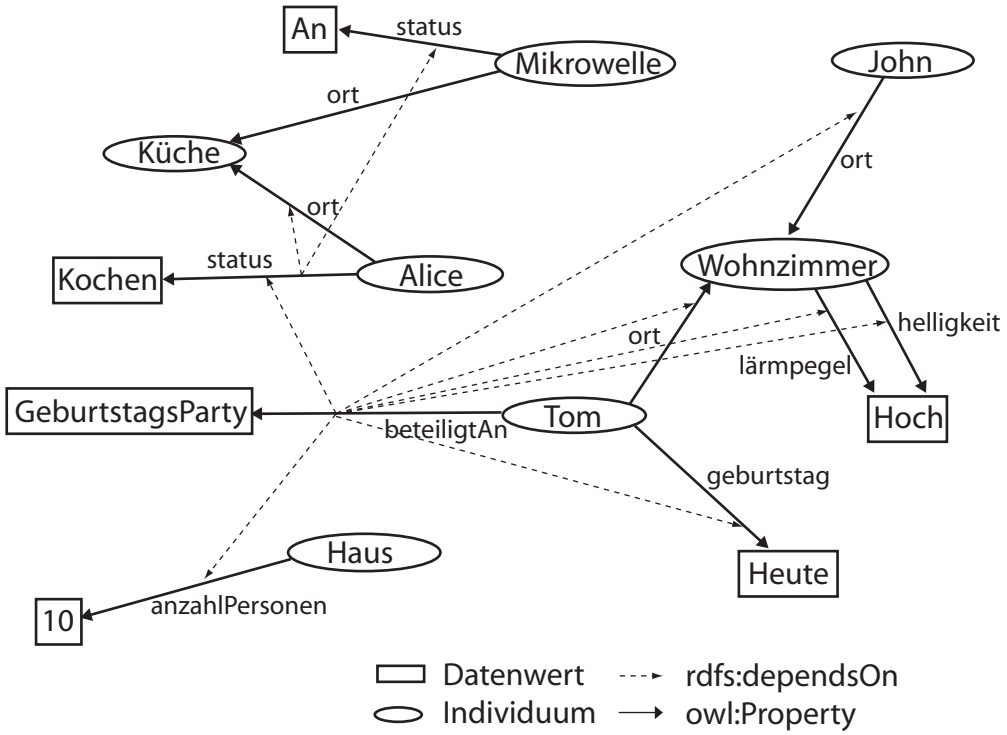


Abbildung 7.7: Teil einer Kontextwissensbasis nach [GPZ04].

Wie zu erkennen ist, erlaubt diese Modellierung im Bedingungsteil mehrere, auch negierte Variablen und ist damit mächtiger als der Ansatz von Ding und Peng. Durch einen Bedingungsterm mit dem Wert „notC“ wird allerdings eine Syntax innerhalb der XML-Syntax verschachtelt, die nicht mehr mit XML-Mitteln zu lesen ist, sondern einen zusätzlichen Parser braucht. Gu et al. richten ihr Augenmerk nicht auf Ontologien, wie Ding und Peng, sondern konstruieren Bayessche Netze auf Kontextwissensbasen.

Dazu benutzen sie die Assoziation „rdfs:dependsOn“, um in OWL modellierte Kontextaussagen untereinander in Beziehung zu setzen. Jede Kontextaussage („owl:Property“) wird dann zu einem Knoten in einem Bayesschen Netz überführt, wobei zwei Knoten dann und nur dann als abhängig verbunden werden, wenn es die entsprechende „dependsOn“-Beziehung in der Kontextwissensbasis gibt (vgl. Abbildung 7.7). Dies sei möglich, weil eine OWL-Kontextwissensbasis (von Gu et al. verwirrender Weise als *context ontology* bezeichnet) in ihrer RDF-Struktur ebenfalls ein gerichteter, azyklischer Graph ist. Diese Aussage ist jedoch nicht nachvollziehbar, da Kontextwissensbasen sehr wohl Zyklen enthalten können – im einfachsten Fall bereits durch symmetrische Kontextinformationsklassen (wie „befindetSichIn“ und „enthält“). Abbildung 7.8 zeigt,wie aus abhängigen Kontextaussagen auf die Aktivität von Tom geschlossen werden kann. Zur Konstruktion der bedingten Wahrscheinlichkeitsverteilungen sagen Gu et al. nichts aus, sie gehen

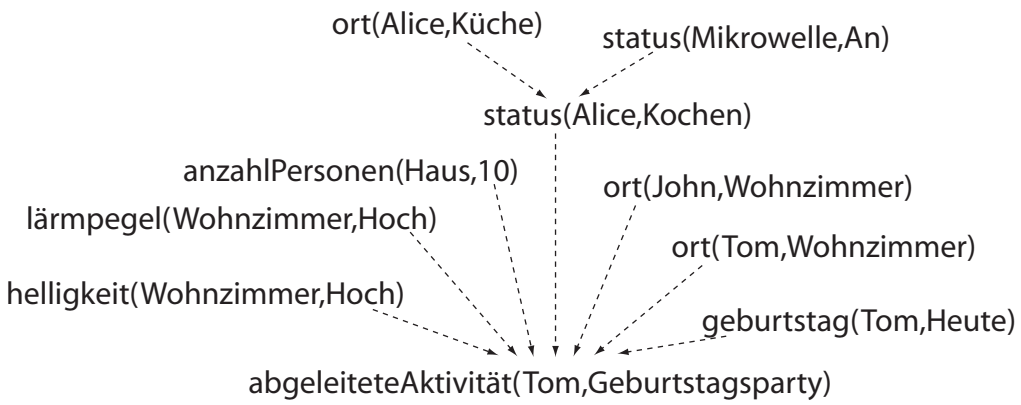


Abbildung 7.8: Schließen aus abhängigen Kontextinformationen nach [GPZ04].

davon aus, dass die Wahrscheinlichkeiten nach Bildung der Netzstruktur „trainiert“ werden. Damit geben weder Ding und Peng noch Gu et al. eine praktikable Vorgehensweise an, probabilistische bedingte Aussagen auf Ontologieebene in einer sich stetig ändernden Kontextwissensbasis anzuwenden.

Fuzzy SWRL

Fuzzy SWRL ist eine Entwicklung der *Fuzzy RuleML Technical Group*. Diese Gruppe ist Teil der *RuleML Community* und der *Semantic Web Community* und will das *RuleML framework* [HBG+05] um Fuzzy-Mengen und Fuzzy-Logik zu Fuzzy RuleML erweitern. Damit sollen sowohl sichere als auch unsichere Informationen dargestellt werden. Dafür untersucht die Gruppe, welche syntaktischen und semantischen Veränderungen notwendig sind, um mit RuleML auch *fuzzy* Informationen repräsentieren zu können. Eines der Ergebnisse ist eine Erweiterung von SWRL (siehe 4.3.3) zu f-SWRL [PSTH05, SPTH05]. Dabei können OWL-Axiome eine Angabe des Grades („*degree*“) der Sicherheit (Wahrheitswert zwischen 0 und 1) darüber enthalten, dass ein Individuum Instanz einer bestimmten Klasse ist. Außerdem können SWRL-Atome eine Gewicht („*weight*“) enthalten, das die relative Bedeutung innerhalb einer Regel repräsentiert. Die folgenden Fuzzy-Regeln besagen, dass jemand, der einen glücklichen Elternteil beziehungsweise einen glücklichen Bruder hat, selbst wahrscheinlich (mit einem Gewicht von 0.8 bzw. 0.4) glücklich ist.

$$\textit{Elternteil(?x, ?p)} \wedge \textit{IstGlücklich(?p)} \rightarrow \textit{IstGlücklich(?x)*0.8}$$

$$\textit{Bruder(?x, ?b)} \wedge \textit{IstGlücklich(?b)} \rightarrow \textit{IstGlücklich(?x)*0.4}$$

Dabei wird der Regel, die sich auf den Elternteil bezieht, mehr Gewicht eingeräumt. Anders das folgende Beispiel, in dem eine Regel besagt, dass ein Individuum glücklich ist, wenn es die Augenbrauen hochgezogen und den Mund geöffnet hat.

$$\textit{AugenbrauenHochgezogen(?a)*0.9} \wedge \textit{MundOffen(?a)*0.8} \rightarrow \textit{IstGlücklich(?a)}$$

Hierbei sind *AugenbrauenHochgezogen*, *MundOffen* und *IstGlücklich* Klassen, ?*a* eine Variable für ein Individuum und 0.9 sowie 0.8 die Gewichte der Regelatome *AugenbrauchenHochgezogen(?a)* und *MundOffen(?a)*.

Fuzzy Regeln mit Beschreibungslogiken Sudhir Agarwal und Pascal Hitzler haben ähnliche Regel-Erweiterungen entwickelt, die sie in [AH05] vorstellen. Sie zeigen formal, wie eine Fuzzy „If-Then"-Regel mit den Mitteln einer geeigneten Beschreibungslogik modelliert werden kann und wie darauf Anfragen entschieden werden können. Dazu gehört auch eine auf Diskretisierung der Mitgliedsgradfunktion beruhende Methode, den Mitgliedsgrad eines Individuums zu unscharfen Mengen zu berechnen. Agarwal und Hitzler führen ein eigenes Konzept „Regel" ein, das wie folgt definiert ist:

$$\textit{Rule} \sqsubseteq \exists\, \textit{antecedent.TermExp} \sqcup \exists\, \textit{consequent.Term} \sqcup \exists\, \textit{degree.}\mathbb{R}_{[0,1]}$$

Dabei sind „antecedent" und „consequent" jeweils funktionale Rollen, wobei der Antezedensteil aus mehreren Termen und der Konsequenzteil aus genau einem Term besteht. Während der Grad der Erfüllung eines einzelnen Terms von dem Wert der dazu gehörenden Mitgliedsgradfunktion abhängt, hängt der Grad der Erfüllung der gesamten Regel von der Kombination dieser Werte ab. Dabei definieren Agarwal und Hitzler die Konjunktion, die Disjunktion und die Negation wie folgt:

$$\begin{aligned}
\textit{TermExp}_{\wedge} &\sqsubseteq P_{=}(\textit{degree}, \min\{\textit{conjunct} \circ \textit{degree}\}) \\
\textit{TermExp}_{\vee} &\sqsubseteq P_{=}(\textit{degree}, \max\{\textit{disjunct} \circ \textit{degree}\}) \\
\textit{TermExp}_{\neg} &\sqsubseteq P_{=1-}(\textit{degree}, \min\{\textit{operand} \circ \textit{degree}\})
\end{aligned}$$

Mit welchen Funktionen Disjunktion, Konjunktion und Negation interpretiert werden, ist davon unabhängig. Um eine Fuzzy „If-Then"-Regel interpretieren zu können, bedarf es noch der Interpretation der Implikation. Ohne sich auf eine Interpretationsart festzulegen, definieren Agarwal und Hitzler

$$\textit{Rule} \sqsubseteq P_{\pi}(\textit{degree, antecedent} \circ \textit{degree, consequent} \circ \textit{degree})$$

Dabei ist P_{π} ein ternäres Prädikat, das die Interpretation der Implikationsfunktion π repräsentiert. Das bedeutet, für gegebene $a, b, c \in \mathbb{R}$ wird $P_{\pi}(a,b,c)$ wahr dann und nur dann, wenn $a = \pi(b,c)$.

Die Arbeit von Agarwal und Hitzler überschneidet sich in weiten Teilen mit den Überlegungen zu f-SWRL. Wo dort mehr Gewicht auf die Umsetzung in der Regelsprache gelegt wird, geht es hier überwiegend um die Methodik und die Berechnung – insofern ergänzen sich die Arbeiten auch.

Es bleibt festzuhalten, dass es große Anstrengungen gibt, Regelsprachen der Beschreibungslogiken dahingehend zu erweitern, dass sie nicht nur mit binären Werten, sondern

auch mit dem Werteintervall $[0, 1]$ umgehen können. Obwohl wie bereits erwähnt Fuzzy-Logik und Probabilistische Aussagen von gänzlich unterschiedlicher Semantik sind, eignen sich die Modellierungstechniken wegen der gleichen Syntax für beide. Es ist deshalb zu erwarten, dass die Entwicklung der vorgestellten f-SWRL für probabilistische Aussagen ganz genauso verwandt werden kann.

7.2 Vorstellung der Strategien

In diesem Abschnitt werden nun Strategien vorgestellt, die dazu dienen, Kontextinformationen bereit zu stellen, die nicht direkt von einem Kontextinformationsdienst beschafft werden können.

7.2.1 Kriterien

Welche davon die geeignetste ist, ist abhängig vom jeweiligen Einzelfall. Deshalb werden die Strategien anhand ihrer Auswahlkriterien diskutiert, denen hier die Überbegriffe „Geltungsbereich“, „Wirkungsbereich“, „Kosten“, „Erfolgschancen“ und „Prognosechancen“ gegeben werden. Im Folgenden werden zunächst diese Kriterien erläutert.

Geltungsbereich Nicht jede Möglichkeit, eine Kontextinformation auf einem Ersatzweg oder eine Ersatzinformation bereit zu stellen, ist allgemein gültig. Eine Strategie mag zum Beispiel nur für eine bestimmte Art von Anwendung gelten, oder gar nur für eine bestimmte Anwendung. Ebenso mag sie nur für eine Benutzergruppe oder gar nur einen Benutzer, eventuell sogar nur innerhalb einer bestimmten Anwendung gelten. Dieses Kriterium bezieht sich also auf den Abnehmer der Kontextinformation (vgl. Kapitel 3).

Wirkungsbereich Eine Strategie kann allgemein anwendbar sein auf eine bestimmte Kontextinformationsklasse, gegebenenfalls lediglich noch eingeschränkt durch gewisse Bedingungen, die erfüllt sein müssen. Der Wirkungsbereich einer Strategie kann aber auch eingeengt sein auf bestimmte Entitätsklassen. Eine Strategie kann aber genauso gut nur für bestimmte Instanzen der Entität oder der Kontextinformationen anwendbar sein, und selbst hier noch weiteren Einschränkungen – zum Beispiel zeitlich oder kausal – unterliegen.

Kosten Eine Kontextinformation auf einem Ersatzweg zu beschaffen, beziehungsweise eine Ersatzinformation zu beschaffen, kostet Zeit, Rechenaufwand und eventuell auch direkt Geld (auch wenn *Accounting* im Rahmen dieser Arbeit nicht betrachtet wird). Diese Kosten sind gegenüber ihrem Nutzen abzuwiegen.

Erfolgschancen Ebenso, wie das direkte Beschaffen einer Kontextinformation mit einem recht großen Risiko des Scheiterns behaftet ist, gibt es auch für die Ausweichstrategien keine Garantie, dass sie erfolgreich sind. Erfolgreich bedeutet in diesem Fall,

dass sie eine Information liefern, die den Anforderungen des Abnehmers genügt, und dass sie dies innerhalb des vom Abnehmer gesteckten Zeit- und Kostenrahmens tun. Allerdings haben die unterschiedlichen Strategien unterschiedliche Erfolgschancen.

Prognosechancen Um die Erfolgschancen in eine Kosten-Nutzen-Betrachtung bei der Planung der Strategien eingehen zu lassen, muss man diese Chancen kennen. Diese Vorhersagbarkeit ist bei den Strategien ebenfalls sehr unterschiedlich ausgeprägt.

7.2.2 Direkte Äquivalenzen

Ist eine Kontextinformation auf direktem Weg von einem Kontextinformationsdienst nicht zu beschaffen, so besteht die einfachste Möglichkeit darin, als Ersatz eine äquivalente Kontextinformation zu besorgen. Notwendig sind Vorschriften, welche Kontextinformationen äquivalent sind. So ließen sich zum Beispiel Äquivalenzen festlegen zwischen einer Entitätsklasse „Mensch“ in einer Ontologie aus dem Fachbereich der Medizin und der Klasse „Fahrgast“ aus einer Ontologie für die Personenbeförderung. Ebenso können die Kontextinformationsklassen „istAmOrt“ und „hatPosition“ aus verschiedenen Ontologien als äquivalent gesetzt werden. Listing 7.4 zeigt zum Beispiel, wie eine eine in CMPlus formulierte Beispielontologie „Example1_Ontology“ die Kontextinformationsklasse „Name“ einer anderen Beispielontologie „Example2_Ontology“ äquivalent setzt zur eigenen Kontextinformationsklasse „Name“.

```
<rdf:RDF
xmlns="http://www.mobile.ifi.lmu.de/~krausem/owl/Example1_Ontology.owl#
    "
.../>

<owl:Class rdf:ID="Name">
 <cmp:equivalentContextInformation rdf:resource="http://www.mobile.ifi.
    lmu.de/~krausem/owl/Example2_Ontology.owl#Name"/>
 ...
</owl:Class>
```

Listing 7.4: Beispiel für eine Äquivalenz zwischen Kontextinformationsklassen

Die Gültigkeit von Äquivalenzen lässt sich mit Bedingungen versehen, etwa der Art: Nach 21 Uhr abends ist der Aufenthaltsort eines Benutzers gleich dem Wohnort des Benutzers, wenn er keine Termine in seinem Kalender eingetragen hat. Oder: Die Außentemperatur an einem Ort ist äquivalent der Außentemperatur an einem anderen Ort, sofern dieser nicht weiter als einen Kilometer entfernt ist. Das ließe sich noch insofern präzisieren, dass die Ungenauigkeit der Messung eine größere wird, je weiter der Messort vom Zielort entfernt ist. Das entspräche dann weniger einer Äquivalenz als einer Schätzung. Für alle diese Aussagen ist ein gewisses Maß an Logik erforderlich und insofern werden sie mit Regeln formuliert, die in Abschnitt 7.2.4 diskutiert werden.

Geltungsbereich Im allgemeinsten Fall sind Äquivalenzen in den jeweiligen Ontologien hinterlegt. CMPlus hält dafür etwa die Assoziation „equivalentContextInformation“ (eine Spezialisierung von „owl:equivalentClass“) vor, mit der es möglich ist, Kontextinformationsklassen als äquivalent zu kennzeichnen. In Listing 7.4 werden die beiden Kontextinformationsklassen „Name“ (zufällig mit dem gleichen Klassennamen) aus den Ontologien „Example1_Ontology“ und „Example2_Ontologie“ als äquivalent definiert. Da es nicht sinnvoll ist, innerhalb einer Ontologie äquivalente Kontextinformationsklassen mit verschiedenen Namen vorrätig zu halten (der einzige Anwendungsfall für diese Beziehung innerhalb derselben Ontologie), werden Äquivalenzbeziehungen zwischen verschiedenen Ontologien gleicher Modellierungsart eingesetzt. Aufgrund der Allgemeinheit kann der Vermittler nach Ersatzkontextinformationen suchen, und zwar ohne weitere Informationen oder Vorgaben vom Abnehmer. Diese Äquivalenzen gelten auch immer unabhängig von der Entität der Kontextinformation.

Äquivalenzen lassen sich auch in spezielleren Fällen formulieren. Etwa wenn zwei Begriffe (seien es Entitäten oder Kontextinformationsklassen) zwar eine ähnliche, aber eben doch grundsätzlich unterschiedliche Semantik tragen, aber für einen bestimmten Anwendungsfall trotzdem als gleichwertig zu verwenden sind. Dies kann in den verschiedensten Geltungsbereichen definiert werden. Etwa innerhalb einer bestimmten Anwendung, einer bestimmten Anwendungssitzung, für einen bestimmten Benutzer oder gar nur für eine einzige Kontextanfrage. In diesem Fall müssen jedoch die jeweiligen Äquivalenzen dem Kontextvermittler mitgeteilt werden, damit dieser seine Suche nach passenden Kontextinformationsdiensten entsprechend ausdehnen kann.

Wirkungsbereich Die Äquivalenzen von Klassen gelten per se nur auf Klassenebene, unabhängig von den Individuen, selbst wenn der Geltungsbereich auf bestimmte Anwendungen oder Benutzer eingeschränkt sein sollte.

Kosten Die Kosten sind sehr gering für allgemeine Äquivalenzen. Hier kann der Suchbereich nach geeigneten Kontextinformationsdiensten entsprechend erweitert werden, was eine Parallelisierung der Suche bedeutet und damit quasi keinen zusätzlicher Zeitaufwand bedingt. Geringer zusätzlicher Rechenaufwand entsteht, um den größeren Suchbereich über die äquivalenten Kontextinformationsklassen aufzuspannen. Geringer zusätzlicher Rechenaufwand entsteht gegebenenfalls auch, falls eine größere Kandidatenmenge daraufhin auszuwerten ist, welche Kontextinformationsdienste geeignete Kontextinformationen liefern könnten.

Erfolgschancen Die Chancen, eine äquivalente Kontextinformation zu beschaffen, sind qualitativ genauso groß wie bei schon bei der Beschaffung der ursprünglichen angefragten. Ist diese Wahrscheinlichkeit für jede Kontextanfrage näherungsweise p, so sinkt die Gefahr, keinen Kontextinformationsdienst zu finden, der die angefragte Kontextinformation bereitstellen kann, damit auf $(1-p)^{(1+k)}$ bei k angegebenen Äquivalenzen.

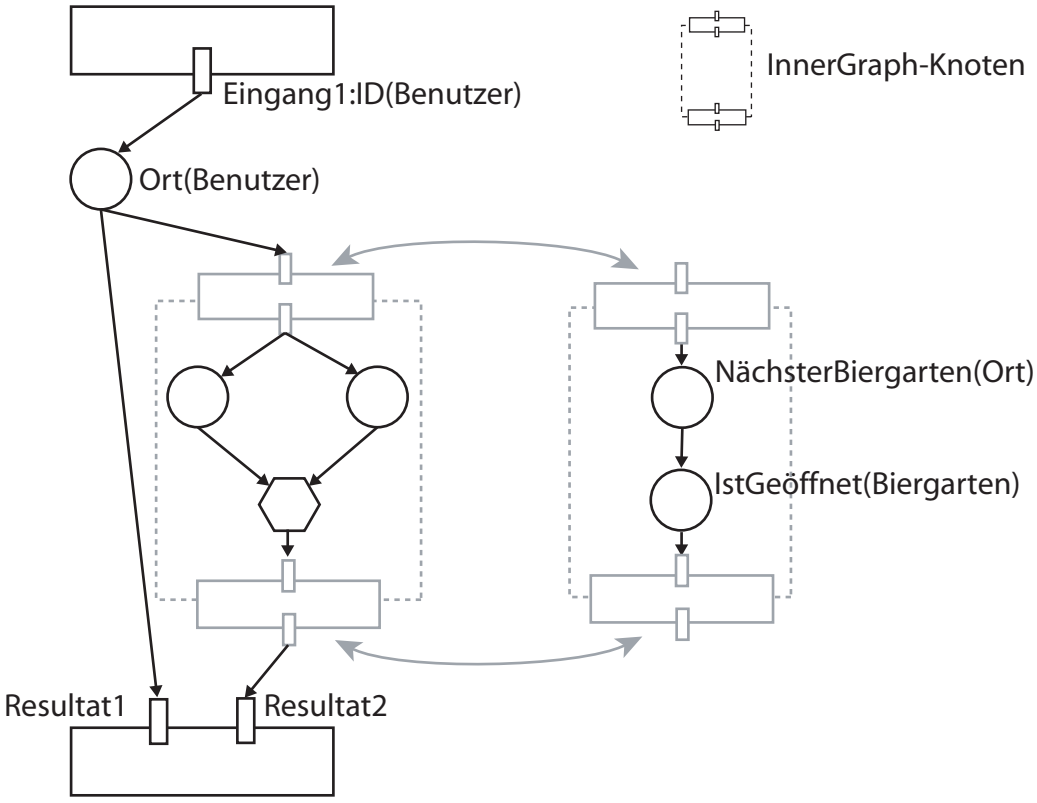


Abbildung 7.9: Äquivalenzen von Kontextkompositionsgraphen nach [HK04].

Prognosechancen Ob die Beschaffung einer äquivalenten Kontextinformation erfolgreich sein wird, lässt sich genauso wenig oder genauso gut prognostizieren, wie bei jedem anderen Versuch, eine Kontextinformation zu beschaffen. Prognosen lassen sich nur treffen, wenn der Kontextvermittler bereits ähnliche oder gleiche Kontextanfragen gestellt hat. Deren Erfolg oder Misserfolg ließen dann positive oder negative Prognosen zu.

7.2.3 Äquivalenzen mit Kompositionsgraphen

Eine Kontextanfrage mit einem Kontextkompositionsgraphen ermöglicht eine weiter gehende Definition von Äquivalenzen als die Äquivalenz zwischen zwei Kontextinformationsklassen, nämlich die Äquivalenz von Kompositionsgraphen und deren Teilen [HK04].

Die Abbildung 7.9 greift dabei das Beispiel von Abbildung 7.1 wieder auf. Darin ist ein Kontextkompositionsgraph beschrieben, wie ihn etwa ein Restaurant-Finder-Dienst benutzen könnte, um zwei Kontextinformationen anzufordern: Zum einen den Ort des Benutzers, zum anderen die Information, ob schönes Wetter ist (was erlauben würde, zum Beispiel Biergärten oder Straßencafés vorzuschlagen). Letztere wird durch eine Betrachtung der Temperatur und des Regenrisikos am aktuellen Ort errechnet. Ist jetzt eine der

beiden Informationen nicht verfügbar, so können die drei Knoten gegen einen anderen Teilgraphen ausgetauscht werden, der ersatzweise versucht, herauszufinden, ob der nächst gelegene Biergarten geöffnet ist. Diese – anwendungsspezifische – Äquivalenz von Teilgraphen lässt sich spezifizieren mit so genannten InnerGraph-Knoten der Kontextkompositionsgraphen. Diese enthalten einen zusammenhängenden Teilgraphen, der für sich genommen selbst ein kompletter Kontextkompositionsgraph ist. Durch dieses Abteilen werden auch gleichzeitig Zielbereiche für Steuerungsanweisungen definiert. Zum Beispiel um Kosten- oder Zeitgrenzen zu definieren, die beim Abarbeiten aller Knoten des Bereichs nicht überschritten werden dürfen.

Für einen Teilgraphen können selbstverständlich mehrere Äquivalenzen definiert werden. Es gibt schließlich keine Garantie, dass bereits alle Knoten des ersten eingetauschten Teilgraphen erfolgreich abgearbeitet werden können.

Geltungsbereich Mit dieser Strategie lassen sich sowohl allgemeine als auch spezifische Äquivalenzen (auf Anwendungs- oder Benutzerebene) definieren. Dabei unterscheidet sich allerdings das Vorgehen:

Allgemein gültige Äquivalenzen zwischen Kontextkompositionsgraphen (auch Teilgraphen sind valide Kontextkompositionsgraphen) lassen sich allgemein gültig definieren. Dies ist dann sinnvoll, wenn die Kontextkompositionsgraphen häufige und allgemeine Kontextkompositionen beschreiben. In [HK04] ist beschrieben, wie solche Standard-Kontextkompositionsgraphen in einem Speicher vorgehalten werden, so dass eine derartige Kontextanfrage nur noch den Verweis auf den Standardgraphen und die Eingabewerte für die Eingangsknoten enthalten braucht. In dem Beispiel der Abbildungen 7.1 und 7.9 ist der Eingangswert die ID des Benutzers des Dienstes. In diesen Speichern von Standard-Kontextkompositionsgraphen werden nun allgemein gültige Äquivalenzen zwischen den Graphen oder zwischen Graphen und Teilgraphen (im Extremfall ein einzelner Knoten) definiert, auf die der Kontextvermittler zugreifen kann. Diese Äquivalenzen haben dann den Ontologie-Rang.

Sollen für eine Kontextanfrage nicht allgemeine (also anwendungsspezifische) Äquivalenzen gelten, so muss der anfragende Dienst diese dem Kontextvermittler eigens mitteilen. Dafür gibt es drei Möglichkeiten: (a) Entweder die Alternativ-Graphen werden direkt in das Dokument des Kontextkompositionsgraphen geschrieben und somit mit der Kontextanfrage versandt. (b) Oder der anfragende Dienst hält selbst einen Speicher mit Äquivalenzgraphen vor, baut in den Kontextkompositionsgraphen lediglich die entsprechenden Verweise auf diesen Speicher hin und erlaubt dem Kontextvermittler für diese Anfrage den Zugriff darauf. Handelt es sich um innerhalb des Dienstes häufig benutzte Äquivalenzen, so kann auch (c) der Kontextvermittler als Proxy einen Teil der dienstspezifischen Äquivalenzgraphen speichern, um sich den Zugriff auf den entfernten Speicher zu sparen.

Wirkungsbereich Äquivalenzen zwischen Teilgraphen sind im Grunde Äquivalenzen auf Klassenebene, da in der Regel erst zur Laufzeit die Entitäten und Kontextinformationen ermittelt werden, die in die Knoten und Teilgraphen eingehen. Für Äquivalenzen zwischen Standard-Kontextkompositionsgraphen gilt dies im Besonderen. Ein wenig anders sieht es bei anwendungsspezifischen Graphäquivalenzen aus. Durch Angaben in den Kontextkompositionsgraphen kann die Allgemeinheit sehr eingeschränkt sein, so dass sie nur noch für bestimmte Instanzen gilt. Insbesondere ist dies der Fall, wenn die Äquivalenz nur innerhalb einer Kontextanfrage definiert ist, die schon durch die Eingabewerte des Graphen die mögliche Instanzenmenge stark einschränkt.

Kosten Kosten an Zeit und Rechenaufwand entstehen beim Austausch eines Graphenteils gegebenenfalls durch die Suche nach äquivalenten Graphen und durch die Abarbeitung des Teilgraphen, also für jede einzelne Kontextanfrage, für jede Verarbeitung in einem Operatorknoten, für alle unter Umständen notwendigen Transformationen der Repräsentationsformate und die Steuerung der Abarbeitung. Die Kosten sind also eine Frage der Gestalt des eingetauschten Graphen. Im Extremfall können die Kosten durch einen Austausch sogar geringer werden als im Erfolgsfall des originalen Kontextkompositionsgraphen.

Erfolgschancen Der restrukturierte Kontextkompositionsgraph ist qualitativ äquivalent zum ursprünglichen Graphen, im Grundsatz ist seine Erfolgschance deshalb genauso gut oder schlecht. Die Erfolgswahrscheinlichkeit erhöht sich wie bei der vorangegangen Strategie durch die Vermehrung der Alternativen.

Prognosechancen Die Erfolgschancen des umstrukturierten Graphen sind ebenso wenig prognostizierbar, wie die für den ursprünglichen Kontextkompositionsgraphen.

7.2.4 Regeln

Alle bislang genannten Strategien beziehen sich auf das Beschaffen von Kontextinformationen von Kontextinformationsdiensten. Das Wissen über eine bestimmte Kontextinformation ergibt sich dabei, indem diese Kontextinformation explizit neu geliefert wird. Anders bei dieser Strategie: Hier wird aus Kontextwissen, das bereits angesammelt worden ist, auf Kontextinformationen geschlossen, die explizit noch nicht vorliegen. Ist zum Beispiel der Ort einer Person nicht bekannt, aber der Ort eines Mobiltelefons, das diese bei sich trägt, so kann geschlossen werden, dass sich die Person am Ort des Mobiltelefons befindet. Ein Beispiel für eine solche Regel ist die folgende:

$$Person(?p) \wedge Mobiltelefon(?m) \wedge Ort(?o) \wedge traegt(?p,?m) \wedge istAmOrt(?m,?o) \rightarrow istAmOrt(?p,?o)$$

Grundlagen dafür sind in den Abschnitten 4.3.3 und 7.1.3 erläutert worden. Wie dort ebenfalls bereits ausgeführt, kann das Ergebnis solcher Regeln auch eine Kontextinformation sein, die nur mit einer gewissen Wahrscheinlichkeit gültig ist. Je nach Anwendungsfall ist dies für kontextsensitive Dienste oft ausreichend. Ist im Beispiel von eben etwa die Wahrscheinlichkeit dafür, dass die Person das Mobiltelefon bei sich trägt, 0.8, so ist die Wahrscheinlichkeit dafür, dass sie sich an dem durch das Mobiltelefon bestimmten Ort befindet, ebenfalls 0.8.

Kontextregeln werden in einer Kontextwissensbasis gespeichert (Vgl. die Abschnitte 4.1 und 4.3.3) und von Inferenzsystemen ausgewertet. Siehe dazu die Arbeit von Jing Mei und Elena Paslaru Bontas [MB05].

Sind in der Kontextwissensbasis auch probabilistische Kontextinformationen möglich (für deren Gültigkeit eine bestimmte Wahrscheinlichkeit angegeben ist), so müssen auch die Regeln in der Lage sein, dies zu verarbeiten. Das bedeutet, dass für die Terme im Konsequenzteil jeweils ein Wahrscheinlichkeitswert angegeben wird, sobald über Aussagen in der Voraussetzung nur eine bestimmte Wahrscheinlichkeit angegeben werden kann.

Paralell werden deshalb Funktionen zur Gewichtung der jeweiligen Wahrscheinlichkeitswerte gespeichert. Für eine Beispielregel der Form

$$\text{Regel}_1 : A \circ B \circ C \rightarrow D \circ E$$

wird dann die Gewichtungsfunktion

$$f_{\text{Regel}_1}(d_A, d_B, d_C) = (d_D, d_E)$$

vorrätig gehalten. Dabei bezeichnet d_X den Wahrscheinlichkeitskoeffizienten für den Term X. Mit dieser Funktion lässt sich auch rückwärts berechnen, wie sicher eine Aussage im Voraussetzungsteil sein muss, um ein Resultat zu bekommen, das die Wahrscheinlichkeitsvorgaben einer Anforderung erfüllt.

Geltungsbereich Der Geltungsbereich von Regeln kann vielfältig sein. Allgemeine Regeln gelten auf Ontologieebene, während private Regeln, die nur für bestimmte Anwendungen oder Benutzer gelten, der Wissensbasis extra injiziert werden müssen.

Wirkungsbereich Allgemein gültige Regeln können nur auf Klassenebene formuliert werden, die nur Variablen für Entitäten enthalten. Sobald konkrete Entitäten in einer Regel formuliert werden, ist sie zwar immer noch allgemein gültig, aber nicht mehr allgemein anwendbar.

Kosten Regeln ermöglichen es, implizites Wissen explizit zu machen. Das schließt ein, dass sie nur Wissen umformen, das bereits in der Kontextwissensbasis enthalten ist. Es fallen also keine Kosten für die externe Suche nach Diensten und Informationen und deren Übertragung an, sondern lediglich Berechnungskosten. Dazu gehört das Suchen nach geeigneten Regeln und deren Anwendung. Je nach Komplexität und Menge der Regeln kann der Aufwand dafür sehr groß werden.

Erfolgschancen Da Regeln nur auf bestehendes Wissen zurückgreifen, gibt es keine Gefahr des Scheiterns wegen fehlender Informationen. Scheitern kann diese Strategie nur aus zwei Gründen: Zum einen, wenn es keine geeignete Regel gibt, die aus dem zur Verfügung stehenden Wissen die geforderte Kontextinformation schließen kann, und zum anderen, wenn die zu Grunde liegenden Informationen zwar vorhanden sind, aber so unsicher, dass die Wahrscheinlichkeit der resultierenden Information zu gering wird, um die Anforderung zu erfüllen. Den diesbezüglichen Grenzwert bestimmt der anfragende Dienst.

Prognosechancen Da diese Strategie allein auf Berechnung beruht, lassen sich auch die Erfolgschancen exakt berechnen. Die Aufgabe besteht lediglich darin, Algorithmen zu finden, die die Berechnung dieser Prognose soweit vereinfachen, dass der Aufwand wesentlich geringer ist, als die eigentliche Berechnung selbst.

7.2.5 Bayessche Netze

Die bisherigen Ansätze, Bayessche Netze für Kontextwissen einzusetzen, zielen darauf ab, eine komplette Kontextwissensbasis in ein Bayessches Netz zu transformieren [GPZ04] oder eine komplette Ontologie, innerhalb welcher probabilistische Abhängigkeiten formuliert worden sind, in ein Bayessches Netz zu übersetzen [DP04].

Für den Anwendungszweck, einzelne Kontextinformationen abzuschätzen, sind beide Herangehensweisen nicht hinreichend. Zum einen ist der Overhead durch die Umwandlung kompletter Wissensbasen beziehungsweise vollständiger Ontologiedateien zu groß, wenn nur einzelne Kontextinformationen gesucht werden. Während der eine Ansatz sich auf die Modellebene und der andere sich auf die Instanzebene bezieht, bleibt unbeantwortet, wie Vorschriften für Bayessche Netze von der Modellebene auf die Kontextwissensbasis instanziiert werden.

Im Folgenden wird vorgeschlagen, keine kompletten Kontextwissensbasen oder komplette Ontologien zu betrachten, sondern nur geeignet gewählte Ausschnitte, die aus zwei Gründen so klein wie möglich sein sollten: Zum einen, um den notwendigen Aufwand bei der Berechnung der Wahrscheinlichkeiten in den Bayesschen Netzen zu minimieren. Und zum anderen, um die Chance zu erhöhen, für einen möglichst großen Anteil der Knoten in dem Bayesschen Netz über beobachtetes Wissen zu verfügen, statt „nur“ die a-priori-Wahrscheinlichkeiten benutzen zu können.

Dazu werden Bayessche Netze um einen Voraussetzungsteil erweitert. Erst wenn eine Kontextwissensbasis diesen Voraussetzungsteil erfüllt, kann das Bayessche Netz entsprechend instanziiert werden. Wird dieser Vorausetzungsteil mehrmals erfüllt, kann das Netz auch entsprechend mehrmals über den jeweiligen Stellen instanziiert werden. Abbildung 7.10 zeigt ein Beispiel dafür. Ziel dieses Bayesschen Netzes ist es, eine Abschätzung dafür zu bekommen, ob eine Person ihr Mobiltelefon bei sich trägt oder nicht. Die Wahrscheinlichkeit wird davon beeinflusst, ob es sich um ein privates oder um ein Diensttelefon handelt, und davon, ob der aktuelle Tag ein Werktag ist oder nicht.

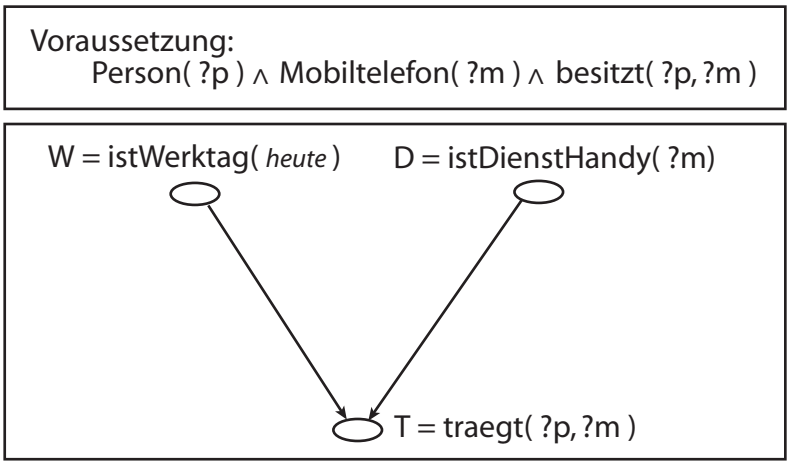


Abbildung 7.10: Bayessches Netz mit Voraussetzungsteil.

Hat man diese Vorlage für ein Bayessches Netz und eine Kontextwissensbasis, in der die in Abbildung 7.11 gezeigten Kontextinformationen enthalten sind, dann lassen sich – wenn es keine weiteren Einschränkungen gibt – aus der Vorlage drei Bayessche Netze wie angegeben darüber instanziieren. Im ersten Netz wird dabei P mit A substituiert und M mit $M1$. Im zweiten Netz wird P mit B substituiert, M mit $M2$, und außerdem ist bekannt, dass $D = true$. Im dritten Bayesschen Netz ist ebenfalls Person B das Substitut für P, M wird allerdings mit $M3$ belegt. Person C und Mobiltelefon $M4$ erfüllen die Voraussetzung $besitzt(C, M4)$ nicht, deswegen kann über ihnen aus der Vorlage kein Bayessches Netz instanziiert werden.

Die Resultate der instanziierten Bayesschen Netze zeigt Abbildung 7.12. Aus dem ersten Bayesschen Netz wird geschlossen, dass Person A mit einer Wahrscheinlichkeit von (beispielsweise) 87,4% ihr Mobiltelefon $M1$ trägt. Gleiches gilt für Person B und ihr Mobiltelefon $M3$. Bezüglich $M2$ ist aber zudem bekannt, dass es sich um ein Diensthandy handelt. Deshalb liefert das Bayessche Netz einen anderen Wert, beispielsweise 62,5%. Diese neuen Informationen werden Teil der Kontextwissensbasis. Da es sich um unsichere Informationen handelt, wird die Wahrscheinlichkeit des Zutreffens ebenfalls vermerkt. Anwendungen, denen eine Sicherheit von 62,5% beziehungsweise 87,4% für die Information genügt, ob eine Person ein bestimmtest Handy trägt, können mit diesen Kontextinformationen arbeiten. Außerdem wird in der Kontextwissensbasis eingetragen (in der Abbildung durch die Pfeile für die Relation „cmp:isDerivedFrom“), welche anderen Kontextinformationen Grundlage für die Schätzung war. Ändern sich diese Kontextinformationen oder verlieren sie ihre Gültigkeit, etwa weil sie zu alt sind, dann ändert sich auch die geschätzte Kontextinformation entsprechend.

Wird die Kontextinformation direkt für eine bestimmte Person und unter Umständen auch für ein bestimmtes Mobiltelefon angefragt, so werden diese Entitäten zunächst in die Voraussetzungen eingesetzt und dann mit der Kontextwissensbasis abgeglichen. Da-

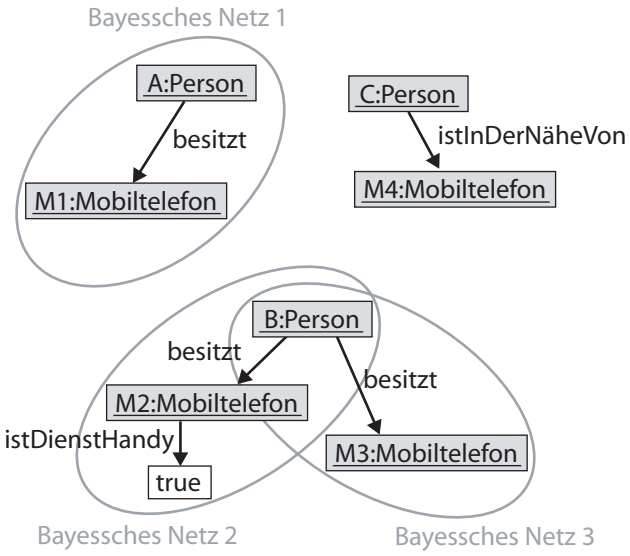


Abbildung 7.11: Instanziierung von Bayesschen Netzen.

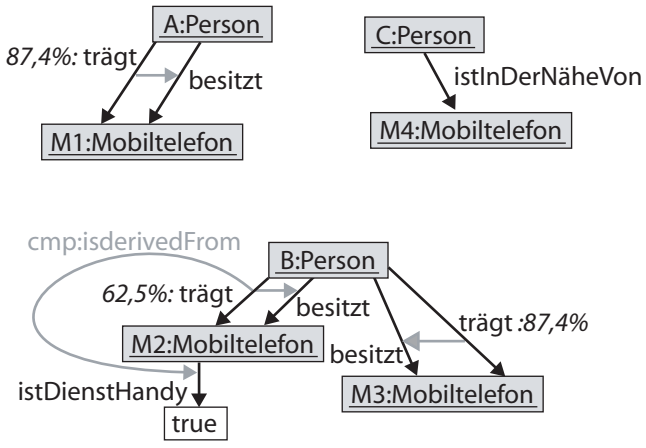


Abbildung 7.12: Aus den Bayesschen Netzen ergeben sich Schätzwerte für Kontextinformationen.

nach wird logischerweise das Bayessche Netz nur einmal instanziiert. Im Beispiel können die Zufallsvariablen, die für Kontextaussagen stehen, nur die Wahrheitswerte „*wahr*“ und „*falsch*“ annehmen. Damit lassen sich nur Kontextinformationen fassen, die eine Beziehung zwischen zwei Entitäten beschreiben (diese besteht oder besteht nicht), oder lediglich einen booleschen Datenwert annehmen können.

Tatsächlich ist diese Einschränkung unnötig (und dient im Beispiel auch nur der Einfachheit). Wie in Abschnitt 7.1.3 erklärt, können die Zufallsvariablen, die durch die Knoten repräsentiert werden, auch mehrere diskrete Werte annehmen oder sogar kontinuierliche Werte. Um den Rechenaufwand in Grenzen zu halten, ist es allerdings ratsam, einen kontinuierlichen Wertebereich durch Aufgliedern in Intervalle auf diskrete Werte abzubilden. Jeder diskrete Wert steht dann für ein Intervall (zum Beispiel für das Intervall $[15.0 - 18.0[$ bei Temperaturangaben in Grad Celsius). Dies funktioniert, wenn es mindestens eine Ordnung auf dem Wertebereich gibt.

Geltungsbereich Die Vorlagen der Bayesschen Netze gelten allgemein. Sie lassen sich für alle Ausschnitte von Kontextwissensbasen instanziieren, die die jeweilige Voraussetzung erfüllen. Individualisiert auf Anwendung oder Benutzer werden Vorlagen, indem die Variablen ihres Voraussetzungsteils mit Individuen substituiert werden, bevor sie instanziiert werden.

Individualisierte Vorlagen sind nur dann sinnvoll, wenn die Existenz von Abhängigkeiten oder der Wert von bedingten Wahrscheinlichkeiten nicht allgemein gültig sind. Individualisierte Vorlagen erhält der Kontextvermittler auf zwei Wegen: Entweder sie werden ihm von extern zur Verfügung gestellt, oder er optimiert allgemeine Vorlagen durch Training (Lernen der bedingten Wahrscheinlichkeiten durch statistischen Vergleich mit beobachteten Werten) für bestimmte Individuen. Dies ist nur sinnvoll für Werte, die häufig abgefragt werden.

Wirkungsbereich Die Vorlagen für Bayessche Netze werden im Allgemeinen auf Klassenebene der Kontextinformationen definiert und auf der Instanzebene der Kontextinformationen instanziiert. Für individualisierte Vorlagen werden die Definitionen auf bestimmte Individuen eingeschränkt.

Kosten Da diese Strategie ohne die Abfrage von Kontextinformationsdiensten gelingt, fallen nur Berechnungskosten an. Wie groß die sind, ist abhängig von der Komplexität des Bayesschen Netzes. Da die Berechnung eines Bayesschen Netzes, wie bereits erläutert, ein NP-hartes Problem werden kann, ist außerdem die Güte des Näherungsalgorithmus entscheidend. Zu den Berechnungskosten gehört das Durchsuchen der Vorlagen für Bayessche Netze und das Durchsuchen der Kontextwissensbasis nach Ausschnitten, die den Voraussetzungen einer geeigneten Vorlage entsprechen. Also sind auch Größe des Vorlagenspeichers und der Kontextwissensbasis entscheidend für die Kosten dieser Strategie.

Erfolgschancen Der Erfolg des Abschätzens von Kontextinformationen mit Hilfe von Bayesschen Netzen ist abhängig von drei Faktoren:

- Gibt es eine geeignete Vorlage für ein Bayessches Netz, das die gewünschte Kontextinformation als Ziel hat?
- Gibt es einen entsprechenden Ausschnitt in der Kontextwissensbasis, damit die Vorlage auch instanziiert werden kann?
- Besitzt das Ergebnis eine Sicherheit, die hoch genug ist, um von der anfragenden Anwendung als Kontextinformation genutzt zu werden?

Das bedeutet, es sind allein interne Faktoren, die darüber entscheiden, ob eine Kontextanfrage mit dieser Strategie beantwortet werden kann.

Dass mit Bayesschen Netzen nur geschätzte Informationen geliefert werden können und keine beobachteten, ist natürlich ein Nachteil dieser Strategie. Dem ist entgegen zu halten, dass auch bei Kontextinformationen, die von logischen oder physikalischen Sensoren abstammen, durch Messungenauigkeiten, Messfehler oder nicht vertrauenswürdige Quellen die Sicherheit weit entfernt ist vom Absoluten. Letztlich ist es die Frage der Anwendung, welche Wahrscheinlichkeit sie für das Zutreffen einer Kontextinformation als notwendig mindestens voraus setzt. In sehr vielen Fällen sind geschätzte Informationen sehr viel hilfreicher als gar keine Informationen zu haben.

Prognosechancen Da nur interne Faktoren über den Erfolg entscheiden, kann auch eindeutig prognostiziert werden, ob diese Faktoren zum Erfolg führen werden. Die Frage ist nur, wie aufwändig es sein wird, diese Voraussage zu erstellen. Im schlechtesten Fall sind die Berechnungskosten so hoch, wie die der gesamten Abschätzung.

7.3 Kombination aller Strategien

In diesem Abschnitt wird aufgezeigt, wie die eben erläuterten Strategien mit Kontextkonstruktionsbäumen [KSLP07] geeignet miteinander kombiniert werden können.

7.3.1 Signaturen mit Antezedens- und Konsequenzteil

Vorlagen für Bayessche Netze lassen sich durch die Einführung des Voraussetzungsteils beschreiben durch eine Signatur mit Antezedens- (Voraussetzung) und Konsequenzteil. Nennen wir das Beispiel aus dem vorangegangenen Abschnitt (Abbildung 7.10) Bayes_1, so lautet die Signatur

$$\text{Bayes}_1(A_1, K_1)$$

mit dem Antezedensteil

$$A_1 = (Person(?p) \wedge Mobiltelefon(?m) \wedge besitzt(?p, ?m))$$

und dem Konsequenzteil

$$K_1 = (trägt(?p, ?m)).$$

Dasselbe gilt (wie bereits in Abschnitt 4.3.3 erläutert) für Regeln. Die Beispielregel aus Abschnitt 7.2.4 – nennen wir sie Regel_2 – lässt sich ebenfalls mit einer zweigeteilten Signatur schreiben:

$$\text{Regel}_2(A_2, K_2)$$

mit dem Antezedensteil

$$A_2 = (\ Person(?p) \wedge \text{Mobiltelefon}(?m) \wedge Ort(?o) \wedge traegt(?p, ?m) \wedge istAmOrt(?m, ?o))$$

und dem Konsequenzteil

$$K_2 = (istAmOrt(?p, ?o)).$$

Sogar Kontextkompositionsgraphen lassen sich durch eine Antezedens-Konsequenz-Signatur beschreiben. Die folgende Signatur beschreibt den Beispielgraphen – nennen wir ihn CoCo_3 – aus Abbildung 7.1:

$$\text{CoCo}_3(A_3, K_3)$$

mit dem Antezedensteil

$$A_3 = (Person(?p))$$

und dem Konsequenzteil

$$K_3 = (Ort(?o) \wedge boolean(?i) \wedge istAmOrt(?p, ?o) \wedge istSchoenesWetter(?o, ?i)).$$

In allen drei Fällen, sind die Elemente der Antezedens- und Konsequenzteile Variablen, die für Entitäten stehen, individuelle Entitäten und Kontextinformationen (binäre Prädikate).

Durch die analogen Signaturen lassen sich die drei Konstrukte bei der Suche nach einer Kontextinformation kombinieren. Diese Suchsystematik soll nun erläutert werden.

7.3.2 Algorithmus: Bildung von Konstruktionsbäumen

Gegeben sei eine Kontextwissensbasis und daneben ein Speicher von Signaturen mit Antezedens- und Konsequenzteil wie sie eben beschrieben worden sind. Diese stehen für Vorschriften für Regeln, Kontextkompositionsgraphen und Vorlagen für Bayessche Netze. Gesucht sei eine Kontextinformation, die nicht auf direktem Wege von einem Kontextinformationsdienst zu erlangen ist. Ziel ist es, Konstruktionsbäume zu erstellen, die alternative Wege aufzeigen, zu der gewünschten Kontextinformation zu gelangen. Dazu ist der folgende Algorithmus zu durchlaufen:

Abbildung 7.13: Die Wurzel mit der Kontextanforderung.

Phase I: Die Wurzel

Die Anforderung der gesuchten Kontextinformation bildet die Wurzel eines Konstruktionsbaumes.

Beispiel Es wird der Ort gesucht, an dem sich die Person namens „Michael Krause“ befindet. Formal logisch vollständig und ausführlich geschrieben sähe das zum Beispiel so aus:

$$Ort(?o) \wedge Person(P) \wedge IDName(P, \text{„}Michael\ Krause\text{“}) \wedge istAmOrt(P, ?o)$$

wobei die Variable ?*o* gesucht ist. Der Übersichtlichkeit halber wird das im Folgenden (wie auch in Abbildung 7.13) abgekürzt auf den Term $istAmOrt(P, ?o)$ mit $P = \text{„}Michael\ Krause\text{“}$.

Phase II: Suche geeigneter Vorschriften

In dieser Phase wird nach geeigneten Vorschriften zur Bereitstellung der im Vaterknoten formulierten Kontextanforderung gesucht (die nicht in der Kontextwissensbasis enthalten ist). Die erste Möglichkeit ist dabei immer die explizite Anforderung von externen Kontextinformationsdiensten. Diese Anforderung wird als Kindknoten hinzugefügt und der Ast dann als fertig gestellt markiert. Diesen Ast nennen wir „finalisiert“.

Für jede weitere geeignete Vorschrift wird ein Duplikat des Baumes erstellt, wie er vor Beginn der Phase aussah, und die betreffende Vorschrift als Kindknoten hinzugefügt. Um geeignete Vorschriften zu finden, werden die Konsequenzteile aller Signaturen durchsucht, ob sie eine passende Kontextinformation liefern. Von Signaturen, die diese Voraussetzung erfüllen, werden nun die Voraussetzungsteile untersucht, ob die Kontextwissensbasis die Entitäten enthält, um die Vorschrift zu instanziieren. Für jede Instanz der Vorschrift wird ein eigener Baum erstellt (wie beschrieben) mit dessen aktuellem Kindknoten dann weiter verfahren wird wie in Phase III, sofern dieser nicht finalisiert ist.

Beispiel Die in vorausgegangenen Abschnitten als „Regel$_2$“ beschrieben Regel produziert im Konsequenzteil die Kontextinformation „istAmOrt“ für eine Person. Dies entspricht der Kontextanforderung im Vaterknoten (der hier in Fortführung des Beispiels von Phase I auch Baumwurzel ist). Im Voraussetzungsteil wird eine Entität der Klasse Mobiltelefon verlangt. Für dieses Beispiel sei angenommen, dass die Kontextwissensbasis

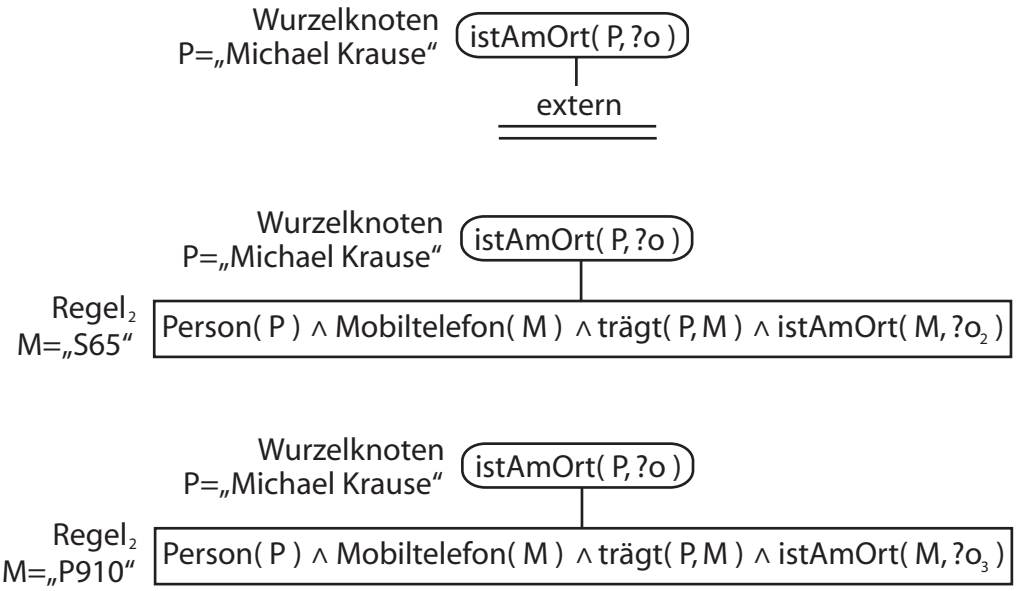


Abbildung 7.14: Verdreifachung der Konstruktionsbäume nach Phase II. Der Doppelstrich markiert die Finalisierung des ersten Baumes.

zwei Entitäten dieser Klasse enthält, die hier als „S65“ und „P910“ identifiziert werden sollen. Also kann die Regel zweimal angewandt werden durch jeweils entsprechende Belegung der Variablen. Das Resultat nach Durchlaufen dieser Phase beschreibt Abbildung 7.14. Sie enthält zweimal die gleiche Regel, die allerdings angewandt auf zwei Verschiedene Entitäten (zwei Mobiltelefone) zwei verschiedene Ergebnisse (zwei Orte) liefert. In den Knoten für die Anwendung der „$Regel_2$“ ist der Antezedensteil aufgeführt und die konkrete Entität für welche die Regel angewandt wird. Nur die beiden nicht finalisierten Konstruktionsbäume durchlaufen danach Phase III.

Phase III: Vervollständigen des Antezedensteils

Alle Kontextinformationen aus dem Voraussetzungsteil der betrachteten Vorschrift, die direkt der Kontextwissensbasis zu entnehmen sind, sind als vorhanden zu markieren. Sind alle Kontextinformationen des Voraussetzungsteils vorhanden, so ist der Ast dieses Knotens als finalisiert zu markieren.

Für alle noch fehlenden Kontextinformationen ist unter dem Knoten jeweils ein Kindknoten mit der entsprechenden Kontextanforderung einzufügen. Existiert weiter vorne im Baum bereits die gleiche Kontextanforderung, so ist der gesamte Baum zu löschen, da ein Zyklus entstanden ist.

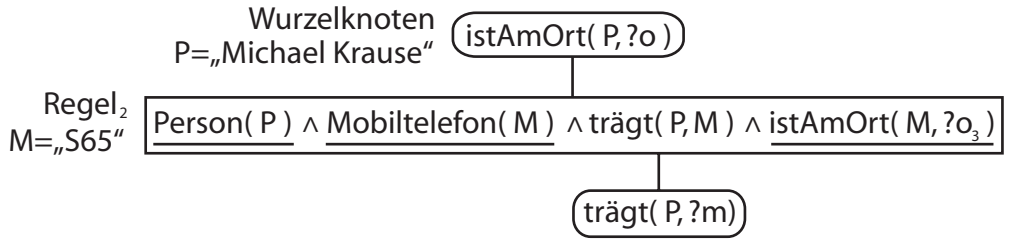


Abbildung 7.15: Konstruktionsbaum nach Phase III. Unterstrichene Teile des Antezedensteils sind in der Wissensbasis bereits enthalten.

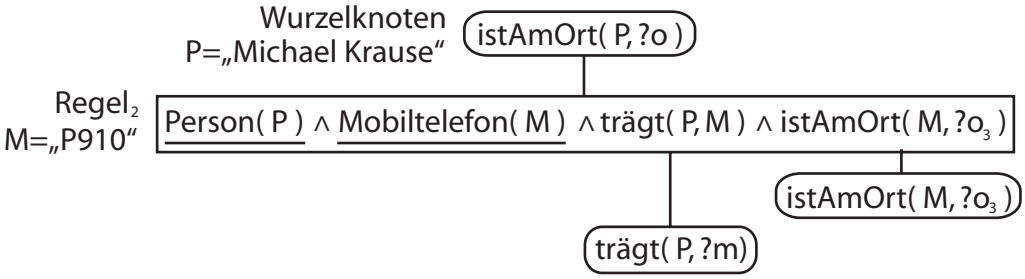


Abbildung 7.16: Weiterer Konstruktionsbaum nach Phase III.

Beispiel Für die beiden Mobiltelefone aus dem vorangegangenen Beispiel ist in der Kontextwissensbasis keine Information darüber enthalten, ob eines davon aktuell von Michael Krause getragen wird. Also wird jeweils die Kontextanfrage nach allem, was aktuell von ihm getragen wird, als Kindknoten angehängt (vgl. Abbildungen 7.15 und 7.16). Ob er genau das „S65“ oder genau das „P910“ trägt, ist so nicht als Kontextanfrage formulierbar, weswegen die Anfrage offener gehalten werden muss. Vom „S65“ sei für dieses Beispiel der Ort in der Kontextwissensbasis, vom „P910“ nicht.

Phase IV: Terminierung

Da Zyklen ausgeschlossen worden sind und die Menge an Signaturen endlich ist, wird jeder Baum, der nicht wieder gelöscht wird, in endlicher Zeit in allen Ästen finalisiert. Das Ergebnis des Algorithmus' ist eine Menge von finalisierten Konstruktionsbäumen.

Beispiel Mit der in vorangegangenen Abschnitten erläuterten Vorlage für ein Bayessches Netz „$Bayes_1$“ lässt sich abschätzen, mit welcher Wahrscheinlichkeit ein Mobiltelefon von einer Person getragen wird, die dieses Mobiltelefon besitzt. Sei angenommen, dass in der Kontextwissensbasis bekannt ist, dass Michael Krause das Telefon „S65“ besitzt, so lässt sich das Bayessche Netz instanziieren und damit abschätzen, mit welcher

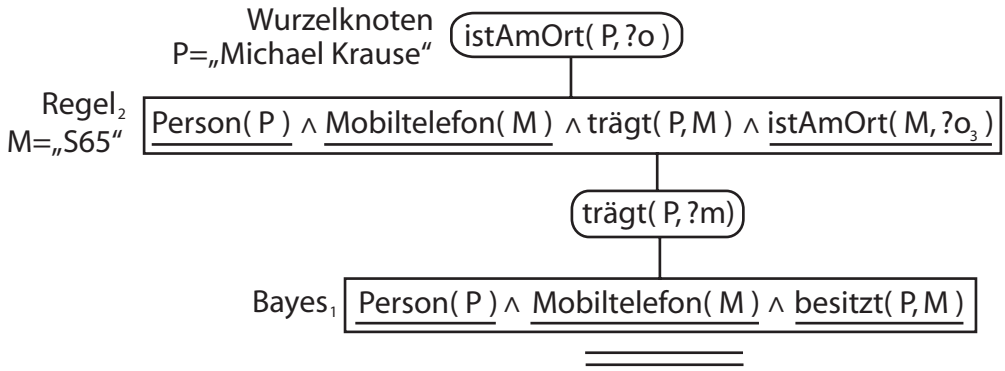


Abbildung 7.17: Finalisierter Kontextkonstruktionsbaum.

Wahrscheinlichkeit Michael Krause dieses Gerät bei sich trägt. Damit ist dieser Ast und gleichzeitig der gesamte Konstruktionsbaum finalisiert.

Sei angenommen, dass sich in der Wissensbasis keine Information darüber befindet, ob Michael Krause das „P910“ besitzt, und es keine Vorschrift gibt, die diese Kontextinformation produzieren kann, so muss diese extern besorgt werden. Ohne diese kann das Bayessche Netz nicht instanziiert werden und der Kontextkonstruktionsbaum nicht weiter abgearbeitet werden.

Neben den Bäumen in den Abbildungen 7.17 und 7.18 gibt es selbstverständlich noch weitere finalisierte Bäume nach Durchlaufen des Algorithmus, nämlich diejenigen, die jeweils statt „$Bayes_1$“ und „$Regel_2$“ die externe Anforderung der betreffenden Kontextinformation enthalten.

Über die Konstruktionsbäume

Zur Tauglichkeit der erhaltenen Konstruktionsbäume zur Bereitstellung der gewünschten Kontextinformation lassen sich je nach deren Gestalt die folgenden Aussagen treffen:

- Bäume, die außer der Kontextanforderung in der Wurzel nur einen Kindknoten mit der Anweisung enthalten, diese Information extern zu beschaffen, sind trivialer Weise wertlos. Sie ergeben keine Alternative zur Bereitstellung der Kontextinformation.
- Bäume, die mindestens eine Anforderung von externen Kontextinformationen enthalten (entweder direkt oder als Teil eines Kontextkompositionsgraphen), scheitern, wenn eine dieser Anforderungen kein Ergebnis liefert.
- Selbst Kontextkompositionsgraphen, die Kontextanfrageknoten enthalten, bedeuten nicht unbedingt eine externe Kontextanfrage – die Kontextinformation kann sich auch bereits in der Kontextwissensbasis befinden. Ist eine in einem Kontextkompositionsgraphen enthaltene Kontextinformation nicht in der Wissensbasis enthalten,

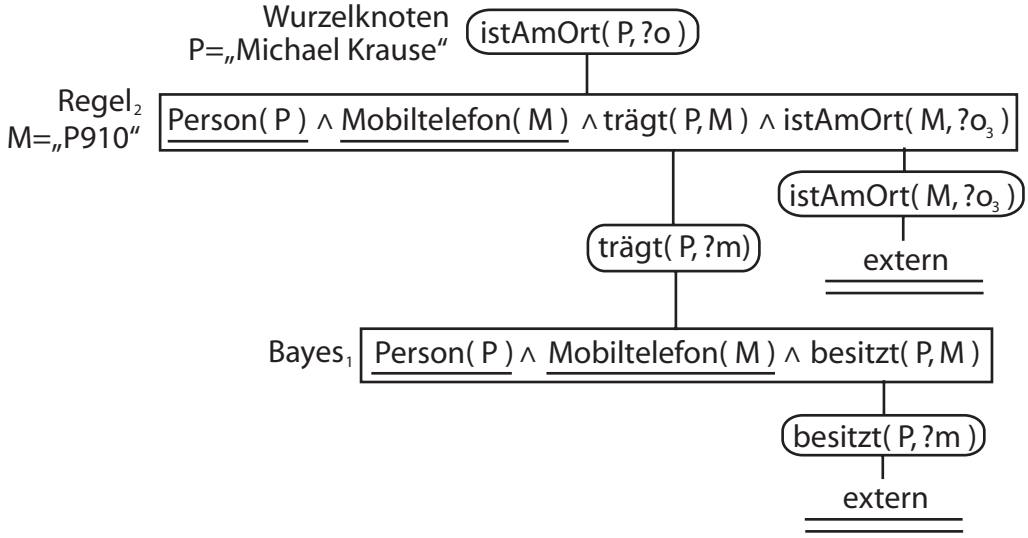


Abbildung 7.18: Weiterer finalisierter Konstruktionsbaum, allerdings mit Anforderungen externer Kontextinformationen.

so ist aus offensichtlichen Gründen zunächst ein anderer Kontextkonstruktionsbaum vorzuziehen, nämlich derjenige, der an der Stelle des Kontextkompositionsgraphen die eine direkte externe Anforderung der gesuchten Kontextinformation hat. Erst wenn diese gescheitert ist, ist es sinnvoll, das Duplikat mit dem aufwändigeren Kontextkompositionsgraphen einzusetzen.

- Bäume, die auf probabilistische Informationen zurückgreifen, können ein Ergebnis liefern, das den Anforderungen an die Wahrscheinlichkeit von dessen Gültigkeit nicht genügt. Auf probabilistische Informationen wird zurückgegriffen, wenn eine benutzte Information aus der Kontextwissensbasis mit einem Wahrscheinlichkeitskoeffizienten behaftet ist, wenn die Information durch ein Bayessches Netz erzeugt worden ist oder durch eine Regel, die in jedem Fall nur ein mit einer Wahrscheinlichkeit behaftetes Resultat bringt (Beispiel für eine solche Regel ist: $Ort(A) \wedge Ort(B) \wedge näherAls(A, B, \text{"}2km\text{"}) \wedge esRegnetAnOrt(A, \text{"}wahr\text{"}) \rightarrow esRegnetAnOrt(B, \text{"}wahr\text{"}) * 0.8$ – Sprich: Regnet es an Ort *A* und ist dieser von Ort *B* weniger als 2 Kilometer entfernt, so gilt mit einer Wahrscheinlichkeit von 80%, dass es auch an Ort *B* regnet).

- Bäume, die weder externe Aufrufe noch probabilistische Informationen enthalten, liefern in jedem Fall ein gültiges Ergebnis.

Enthält die Menge von Konstruktionsbäumen mehr als ein Element, so ist die Auswahl abhängig von der Kontextanfrage und den zu geschätzten Kosten der Konstruktion. Denn die Anfrage bestimmt, ob Kontextinformationen geliefert werden dürfen, die mit Wahrscheinlichkeiten behaftet sind. Ist eine untere Schranke für die Wahrscheinlichkeit ange-

geben, so kann unter Umständen anhand der in die Konstruktion einfließenden Kontextinformationen abgeschätzt werden, welche Konstruktionsbäume ein Ergebnis liefern, das diese Bedingung noch erfüllt.

7.4 Zusammenfassung

In diesem Kapitel ist mit den Kontextkonstruktionsbäumen eine Methode vorgestellt worden, wie Kontextinformationen anders als auf dem direkten Weg von externen Kontextinformationsdiensten beschafft werden können. Ein ebenfalls entwickelter Algorithmus beschreibt, wie zur Erstellung dieser Bäume Kontextkompositionsgraphen, logische Regeln und Bayessche Netze kombiniert werden können. Die einzelnen Strategien sowie ihre Kombination sind hinsichtlich ihrer Erfolgschancen und des zu erwartenden Aufwands diskutiert worden.

Damit ist eine Grundlage erarbeitet worden, wie entsprechende Systeme diese Strategien entsprechend der individuellen Anforderungen geeignet implementieren können. Abzuwägen ist dabei immer, wie viel Aufwand man betreiben möchte, um auf alternativem Weg eine Kontextinformation zu erhalten.

8 Prototypische Implementierung und Evaluation

> It gets worse. I have, before now, waited for a pen to perform a macro.
>
> *(Terry Pratchett)*

Wie in Abschnitt 2.2 bereits ausgeführt, gibt es bis zur Realisierung der Vision des Ubiquitous Computing in einem Maße bis sie der Beschreibung Mark Weisers gerecht wird, noch sehr viele offene Punkte. Dazu gehören die Fragen,

- welche und wie viele Schnittstellen-Standards sich für die beteiligten Dienste und deren Kommunikation sich herausbilden werden,
- welche tragfähigen Geschäftsmodelle erkannt werden (dazu gehören auch so genannte „Killerapplikationen“),
- welche Eigenschaften von den Benutzern überhaupt akzeptiert werden (da spielen die Frage der Kosten, der Bedienbarkeit, der gesellschaftlichen inklusive der juristischen Einbettung und die große Frage nach der Privatsphäre eine Rolle),
- wie die verfügbare Netz-Infrastruktur sowohl topologisch als auch bezüglich der angebotenen Funktionalität gestaltet sein wird und
- wie die unternehmerische Landschaft dann geprägt sein wird. Dazu gehören die Hardware-Anbieter, die Netzbetreiber und vor allem die Dienst- und Informationsanbieter.

Zusammenfassend ist zu sagen: Das auch vom Autor dieser Arbeit angestrebte ubiquitäre und offene System, das groß und verbreitet genug ist, um die gewünschten Netzeffekte (z.B. Ressourcennutzung über Domänengrenzen hinweg, ubiquitäre Dienst- und Informationsverfügbarkeit etc.) zu erzielen, gibt es noch nicht. Es muss sich zunächst entwickeln.

Aufgrund der vielen aufgezählten Unbekannten in diesem Entwicklungsprozess kann über jedes jetzt aufgestellte, konkrete Szenario eines mit Gewissheit gesagt werden: Es wird nicht mit der tatsächlichen Entwicklung übereinstimmen.

Solange die Entwicklung nicht weiter ist, sind Anforderungen an die Skalierbarkeit der Dienste, an die Rechenleistung der Geräte, die Bandbreite der Übertragungstechnologien, die Größe der Antwortzeiten wenig stichhaltig. Aus diesem Grund ist für die Entwicklung der in dieser Arbeit vorgestellten Technologien eine prototypische *Proof-of-Concept*-Implementierung als Evaluierung gewählt worden. Weiter gehende Simulationen entbehrten der sie rechtfertigenden Aussagekraft aus den genannten Gründen.

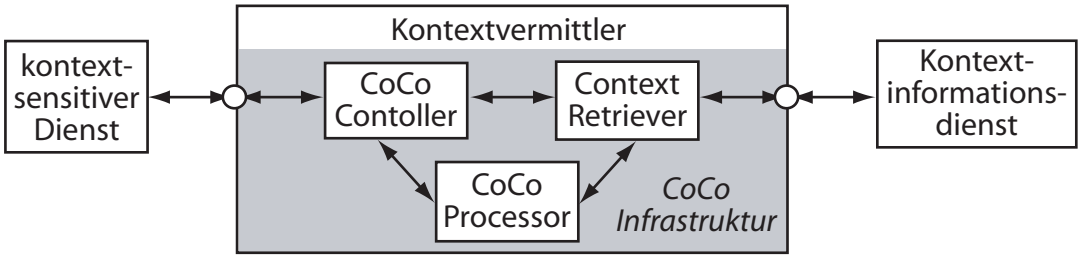


Abbildung 8.1: Vereinfachter Überblick über die CoCo-Infrastruktur.

8.1 Die CoCo-Infrastruktur

Laborumgebung eines Kontextvermittlers war dabei die Referenz-Implementierung der CoCo-Infrastruktur (siehe Abb. 8.1), die bereits in mehreren Arbeiten [Kra03, KH05, Buc05] vorgestellt worden ist.

Die Aufgaben der drei Hauptkomponenten der CoCo-Infrastruktur sind:

- Der *CoCo Controller*, der Kontextanfragen im Format eines Kontextkompositionsgraphen erhält, parst und abarbeitet. Für einzelne Anfragen beauftragt er den *Context Retriever*. Zur Umwandlung der Kontextinformationen in andere Repräsentationsformate und zur Durchführung von Ersetzungen, falls einzelne Knoten des Kontextkompositionsgraphen keine direkten Ergebnisse erhalten, beauftragt er den *CoCo Processor*.

- Der *Context Retriever* sucht sich einen oder mehrere passende Kontextinformationsdienste für eine Anfrage nach einer einzelnen Kontextinformation und fragt diese nach der Kontextinformation an. Die resultierende Kontextinformation oder eine Fehlermeldung gibt er zurück an den *CoCo Controller*.

- Der *CoCo Processor* verfügt über das Modellwissen. Er speichert die Ontologien beziehungsweise die Links zu Ihnen; er beinhaltet die Kontextwissensbasis, in der Kontextinformationen gespeichert werden und Methoden, um Kontextinformationen zu vergleichen oder zu verarbeiten. Deshalb wird er sowohl vom *CoCo Controller* als auch vom *Context Retriever* herangezogen, wenn Ontologiewissen (wie zum Beispiel Äquivalenzen zwischen Klassen) oder Verarbeitungen notwendig sind.

Die Java-Implementierung stützt sich auf die J2EE-Spezifikation (Version 5, hieß zuvor Version 1.5) [Sun06b] und läuft innerhalb eines Tomcat-Application-Servers (Version 5.5) [Apa06]. Zur Übertragung wird Simple-Object-Access-Protocol SOAP verwendet als SOAP-Engine und SAX-Parser für XML dient Apache AXIS [Apa05]. Zu weiteren Einzelheiten wie etwa zur Funktionsweise des Parsers, der Dokumente mit Kontextkompositionsgraphen in Java-Objekte umwandelt, sei auf Kapitel 3.4 in der Arbeit „Skalierbare kontextsensitive Dienste“ von Thomas Buchholz verwiesen. Zur Unterstützung bei der Erstellung von Ontologien wird der OWL-Editor der Plattform Protégé 3.1 [Ins05] verwendet.

8.1.1 Anbindung der Kontextinformationsdienste

Als Quelle für Kontextinformationsdienste stehen gekapselte physische und logische Sensoren (wie z.B. Web-Services) zur Verfügung:

Kontextinformationsdienst „Sensordata“ Mit zwei Multisensoren der österreichischen Firma „Medhost“ werden Daten über die Luftfeuchtigkeit, die Temperatur und die Helligkeit gemessen. Ein Kontextinformationsdienst bietet diese Kontextinformationen bezogen auf eine Ortsentität an, nämlich den entsprechenden Laborraum im Institut für Informatik [Dün05].

Kontextinformationsdienst „Weatherdata“ Datenlieferant für diesen Kontextinformationsdienst ist der Webservice „GlobalWeather“ von der Website www.webservicex.net. Dieser Webservice liefert für internationale Städte aktuelle Wetterdaten wie Temperatur, Luftfeuchtigkeit, Wind- und Sichtverhältnisse. Außerdem bietet er eine Methode, die pro Land eine Liste der Städte zurückgibt, für die Wetterdaten geliefert werden können [Dün05].

Kontextinformationsdienst „Stockdata“ Aktiendaten werden nicht von einem Webservice, sondern von einer „.csv“-Datei der Website finance.yahoo.com entnommen. Diese Kursdaten, die für Aktien aus dem DAX oder dem Dow Jones zur Verfügung stehen, sind in der Regel nicht älter als 20 Minuten. Diese Daten werden ebenfalls in einem Kontextinformationsdienst angeboten.

Kontextinformationsdienst „Calenderevent“ Dieser Dienst kapselt Termindaten im iCalender-Standard [DS98]. Integriert sind zum einen der persönliche Kalender des Autors, ein Kalender, der Formel-1-Termine enthält, und einer, der die deutschen Feiertage liefert [Dün05].

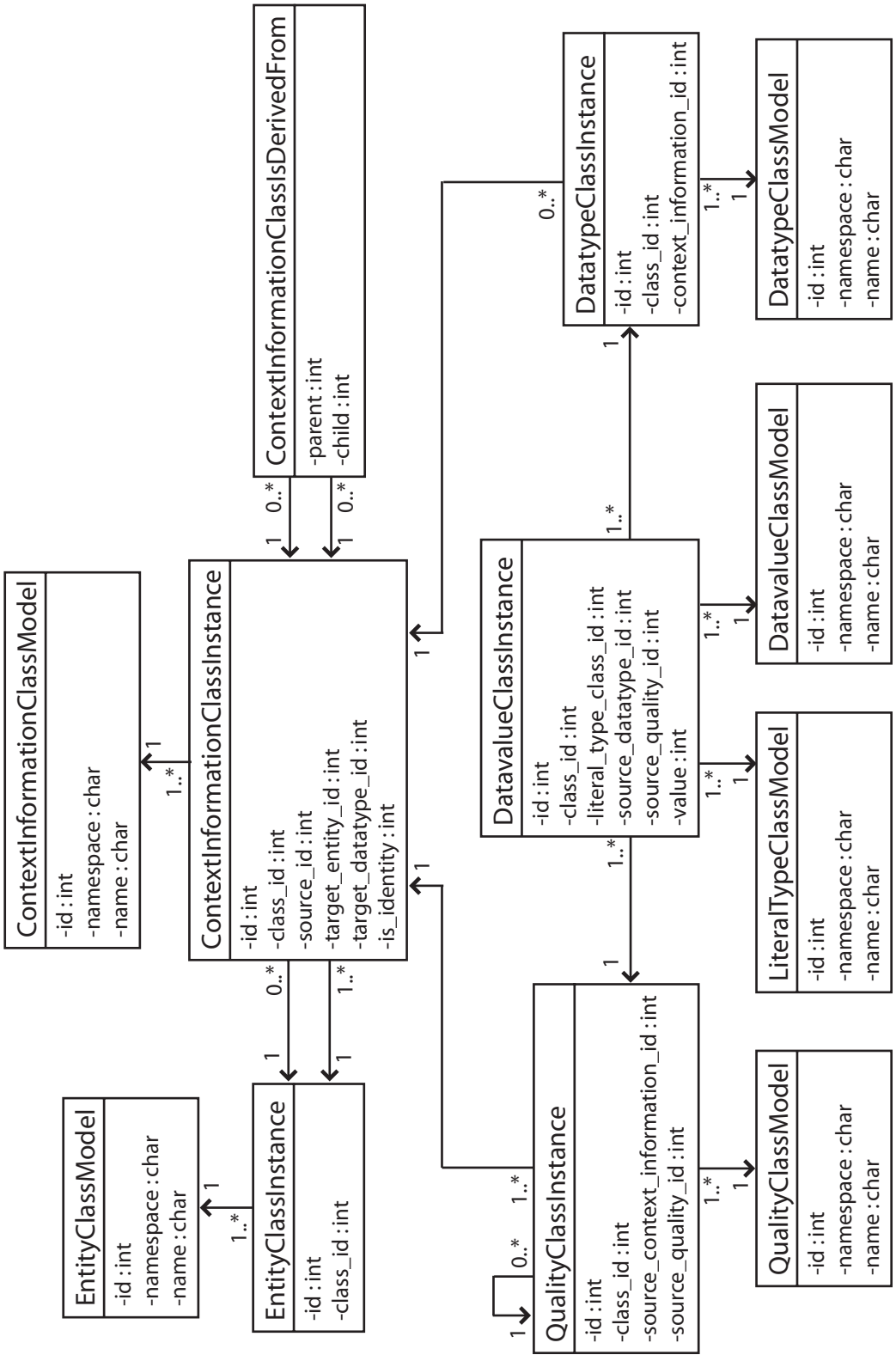


Abbildung 8.2: Datenbankschema des Kontextinformations-Caches.

Kontextinformationsdienste „Restaurantfinder“ Für einen prototypischen Restaurantfinderdienst stehen LDAP-Datenbanken mit Beispieldaten für Benutzerprofile und Restaurantdaten zur Verfügung [Wil05].

Wie in Abschnitt 6.2.6 dargelegt, wendet sich ein neuer Kontextinformationsdienst an ein Dienstverzeichnis. Dafür ausgewählt worden ist die Open-Source-Implementierung jUDDI [Apa04]. Diese implementiert zwar nur die ältere zweite Version der UDDI-Standards, lief aber zum Zeitpunkt der Auswahl stabiler als die kleinere Auswahl der Implementierungen der dritten Version. Während für die direkten Web-Service-Aufrufe das Axis-Framework verwendet wird, wird für die *push*-basierte Kommunikation bei der Suche nach geeigneten Kontextinformationsdiensten auf einen Prototyp der Indiana University, den WS-Messenger (WSMG) [Ind06b] zurückgegriffen, der die Standards *WS-Notification* (WSN) [OAS06] unterstützt und sich für das Senden von Nachrichten auf den Standard *Java Messaging System* (JMS) [Sun06a] stützt. Als JMS-Implementation kam das NaradaBrokering-System [Ind06a] zum Einsatz. Für weitere Einzelheiten sei auf die Arbeit „Vermittlung von Kontexterbringerdiensten“ [Mat05] verwiesen.

8.1.2 Die Kontextwissensbasis

Da OWL-Datenbanken und Inferenzsysteme noch nicht weit genug entwickelt sind, ist als Vorläufer einer Kontextwissensbasis ein Kontextinformations-Cache als PostgreSQL-Datenbank [Pos06] entwickelt worden. Abbildung 8.2 zeigt den wichtigsten Teil des entsprechenden Datenbankschemas und Abbildung 8.4 den Einfügevorgang für eine Kontextinformation. Eine Entität ohne Kontextinformation kann es nicht geben, da der notwendige Identifikator selbst eine Kontextinformation ist. Dagegen kann es sehr wohl mehrere Kontextinformationen zu einer Entität geben: Schon wenn die eigentliche Kontextinformation nicht identitätsstiftend ist, beinhaltet das Gesamtkonstrukt mindestens zwei. Zur Validierung der Modellierung von Kontextinformationen mit komplexen Qualitätsbedingungen ist der „QOC-Protoyp“ (siehe Abbildung 8.3) implementiert worden. Dieser stellt Beispielanfragen an den Kontextinformations-Cache und gibt dem Benutzer die Ergebnisse in einer graphischen Benutzeroberfläche zurück. Die vier Fenster der Oberfläche zeigen die Abkürzungen der verwendeten *Namespaces* (1), den aktuellen Inhalt der Datenbank (2), die aktuelle Anfrage (3) und das Ergebnis dazu (4). Näheres dazu in der Arbeit „Qualitätskriterien von Kontextinformationen“ [Alb05].

8.2 Zusammenfassung

Die prototypische Erprobung ergab nicht nur, dass sie grundsätzlich umsetzbar sind. Die praktische Erfahrung zeitigte Erkenntnisse, die sich andernfalls eventuell nicht so einfach ergeben hätten. Zum Beispiel die Einsicht, dass Identifikatoren am besten selbst als Kontextinformationen modelliert werden, oder dass statt einer einzelnen Kontextinformation auch häufig ein Netz von Kontextinformationen an derselben Entität hängend vorkommt.

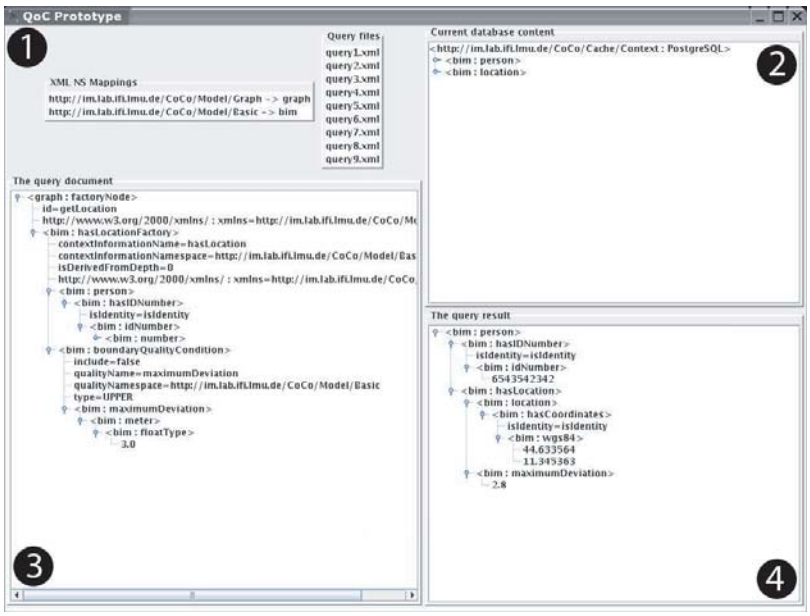


Abbildung 8.3: Screenshot des „QOC-Prototypen“.

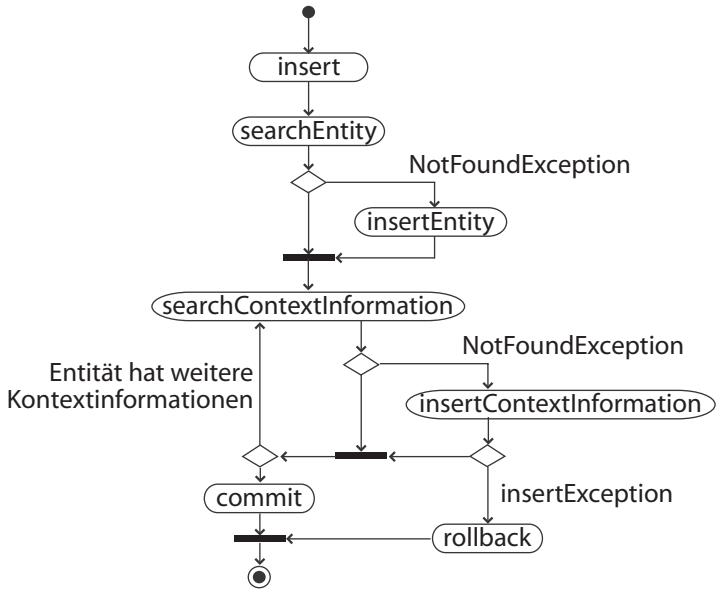


Abbildung 8.4: Einfügen einer Kontextinformation in den Kontextinformations-Cache.

9 Zusammenfassung und Ausblick

> Third Fish: Hey, look.
> Howard's being eaten.
> Second Fish: Is he? Makes you think doesn't it?
> Fourth Fish: I mean... what's it all about?
> Fifth Fish: Beats me.
>
> *(Monty Python)*

Diese Arbeit hat Lösungen vorgestellt, die für den Informationsaustausch in offenen, ubiquitären Systemen notwendig sind. Im Einzelnen sind dies

- mit CMPlus eine Modellierung von Kontextinformationen, die gegenüber bestehenden Ansätzen deutliche Vorteile bezüglich der Ausdruckskraft und der Entscheidbarkeit bietet,
- Techniken zur Suche nach Kontextinformationen, darunter vor allem mit CISP ein Profil zur Beschreibung von Kontextinformationsdiensten
- und schließlich verschiedene, aufeinander aufbauende Strategien, wie Kontextinformationen beschafft oder geschätzt werden können, die nicht direkt bereitgestellt werden. Für die Kombination dieser Strategien ist ein neuer Algorithmus zur Erstellung von Konstruktionsbäumen dargelegt worden.

Einige Aspekte mussten dabei noch außen vor bleiben, entweder weil sie den Umfang gesprengt hätten, oder weil es sich um Anschlussarbeiten handelt.

Dabei handelt es sich vor allem um den Aspekt der Verteilung. Es lohnt sich, in weiterführenden Arbeiten eine Föderationsstrategie für die Komponente des Kontextvermittlers zu erarbeiten. Ausgangspunkt dazu könnte der skizzierte Vorschlag von Johannes Mathes sein, einem Studenten, dessen Diplomarbeit der Autor mitbetreut hat. Abbildung 9.1 zeigt, wie diese durch ein Peer-to-Peer-Netz verbunden werden. Mathes schlägt dabei ein hierarchisches Peer-to-Peer-Netz vor oder eine ringartige Suche wie von Sylvia Ratnasamy et al. beschriebene [RKY+02].

Außerdem gibt es noch das Problem, dass – wie in dieser Arbeit bereits angeführt – eine einheitliche Identitätsbeschreibung von Entitäten zwar aufgrund der Vielgestalt der Entitäten und ihrer Rollen nicht praktisch möglich ist, durch die unterschiedlichen Identifikatorenformate aber identische Entitäten nicht mehr einfach als identisch erkannt werden. Ein Mapping dieser Identifikatoren ist ein weiterer Aspekt, der untersucht werden sollte.

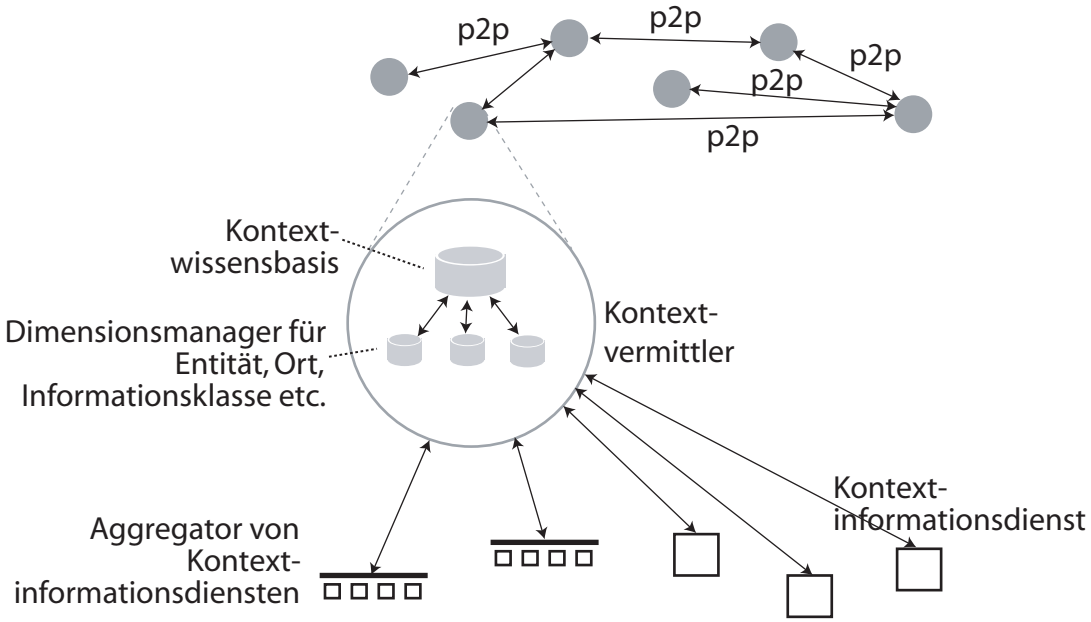


Abbildung 9.1: Föderation von Kontextvermittlern durch ein P2P-Netz nach [Mat05].

Weiter offen und in dieser Arbeit praktisch nicht betrachtet ist auch der Aspekt des Bezahlens für Kontextinformationen und das dafür notwendige Accounting- und Billing-Management. Das führt zu einem weiteren Punkt, der außerhalb der originären Informatik-Betrachtungsweise liegt: Der Betrieb der Infrastruktur, die für ubiquitäre Systeme notwendig ist, und die in den geforderten offenen Systemen auch von Domänen-Fremden genutzt werden darf, muss sich rechnen. Auch Anbieter von Kontextinformationen brauchen einen Anreiz, dies zu tun. Notwendig sind also Geschäftsmodelle für ubiquitäre Systeme.

Ubiquitäre Systeme eigenen sich wie kaum ein anderes Thema zur interdisziplinären Betrachtung. Eine Technologie, die alle Lebensbereiche betrifft und unterstützt, besitzt neben technologischen Gesichtspunkten und den erwähnten betriebswirtschaftlichen auch juristische, politische und sozialwissenschaftliche. Allen voran dabei die Frage nach dem Schutz und der Freigabe privater Daten.

Der Autor dieser Arbeit ist der Überzeugung, dass dieses Thema nicht nur über den Erfolg ubiquitärer Systeme entscheidet, sondern auch, dass es besondere Aufgabe der Informatik ist, die Vorteile und Missbrauchsmöglichkeiten zu erforschen und aufzuzeigen. Spätestens seit J. Robert Oppenheimer, dem Leiter des Manhattan-Projekts, das zum ersten Einsatz einer Atomwaffe führte, muss jedem Wissenschaftler klar sein, dass er auch eine gesellschaftliche Verantwortung trägt für die Technologien, die er hilft, nutzbar zu machen.

Schließlich sei noch ein Argument dieser Arbeit wieder aufgegriffen: Es ist dargelegt worden, von wie vielen noch unbekannten Faktoren oder ungelösten Problemen die Ver-

wirklichung ubiquitärer Systeme abhängt, wie sie zum Beispiel Mark Weiser prophezeit hat. Allen Unbekannten zum Trotz: Es ist zwar die Frage, *wie* diese Systeme ausgestaltet sein werden. Aber kommen werden sie – das ist sicher.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[ACGI03] ALLARD, J., V. CHINTA, S. GUNDALA und GOLDEN G. RICHARD III: *Jini Meets UPnP: An Architecture for Jini/UPnP Interoperability*. In: *Proceedings of the 2003 International Symposium on Applications and the Internet (SAINT 2003)*, Orlando, Florida, USA, Januar 2003.

[ADG+05] ANTONIOU, GRIGORIS, CARLOS VIEGAS DAMÁSIO, BENJAMIN GROSOF, IAN HORROCKS, MICHAEL KIFER, JAN MALUSZYNSKI und PETER F. PATEL-SCHNEIDER: *Combining Rules and Ontologies. A survey.* EU-Projektbericht „REWERSE reasoning on the web“ I3-D3, Science and Technology Park Kreta (Griechenland), Universität Lissabon (Portugal), MIT Sloan School of Managament Cambridge (USA), Universität Manchester (UK), State University New York (USA), Universität Linköping (Schweden), Bell Laboratories (USA), März 2005.

[AH05] AGARWAL, SUDHIR und PASCAL HITZLER: *Modeling Fuzzy Rules with Description Logics*. In: *Proceedings of Workshop on OWL Experiences and Directions*, Galway, Irland, November 2005.

[Alb05] ALBERT, INGMAR V.: *Qualitätsmerkmale von Kontextinformationen: Modellierung und Verwendung*. Diplomarbeit, Ludwig-Maximilians-Universität, München, Oktober 2005.

[Apa04] THE APACHE SOFTWARE FOUNDATION: *jUDDI User's Guide*, Juli 2004. ws.apache.org/juddi/usersguide.html.

[Apa05] THE APACHE SOFTWARE FOUNDATION: *Axis Architecture Guide*, 2005. ws.apache.org/axis/java/architecture-guide.html.

[Apa06] APACHE SOFTWARE FOUNDATION: *The Apache Tomcat 5.5 Servlet/JSP Container*, 2006. tomcat.apache.org/tomcat-5.5-doc/index.html.

[AVS05] ABRAMS, CHARLES, RAY VALDES und DAVID MITCHELL SMITH: *Core Web Service Standard UDDI Evolves With Version 3.0.2*. Technischer Bericht G00126170, Gartner Inc., Stanford, USA, Februar 2005.

[AWSBL99] ADJIE-WINOTO, WILLIAM, ELLIOT SCHWARTZ, HARI BALAKRISHNAN und JEREMY LILLEY: *The design and implementation of an intentional naming system*. In: *Symposium on Operating Systems Principles*, Seiten 186–201, 1999.

[Bay63] BAYES, THOMAS: *An Essay towards solving a Problem in the Doctrine of Chances*. Philosophical Transactions of the Royal Society of London, 53:370–418, 1763. Von www.stat.ucla.edu/history/essay.pdf.

[BBC97] BROWN, PETER J., JOHN D. BOVEY und XIAN CHEN: *Context-aware Applications: from the Laboratory to the Marketplace*. IEEE Personal Communications, 4(5):58–64, Oktober 1997.

[BBHS03] BAUER, MARTIN, CHRISTIAN BECKER, JÖRG HÄHNER und GREGOR SCHIELE: *ContextCube – Providing Context Information Ubiquitously*. In: *23rd International Conference on Distributed Computing Systems Workshops (ICDCSW 2003)*, Providence, Rhode Island, USA, Mai 2003.

[BBKG05] BALAKRISHNAN, DINESHBALU, MAY EL BARACHI, AHMED KARMOUCH und ROCH GLITHO: *Challenges in Modelling and Disseminating Context Information in Ambient Networks*. In: *Proceedings of the 2nd International Workshop on Mobility Aware Technologies and Applications (MATA 2005)*, Lecture Notes in Computer Science (LNCS), Montreal, Kanada, Oktober 2005. IFIP/IEEE, Springer.

[BEK+] BOX, DON, DAVID EHNEBUSKE, GOPAL KAKIVAYA, ANDREW LAYMAN, NOAH MENDELSOHN, HENRIK FRYSTYK NIELSEN, SATISH THATTE und DAVE WINER. Technischer Bericht.

[BHS93] BIBEL, WOLFGANG, STEFFEN HOLLDÖBLER und TORSTEN SCHAUB: *Wissensrepräsentation und Inferenz: eine grundlegende Einführung*. Künstliche Intelligenz. Vieweg, Wiesbaden, Braunschweig, 1993.

[BKLPS04] BUCHHOLZ, THOMAS, MICHAEL KRAUSE, CLAUDIA LINNHOFF-POPIEN und MICHAEL SCHIFFERS: *CoCo: Dynamic Composition of Context Information*. In: *In Proceedings of the First Annual International Conference on Mobile and Ubiquitous Computing (MobiQuitous) 2004*, Boston, Massachusetts, USA, August 2004. IEEE.

[BKS03] BUCHHOLZ, THOMAS, AXEL KÜPPER und MICHAEL SCHIFFERS: *Quality of Context Information: What it is and why we need it.* In: *Proceedings of the 10th International Workshop of the HP OpenView University Association (HPOVUA)*, Band 2003, Genf, Schweiz, Juli 2003. Hewlett-Packard OpenView University Association.

[BLHL01] BERNERS-LEE, TIM, JAMES HENDLER und ORA LASSILA: *The Semantic Web*. Scientific American, 284(4):34–43, Mai 2001.

[BMNPS03] BAADER, FRANZ, DEBORAH L. MCGUINESS, DANIELE NARDI und PETER F. PATEL-SCHNEIDER (Herausgeber): *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[BN03] BAADER, FRANZ und WERNER NUTT: *The Description Logic Handbook: Theory, Implementation, and Applications*, Kapitel Basic Description Logics, Seiten 47–100. Cambridge University Press, 2003.

[BNRM+01] BIEZUNSKI, MICHEL, STEVEN R. NEWCOMB, DANIEL RIVERS-MOORE, KAL AHMED, MURRAY ALTHEIM, SAM HUNTING et al.: *XML Topic Maps (XTM) 1.0*. Spezifikation, TopicMaps.Org, 2001.

[BR00] BETTSTETTER, CHRISTAN und CHRISTOPH RENNER: *A Comparison of Service Discovery Protocols and Implementation of the Service Location Protocol*. In: *Proceedings of Sixth EUNICE Open European Summer School*, Twente, Holland, September 2000.

[Bra95] BRACHTENDORF, JOHANNES: *Fichtes Lehre vom Sein. Eine kritische Darstellung der Wissenschaftslehren von 1794, 1798/99 und 1812*. Ferdinand Schöningh Verlag, Paderborn, 1995.

[BT03] BOLEY, HAROLD und SAID TABET: *RuleML Rules Lite Concrete Syntax*. Draft, The Rule Markup Initiative, September 2003. www.ruleml.org/submission/ruleslite/concretesyntax.html.

[Buc05] BUCHHOLZ, THOMAS: *Skalierbare kontextsensitive Dienste*. Doktorarbeit, Ludwig-Maximilians-Universität München, Dezember 2005.

[Caw03] CAWSEY, ALISON: *Künstliche Intelligenz*. Pearson Studium. Prentice Hall, 2003.

[CCKM01] CASTRO, PAUL, PATRICK CHIU, TED KREMENEK und RICHARD R. MUNTZ: *A Probabilistic Rool Location Service for Wireless Networked Environments*. In: *Proceedings of the 3rd International Conference on Ubiquitous Computing (UbiComp01)*, Seiten 18–34, Atlanta, USA, September 2001.

[CCM04] COHEN, NORMAN H., PAUL CASTRO und ARCHAN MISRA: *What the meaning of what is: descriptive naming of data providers in Context Weaver*. IBM Forschungsbericht RC 23245, IBM Research, Juni 2004.

[CCM05] COHEN, NORMAN H., PAUL CASTRO und ARCHAN MISRA: *Descriptive Naming of Context Data Providers*. In: *Modeling and Using Context, 5th International and Interdisciplinary Conference (CONTEXT2005)*,

Band 3554 der Reihe *Lecture Notes in Computer Science*, Seiten 112–125, Paris, Frankreich, Juli 2005. Springer.

[CCMW01] CHRISTENSEN, ERIK, FRANCISCO CURBERA, GREG MEREDITH und SANJIVA WEERAWARANA: *Web Services Description Language (WSDL) 1.1*. W3C Note, The W3 Consortium, März 2001.

[ccp04] *Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0*. W3C Recommendation, The W3C CC/PP Working Group – The W3 Consortium, Januar 2004.

[CDS04] CHALMERS, DAN, NARANKER DULAY und MORRIS SLOMAN: *Towards Reasoning About Context in the Presence of Uncertainty*. In: *Proceedings of the Workshop on Advanced Context Modelling, Reasoning and Management at UbiComp 2004*, Tokyo, Japan, September 2004.

[CFJ] CHEN, HARRY, TIM FININ und ANUPAM JOSHI: *Semantic Web in in the Context Broker Architecture*. In: *Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04)*. IEEE, IEEE Computer Society, Washington, USA, März.

[CFLGG+04] CAÑADAS, GONZALO, MARIANO FERNÁNDEZ-LÓPEZ, RAÚL GARCÍA-GARCÍA, MANUEL LAMA, ALFREDO SÁNCHEZ-ALBERCA und CARLOS ÓSCAR S. SORZANO: *Framework for automatic generation of ontology mappings*. In: *Jornadas Iberoamericanas de Ineniería del Software e Ingeniería del Conocimiento*, Seiten 293–304, Madrid, Spanien, November 2004.

[Che04a] CHEN, GUANLING: *Solar: Building a Context Fusion Network for Pervasive Computing*. Doktorarbeit, Dartmouth College, Hanover, New Hampshire, USA, August 2004.

[Che04b] CHEN, HARRY LIK: *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*. Doktorarbeit, University of Maryland, Baltimore County, New Hampshire, USA, Dezember 2004.

[Che04c] CHEN, HARRY LIK: *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*. Doktorarbeit, University of Maryland, USA, 2004.

[Chr04] CHRISTIAN, VOLKER: *Pervasive Computing: Towards Self-Aware Systems*. In: *The proceedings of the 2nd Workshop on Applications of Wireless Communications*, Seiten 12–18, University of Technology, Lappeenranta, Finnland, 2004.

[CJ02] CHAPPELL, DAVID A. und TYLER JEWELL: *Java Web Services – Using Java in Service-Oriented Architecture*. O'Reilly & Associates, April 2002.

[CK04] CHEN, GUANLING und DAVID KOTZ: *Dependency management in distributed settings*. In: *Proceedings of the First International Conference on Autonomic Computing*, New York, USA, 2004.

[CLL05] COSTA, PAULO C. G., KATRHYN B. LASKEY und KENNETH J. LASKEY: *PR-OWL: A Bayesian Ontology Language for the Semantic Web*. In: *Proceedings of Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2005)*, Galway, Irland, November 2005.

[Cos05] COSTA, PAULO C. G.: *Bayesian Semantics for the Semantic Web*. Doktorarbeit, Department of Systems Engineering and Operations Research, George Mason University, Fairfax, USA, Juli 2005.

[CPFJ04] CHEN, HARRY, FILIP PERICH, TIM FININ und ANUPAM JOSHI: *SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications*. In: *International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous 2004)*, Boston, USA, August 2004.

[CPW+99] COEN, MICHAEL, BRENTON PHILLIPS, NIMROD WARSHAWSKY, LUKE WEISMAN, STEPHEN PETERS und PETER FININ: *Meeting the Computational Needs of Intelligent Environments: The Metaglue System*. In: *Proceedings of Workshop on Managing Interactions in Smart Environments (MANSE99)*, Dublin, Irland, 1999.

[Cre05] CREGAN, ANNE: *An OWL DL Construction for the ISO Topic Map Data Model*, Mai 2005. Entwurf eines ISO Proposals.

[CvHH+01] CONNOLLY, DAN, FRANK VAN HARMELEN, IAN HORROCKS, DEBORAH L. MCGUINNESS, PETER F. PATEL-SCHNEIDER und LYNN ANDREA STEIN: *DAML+OIL (March 2001) Reference Description*. W3C Note, The W3 Consortium, Dezember 2001.

[DA99] DEY, ANIND K. und GREGORY ABOWD: *The Context Toolkit: Aiding the Development of ContextAware Applications*. In: *Proceedings of Human Factors in Computing Systems (CHI99)*, Seiten 434–441, Pittsburgh, USA, Mai 1999. ACM, ACM Press.

[DA00] DEY, ANIND K. und GREGORY ABOWD: *Towards a Better Understanding of Context and Context-Awareness*. In: *Proceedings of the Workshop The What, Who, Where, When, Why and How of Context-Awareness (as part of the 2000 Conference on Human Factors in Computing Systems (CHI 2000))*, Den Hague, Niederlande, April 2000. ACM.

[dBLK+05] BRUIJN, JOS DE, HOLGER LAUSEN, RETO KRUMMENACHER, AXEL POLLERES, LIVIA PREDOIU, MICHAEL KIFER und DIETER FENSEL: *The Web Service Modeling Language WSML*. WSML Working Draft, Oktober 2005.

[DEFS99] DECKER, STEFAN, MICHAEL ERDMANN, DIETER FENSEL und RUDI STUDER: *Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information*. In: MEERSMANN, R. (Herausgeber): *Semantic Issues in Multimedia Systems. Proceedings of DS-8*, Seiten 351–369, Rotorua, Neuseeland, Januar 1999. Kluwer Academic Publisher, Boston.

[Dey00] DEY, ANIND K.: *Architectural Support for Building Context-Aware Applications*. PhD-Thesis, Georgia Intitute of Technology, USA, 2000.

[Dey01] DEY, ANIND K.: *Understanding and Using Context*. Personal and Ubiquitous Computing, 5(1):4–7, 2001.

[DHN+04] DÜRR, FRANK, NICOLA HÖHNLE, DANIELA NICKLAS, CHRISTIAN BECKER und KURT ROTHERMEL: *Nexus – A Platform for Context-Aware Applications*. In: ROTH, JÖRG (Herausgeber): *1. Fachgespräch Ortsbezogene Anwendungen und Dienste der GI-Fachgruppe KuVS*, Seiten 15–18, 2004.

[DJM02] DEMEY, JAN, MUSTAFA JARRAR und ROBERT MEERSMAN: *A Markup Language for ORM Business Rules*. In: SCHROEDER, M. und G. WAGNER (Herausgeber): *Proceedings of the International Workshop on Rule Markup Languages fo Business Rules on the Semantic Web*, Seiten 107–128, 2002.

[Dün05] DÜNSSER, YVONNE: *Kontextinformationsdienste für die CoCo-Infrastruktur*. Systementwicklungsprojekt, Technische Universität München, München, 2005.

[DP04] DING, ZHONGLI und YUN PENG: *A Probabilistic Extension to Ontology Language OWL*. In: *Proceedings of th 37th Hawaii International Conference on System Science*, 2004.

[DS98] DAWSON, F. und D. STENERSON: *Internet Calendaring and Scheduling Core Object Specification (iCalendar)*. Standards Track RFC 2445, IETF Network Working Group, November 1998.

[DSA01] DEY, ANIND K., DANIEL SALBER und GREGORY D. ABOWD: *A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications*. Human-Computer Interaction (HCI) Journal, 16 (2-4):97–166, 2001.

[EHL01] EBLING, MARIA, GUERNEY HUNT und HUI LEI: *Issues for Context Services for Pervasive Computing*. In: *Proceedings of Middleware'01, Advanced Workshop on Middleware for Mobile Computing*, Heidelberg, November 2001.

[FAC+05] FEIER, CRISTINA, SINUHE ARROYO, EMILIA CIMPIAN, JOHN DOMINGUE, DIETER FENSEL, BIRGITTA KÖNIG-RIES, HOLGER LAUSEN, AXEL POLLERES und MICHAEL STOLLBERG: *Web Service Modeling Ontology Primer*. W3C Member Submission, Juni 2005.

[Fer02] FERSCHA, ALOIS: *Contextware: Bridging Physical and Virtual Worlds*. In: *Proceedings of Reliable Software Technologies – AE2002*, Nummer 2361 in *Lecture Notes in Computer Science*, Seiten 51–64. Springer Verlag, Berlin, 2002.

[Fer03] FERSCHA, ALOIS: *Pervasive Computing*. Datenbank-Spektrum, 3(7):48–51, Oktober 2003.

[FHKB05] FUCHS, FLORIAN, IRIS HOCHSTATTER, MICHAEL KRAUSE und MICHAEL BERGER: *A Metamodel Approach to Context Information*. In: *Proceedings of 2nd IEEE PerCom Workshop on Context Modeling and Reasoning (CoMoRea)*, Hawaii, USA, März 2005. IEEE Computer Society.

[fip] *FIPA Interaction Protocol Specifications*. Spezifikationensammlung, FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS (FIPA).

[fip04] *FIPA Agent Management Specification*. Spezifikation SC00023K, FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS (FIPA), März 2004.

[Fra05] FRANK, KORBINIAN: *Kontextintegration*. Diplomarbeit, Ludwig-Maximilians-Universität, München, Oktober 2005.

[Fuc04] FUCHS, FLORIAN: *A Modeling Technique for Context Information*. Diplomarbeit, Ludwig-Maximilians-Universität München, Dezember 2004.

[Fuk05] FUKUSHIGE, YOSHIO: *Representing Probabilisitic Relations in RDF*. In: *Proceedings of Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2005)*, Galway, Ireland, November 2005.

[GDL+04] GRIM, ROBERT, JANET DAVIS, ERIC LEMAR, ADAM MACBETH, STEVEN SWANSON, THOMAS ANDERSON, BRIAN BERSHAD, GAETANO BORRIELLO, STEVEN GRIBBLE und DAVID WETHERALL: *System Support for Pervasive Computing*. ACM Transactions on Computer Systems, 22(4):421–486, November 2004.

[GMF+02] GENNARI, JOHN H., MARK A. MUSEN, RAY W. FERGUSON, WILLIAM E. GROSSO, MONICA CRUBÉZY, HENRIK ERIKSSON, NATALYA F. NOY und SAMSON W. TU: *The Evolution of Protégé: An Environment for Knowledge-Based Systems Development*. Technischer Bericht SMI-2002-0943, Stanford University, USA, 2002.

[GN87] GENESERETH, MICHAEL R. und NILS J. NILSSON: *Logical Foundation of Artificial Intelligence*. Morgan Kaufmann, Los Altos, Kalifornien, USA, 1987.

[Got93] GOTTWALD, SIEGFRIED: *Fuzzy Sets and Fuzzy Logics*. Verlag Vieweg, Wiesbaden, Braunschweig, 1993.

[GPD99] GUTTMAN, ERIK, CHARLES E. PERKINS und MICHAEL DAY: *Service Location Protocol, Version 2*. Vorschlag zum IP-Standard RFC 2608, IETF Network Working Group, Juni 1999.

[GPZ04] GU, TAO, HUNG KENG PUNG und DA QING ZHANG: *A Bayesian Approach for Dealing with Uncertain Contexts*. In: FERSCHA, ALOIS und FRIEDEMANN MATTERN (Herausgeber): *Proceedings of the 2nd International Conference on Pervasive Computing (Pervasive 2005)*, Band 3001 der Reihe *Lecture Notes on Computer Science*, Wien, Österreich, April 2004. Springer.

[GPZ05] GU, TAO, HUNG KENG PUNG und DA QING ZHANG: *A service-oriented middleware for building context-aware services*. Journal of Network and Computer Applications, 28(1):1–18, 2005.

[GQYP03] GU, TAO, H. C. QIAN, J. K. YAO und HUNG KENG PUNG: *An Architecture for Flexible Service Discovery in OCTOPUS*. In: *Proceedings of the 12th International Conference on Computer Communications and Networks (ICCCN 2003)*, Seiten 291–296, Dallas, Texas, USA, Oktober 2003.

[Gru93] GRUBER, THOMAS R.: *A translation approac to portable ontology specifications*. Knowledge Acquisition, 5(2):199–220, Juni 1993.

[GS01] GRAY, PHILIP D. und DANIEL SALBER: *Modelling and Using Sensed Context Information in the Design of Interactive Applications*. In: *Proceedings of the 8th IFIP International Conference on Engineering for Human-Computer Interaction*, Seiten 317–336, März 2001.

[GSR+03] GLASSEY, RICHARD, GRAEME STEVENSON, MATTHEW RICHMOND, PADDY NIXON, SOTIRIOS TERZIS, FENG WANG und IAN FERGUSON:

Towards a Middleware for Generalised Context Management. In: *Workshop Proceedings of International Middleware Conference*, Seiten 45–52, Rio de Janeiro, Brasilien, Juni 2003.

[Gua98] GUARINO, NICOLA: *Formal Ontology and Information Systems*. In: *Proceedings of the first international conference on Formal ontology in Information Systems (FOIS)*, Seiten 3–15, Trient, Italien, Juni 1998. IOS Press, Amsterdam.

[GWPZ04] GU, TAO, XIAO HANG WANG, HUNG KENG PUNG und DA QING ZHANG: *An Ontology-based Context Model in Intelligent Environments*. In: *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*, San Diego, USA, Januar 2004.

[GWvB+01] GRIBBLE, STEVEN D., MATT WELSH, J. ROBERT VON BEHREN, ERIC A. BREWER, DAVID E. CULLER, N. BORISOV, STEVEN E. CZERWINSKI, RAMAKRISHNA GUMMADI, JON R. HILL, ANTHONY D. JOSEPH, RANDY H. KATZ, Z. M. MAO, S. ROSS und BEN Y. ZHAO: *The Ninja architecture for robust Internet-scale systems and services*. Computer Networks, 35(4):473–497, 2001.

[HB99] HALPIN, TERRY und ANTHONY BLOESCH: *Data modeling in UML and ORM: a comparison*. Journal of Database Management, 10(4):4–13, Oktober 1999.

[HBD04] HIRTLE, DAVID, HAROLD BOLEY und MIKE DEAN: *SWRL RuleML – Accessing SWRL Properties as „Foreign“ Atoms*. www.ruleml.org, August 2004. Version 0.6.

[HBG+05] HIRTLE, DAVID, HAROLD BOLEY, BENJAMIN GROSOF, MICHAEL KIFER, MICHAEL SINTEK, SAID TABET und GERD WAGNER: *Schema Specification of RuleML 0.9*. www.ruleml.org/0.9/, September 2005.

[HBS02] HELD, ALBERT, SVEN BUCHHOLZ und ALEXANDER SCHILL: *Modeling of Context Information for Pervasive Computing Applications*. In: *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002)*, Orlando, Florida, Juli 2002.

[Her97] HERRMANN, JÜRGEN: *Maschinelles Lernen und wissensbasierte Systeme*. Springer-Verlag, Berlin Heidelberg, 1997.

[Hes02] HESSE, WOLFGANG: *Ontologien*. Informatik Spektrum, 25(6):477–480, Dezember 2002.

[HI04] HENRICKSEN, KAREN und JADWIGA INDULSKA: *A Software Engineering Framework for Context-Aware Pervasive Computing*. In: *Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04)*, Seiten 77–86. IEEE, IEEE Computer Society, Washington, USA, März 2004.

[HIM05] HENRICKSEN, KAREN, JADWIGA INDULSKA und TED MCFADDEN: *Modelling Context Information with ORM*. In: MEERSMAN, ROBERT, ZAHIR TARI und PILAR HERRERO (Herausgeber): *OTM Confederated International Workshops and Posters (OTM 2005)*, Band 3762 der Reihe *Lecture Notes in Computer Science*, Seiten 626–635. Springer-Verlag Berlin Heidelberg, November 2005.

[HK04] HOCHSTATTER, IRIS und MICHAEL KRAUSE: *Strategies for On–The–Fly Composition of Context Information Services*. In: *Proceedings of the 11th International Workshop of the HP OpenView University Association (HPOVUA)*, Band 2004, Paris, Frankreich, Juni 2004.

[HKLPR03] HEGERING, HEINZ-GERD, AXEL KÜPPER, CLAUDIA LINNHOFF-POPIEN und HELMUT REISER: *Management Challenges of Context-Aware Services in Ubiquitous Computing Environments*. In: BRUNNER, M. und A. KELLER (Herausgeber): *Self-Management Systems: 14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM)*, Nummer 2867 in *Lecture Notes in Computer Science*, Seiten 246–259, Heidelberg, Deutschland, Oktober 2003. Springer.

[HNBH03] HEER, JEFFREY, ALAN NEWBERGER, CHRIS BECKMANN und JASON I. HONG: *liquid: Context-Aware Distributed Queries*. In: DEY, ANIND K., ALBRECHT SCHMIDT und JOSEPH F. MCCARTHY (Herausgeber): *UbiComp 2003: Ubiquitous Computing, 5th International Conference*, Band 2864 der Reihe *Lecture Notes in Computer Science*, Seiten 140–148, Seattle, USA, Oktober 2003. Springer.

[HNSG05] HÖNLE, NICOLA, UWE-PHILIPP KÄPPELER DANIELA NICKLAS, THOMAS SCHWARZ und MATTHIAS GROSSMANN: *Benefits of Integrating Meta Data into a Context Model*. In: *Third IEEE Conference on Pervasive Computing and Communications Workshops – Workshop on Context Modeling and Reasoning (CoMoRea)*, Kauai Island, Hawaii, USA, März 2005.

[Hon02] HONG, JASON I.: *The Context Fabric: An Infrastructure for Context-Aware Computing*. In: *CHI '02 extended abstracts on Human factors in computing systems*, Seiten 554–555, Minneapolis, Minnesota, USA, April 2002.

[HPS03] HORROCKS, IAN und PETER F. PATEL-SCHNEIDER: *Reducing OWL Entailment to Description Logic Satisfiability*. In: *Proceedings of the 2003 Description Logic Workshop (DL 2003)*, 2003.

[HPS04] HORROCKS, IAN und PETER F. PATEL-SCHNEIDER: *A Proposal for an OWL Rules Language*. In: *Proceedings of the 13th international conference on World Wide Web*, Seiten 723–731, New York, USA, 2004.

[IHMM04] INDULSKA, JADWIGA, KAREN HENRICKSEN, TED MCFADDEN und PETER MASCARO: *Towards a Common Context Model for Virtual Community Applications*. In: *Second International Conference on Smart Homes and Health Telematics (ICOST 2004)*, Singapur, September 2004.

[Ind06a] INDIANA UNIVERSITY, USA: *The NaradaBrokering Project @ Indiana University*, 2006. www.naradabrokering.org.

[Ind06b] INDIANA UNIVERSITY, USA: *WS-Messenger (WSMG) – IU WS-Notification and WS-Eventing Implmentation*, März 2006. www.extreme.indiana.edu/xgws/messenger/.

[Ins05] INSTITUT FÜR MEDIZINISCHE INFORMATIK, Stanford, USA: *what is protégé?*, 2005. protege.stanford.edu/overview/.

[Jen01] JENSEN, FINN: *Bayesian Networks and Decision Graphs*. Springer-Verlag, New-York, USA, 2001.

[JGB+05] JONSSON, ANNIKA, RAFFAELE GIAFFREDA, MAY-EL BARACHI, ROCH GLITHO, FATNA BELQASMI, MICHAEL SMIRNOV, MICHAEL KLEIS, CHRISTOPH REICHERT, AHMED KARMOUCH, MOHAMMED KHEDR, ANDERS KARLSSON, HEIMO LAAMANEN, HEIKKI HELIN, ALEX GALIS, ROEL OCAMPO und JENNY ZHANG: *Ambient Networks ContextWare*. Technischer Bericht IST-2002-507134-AN/WP6/D61, Wireless World Initiative - Ambient Networks, Januar 2005.

[jin03a] *Jini Architecture Specification – Version 2.0*. Technischer Bericht, SUN Microsystems Inc., Juni 2003.

[jin03b] *Jini Technology Core Platform Specification*. Technischer Bericht, SUN Microsystems Inc., Juni 2003.

[Jon02] JONSSON, MARTIN: *Context Shadow: An Infrastructure for Context Aware Computing*. In: *Proceedings of Workshop on Artificial Intelligence in Mobile Systems (AIMS) 2002*, Lyon, Frankreich, Juli 2002.

[Joy03] JOYCE, JAMES: *Bayes' Theorem*. In: ZALTA, EDWARD N. (Herausgeber): *The Stanford Encyclopedia of Philosophy*. September 2003. plato.stanford.edu/archives/win2003/entries/bayes-theorem/.

[jrm03] *Java RMI Specifications*. Spezifikation, Sun Microsystems Inc., 2003. java.sun.com/j2se/1.4.2/docs/guide/rmi/spec/rmiTOC.html.

[JS03] JUDD, GLENN und PETER STEENKISTE: *Providing Contextual Information to Pervasive Computing Applications*. In: *Proceedings of the First IEEE International Conference on Pervasive Computing (PerCom'03)*, Dallas, USA, März 2003. IEEE.

[jsu03] *Jini Technology Surrogate Architecture Specification, v1.0*. JDP-Standard, Sun Microsystems Inc., Juni 2003.

[Kan05] KANT, IMMANUEL: *Akademieausgabe: Kant's gesammelte Schriften*, Band 20, Kapitel Preisschrift über die Fortschritte der Metaphysik, Seiten 253–332. Universität Bonn, 2005. Online Vorversion.

[KBM+02] KINDBERG, TIM, JOHN BARTON, JEFF MORGAN, GENE BECKER, DEBBIE CASWELL, PHILIPPE DEBATY, GITA GOPAL, MARCOS FRID, VENKY KRISHNAN, HOWARD MORRIS, JOHN SCHETTINA, BILL SERRA und MIRJANA SPASOJEVIC: *People, Places, Things: Web Presence for the Real World*. ACM Mobile Networks and Applications (MONET), 7(5):3665–376, Oktober 2002.

[KBW99] KUNZMANN, PETER, FRANZ-PETER BURKARD und FRANZ WIEDMANN: *dtv-Atlas Philosophie*. dtv, Würzburg, 1999.

[KdBH+03] KAWAMURA, TAKAHIRO, JAQUES-ALBERT DE BLASIO, TETSUO HASEGAWA, MASSIMO PAOLUCCI und KATIA SYCARA: *Preliminary Report of Public Experiment of Semantic Service Matchmaker with UDDI Business Registry*. In: *First International Conference on Service-Oriented Computing (ISOC2003)*, Nummer 2910 in *Lecture Notes in Computer Science*, Trient, Italien, 2003. Springer.

[KDM+03] KIRBY, GRAHAM, ALAN DEARLE, RON MORRISON, MARK DUNLOP, RICHARD CONNOR und PADDY NIXON: *Active Architecture for Pervasive Contextual Services*. In: *Procedings of the International Workshop on Middleware for Pervasive and Ad-hoc Computing (MPAC 2003)*, Rio de Janeiro, Brasilien, 2003.

[KH05] KRAUSE, MICHAEL und IRIS HOCHSTATTER: *Challenges in Modelling and Using Quality of Context (QoC)*. In: MAGEDANZ, THOMAS (Herausgeber): *Mobility Aware Technologies and Applications (MATA2005)*, Num-

mer 3755 in *Lecture Notes in Computer Science*, Seiten 324–333, Montreal, Kanada, Oktober 2005. Springer Verlag Berlin Heidelberg.

[KLW95] KIFER, MICHAEL, GEORG LAUSEN und JAMES WU: *Logical Foundations of object-oriented and frame-based languages*. Journal of the ACM, 42(4):741–843, 1995.

[Kob93] KOBSA, ALFRED: *Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen*. In: HERZOG, OTTHEIN, THOMAS CHRISTALLER und DIETER SCHÜTT (Herausgeber): *Grundlagen und Anwendungen der Künstlichen Intelligenz – 17. Fachtagung für Künstliche Intelligenz*, Seiten 152–166, Berlin, Deutschland, September 1993. Humboldt-Universität, Springer-Verlag.

[Küp05] KÜPPER, AXEL: *Location-based Services : Fundamentals and Operation*. John Wiley & Sons Ltd., 2005.

[Krö05] KRÖTZSCH, MARKUS: *Regeln und OWL*. Folien zur Vorlesung „Intelligente Systeme im WWW“, Universität Karlsruhe, 2005.

[Kra03] KRAUSE, MICHAEL: *Eine Sprache zur Komposition von Kontextinformationen*. Diplomarbeit, Ludwig-Maximilians-Universität München, August 2003.

[KRWP01] KELLER, RALPH, JEYASHANKHER RAMAMIRTHAM, TILMAN WOLF und BERNHARD PLATTNER: *Active Pipes: Service Composition for Programmable Networks*. In: *Proceedings of Military Communication Conference (Milcom 2001)*, Washington D.C., USA, Oktober 2001. IEEE.

[KS05] KALFOGLOU, YANNIS und MARCO SCHORLEMMER: *Ontology Mapping: The State of the Art*. In: *Semantic Interoperability and Integration*, Nummer 04391 in *Dagstuhl Seminar Proceedings*, Dagstuhl, Deutschland, März 2005. Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl.

[KSLP07] KRAUSE, MICHAEL, MARKUS STRASSBERGER und CLAUDIA LINNHOFF-POPIEN: *Inference of high-level Context using Bayesian Network Templates*. In: *Design and deployment of context-awareness networks, services and applications (AWARE 2007) / Third International Conference on Autonomic and Autonomous Systems (ICAS07)*, Athen, Griechenland, Juni 2007. Vorbereitet zur Einreichung.

[LA01] LAERHOVEN, KRISTOF VAN und KOFI AIDOO: *Teaching Context to Applications*. Personal and Ubiquitous Computing, 5(1):46–49, Januar 2001.

[Lau91] LAUBSCH, JOACHIM: *Wissensrepräsentation*, Kapitel Einführung: Zum Gegenstand einer Theorie der Wissensdarstellung, Seiten 9–17. Oldenbourg Verlag, München, 1991.

[LC04] LÄMMEL, UWE und JÜRGEN CLEVE: *Künstliche Intelligenz*. fachbuchverlag Leipzig, 2 Auflage, 2004.

[LJS+02] LUKOWICZ, PAUL, H. JUNKER, M. STÄGER, T. VON BÜREN und G. TRÖSTER: *WearNET: A Distributed Multi-sensor System for Context Aware Wearables*. In: *Proceedings of the 4th international conference on Ubiquitous Computing (UbiComp02)*, Seiten 361–370, Göteborg, Sweden, 2002.

[LL03] LEE, CHOONHWA und SUMI LEHAL: *Context Attributes: An Approach to Enable Context-awareness for Service Discovery*. In: *Proceedings of the Symposium on Applications and the Internet (SAINT) 2003*, Seiten 22–30, Orlando, Florida, USA, 2003.

[Lug01] LUGER, GEORGE F.: *Künstliche Intelligenz – Strategien zur Lösung komplexer Probleme*. Pearson Studium. Addison Wesley, 4 Auflage, 2001.

[Mar02a] MARCHAL, BENOÎT: *Java Web Services Unleashed*, Kapitel Understanding SOAP, Seiten 141–168. Sams Publishing, Indianapolis, USA, April 2002.

[Mar02b] MARCHAL, BENOÎT: *Java Web Services Unleashed*, Kapitel SOAP Basics, Seiten 169–213. Sams Publishing, Indianapolis, USA, April 2002.

[Mat97] MATES, BENSON: *Elementare Logik - Prädikatenlogik der ersten Stufe*. Vandenhoeck & Ruprecht, Göttingen, 1997.

[Mat05] MATHES, JOHANNES: *Vermittlungs von Kontexterbringerdiensten*. Diplomarbeit, Ludwig-Maximilians-Universität München, München, Oktober 2005.

[May04] MAYRHOFER, RENE MICHAEL: *An Architecture for Context Prediction*. Doktorarbeit, Johannes Kepler Universität, Linz, Österreich, Oktober 2004.

[MB05] MEI, JING und ELENA PASLARU BONTAS: *Reasoning Paradigms for SWRL-enabled Ontologies*. In: *Proceedings of Workshop Protégé with Rules*, Madrid, Spanien, Juli 2005.

[MBH+04] MARTIN, DAVID, MARK BURSTEIN, JERRY HOBBS, ORA LASSILA, DREW MCDERMOTT, SHEILA MCILRAITH, SRINI NARAYANAN, MASSIMO PAOLUCCIA, BIJAN PARSIA, TERRY PAYNE, EVREN SIRIN, NAVEEN SRINIVASAN und KATIA SYCARA: *OWL-S: Semantic Markup for*

Web Services. www.daml.org/services/owl-s/1.1/overview/, Juni 2004. OWL-S Coalition White Paper.

[MBL+04] MARTIN, DAVID, MARK BURSTEIN, ORA LASSILA, MASSIMO PAOLUCCI, TERRY PAYNE und SHEILA MCILRAITH: *Describing Web Services using OWL-S and WSDL*. www.daml.org/services/owl-s/1.1/owl-s-wsdl.html, 2004. OWL-S Coalition Bericht.

[MDA+95] MOZER, MICHAEL C., ROBERT H. DODIER, MARC ANDERSON, LUCKY VIDMAR, ROBERT F. CRUICKSHANK III und DEBRA MILLER: *Current trends in connectionism*, Kapitel The Neural Network House - An Overview, Seiten 371–380. Erlbaum, Hilldale, NJ, USA, 1995.

[Meß01] MESSMER, JENS: *Modellierung der Augmented World in Nexus*. Diplomarbeit, Universität Stuttgart, Januar 2001.

[Mer02] MERSCH, DIETER: *Einführung in die Wissenschaftstheorie und Wissenschaftsforschung*, Band 4, Kapitel Semiotik und Grundlagen der Wissenschaft, Seiten 323–338. Schneider Verlag, Hohengehren, 2002.

[Möl05] MÖLLER, PETER: *philolex*. Web-Lexikon der Philosophie, 2005.

[MM03] MCILRAITH, SHEILA A. und DAVID L. MARTIN: *Bringing Semantics to Web Services*. IEEE Intelligent Systems, 18(1):90–93, 2003.

[Män03] MÄNTYJÄRVI, JANI: *Sensor-based context recognition for mobile applications*. Doktorarbeit, Universität Oulu, Oulu, Finnland, Dezember 2003.

[Moa97] MOATS, RYAN: *URN Syntax*. Vorschlag zum IP-Standard RFC 2141, IETF Network Working Group, Mai 1997.

[Moz98] MOZER, MICHAEL C.: *The Neural Network House: An Environment that Adapts to its Inhabitants*. In: COEN, M. (Herausgeber): *Proceedings of the American Association for Artificial Intelligence Spring Symposium on Intelligent Environments*, Seiten 110–114, Menlo Park, CA, USA, 1998. AAAI, AAAI Press.

[MPM+04] MARTIN, DAVID, MASSIMO PAOLUCCI, SHEILA MCILRAITH, MARK BURSTEIN, DREW MCDERMOTT, DEBORAH MCGUINNESS, BIJAN PARSIA, TERRY PAYNE, MARTA SABOU, MONIKA SOLANKI, NAVEEN SRINIVASAN und KATIA SYCARA: *Bringing Semantics to Web Services*. In: *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*, San Diega, Kalifornien, USA, Juli 2004.

[MSB04] MILLER, ERIC, RALPH SWICK und DAN BRICKLEY: *Resource Description Framework (RDF)*. Technischer Bericht, The W3 Consortium, November 2004.

[MSS04] MOTIK, BORIS, ULRIKE SATTLER und RUDI STUDER: *Query Answering for OWL-DL with Rules*. In: *Proceedings of the 3rd International Semantic Web Conference (ISWC 2004)*, Seiten 549–563, Hiroshima, Japan, November 2004.

[New05] NEWMARCH, JAN: *UPnP Services and Jini Clients*. In: *Proceedings of Information Systems: Next Generations (ISNG 2005)*, Las Vegas, USA, April 2005.

[OAS06] OASIS OPEN: *OASIS Web Services Notification (WSN) TC*, Juli 2006. www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn.

[OL03] O'SULLIVAN, DECLAN und DAVID LEWIS: *Semantically Driven Service Interoperability for Pervasive Computing*. In: *Third International ACM Workshop on Data Engineering for Wireless and Mobile Access*, San Diego, USA, September 2003.

[OR23] OGDEN, CHARLES K. und A. RICHARDS: *The Influence of Language upon Thought and of The Science of Symbolism*, Kapitel The Meaning of Meaning, Seiten 9–12. London, UK, 10 Auflage, 1923.

[owl04] *OWL Web Ontology Language*. W3C Recommendation, The Web Ontology Working Group – The W3 Consortium, Februar 2004.

[PA04] POOL, MIKE und JEFFREY AIKIN: *KEEPER and Protégé: An Elicitation Envirionment for Bayesian Inference Tools*. In: *Proceedings of Second International Conference on Pervasive Computing (Pervasive 2004)*, Wien, Österreich, Juni 2004.

[PCB00] PRIYANTHA, NISSANKA BODHI, ANIT CHAKRABORTY und HARI BALAKRISHNAN: *The Cricket Location-Support System*. In: *6th ACM MOBICOM*, Boston, USA, August 2000.

[PFCA05] POOL, MIKE, FRANCIS FUNG, STEPHEN CANNNON und JEFFREY AIKIN: *Is it Worth a Hoot? Qualms about OWL for Uncertainty Reasoning*. In: *Proceedings of Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2005)*, Galway, Ireland, November 2005.

[PG03] PAPAZOGLOU, MIKE P. und DIMITRIOS GEORGAKOPOULOS: *Service Oriented Computing*. Communications of the ACM, 46(10):24–28, Oktober 2003.

[PG04] PEPPER, STEVE und LARS MARIUS GARSHOL: *Analysis of TMRM Use Cases*. Technischer Bericht, ISO/IEC JTC 1/SC34 – Document Description and Processing Languages, 2004.

[PKM94] POPIEN, CLAUDIA, AXEL KÜPPER und BERND MEYER: *A Formal Description of Open Distributed Processing (ODP) Trading Based on Guidelines for the Definition of Managed Objects (GDMO)*. Journal of Network and Systems Management, 2(4):383–400, Januar 1994.

[PKPS02] PAOLUCCI, MASSIMO, TAKAHIRO KAWAMURA, TERRY R. PAYNE und KATIA SYCARA: *Semantic Matching of Web Services Capabilities*. In: *Proceedings of the First International Semantic Web Conference (ISWC2002)*, Sardinien, Italien, Juni 2002.

[PKW03] POKRAEV, STANISLAV, JOHAN KOOLWAAIJ und MARTIN WIBBELS: *Extending UDDI with Context-Aware Features Based on Semantic Service Descriptions*. In: *In Proceedings of the First International Conference on Web Services (ICWS2003)*, Las Vegas, USA, Juni 2003.

[PM] PARYS, DARIUSZ und JÜRGEN MAUERER: *Web Services Standards: SOAP, UDDI und WSDL*. Technischer Bericht, Microsoft Inc.

[Pos06] POSTGRESQL GLOBAL DEVELOPMENT GROUP: *PostgreSQL Documentation*, 2006. www.postgresql.org/docs.

[PSTH05] PAN, JEFF Z., GIORGOS B. STAMOU, VASSILIS TZOUVARAS und IAN HORROCKS: *f-SWRL: A Fuzzy Extension of SWRL*. In: *International Conference on Artificial Neural Networks (ICANN 2005)*, Seiten 829–834, Warschau, Polen, September 2005. European Neural Network Society.

[RD01] ROWSTRON, ANTONY und PETER DRUSCHEL: *Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer-Systems*. In: *Proceedings of the 2001 International Middleware Conference*, Seiten 329–350, Heidelberg, Deutschland, November 2001. IFIP/ACM.

[Rei00] REIF, GERALD: *Moderne Aspekte der Wissensverarbeitung*. Diplomarbeit, Technische Universität Graz, Graz, Österreich, Januar 2000.

[RKY+02] RATNASAMY, SYLVIA, BRAD KARP, LI YIN, FANG YU, DEBORAH ESTRIN, RAMESH GOVINDAN und SCOTT SHENKER: *GHT: a geographic hash table for data-centric storage*. In: *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA 2002)*, Seiten 78–87, Atlanta, Georgia, USA, September 2002.

[Rus04] RUSSELL, BERTRAND: *Philosophie des Abendlandes*, Band 4208 der Reihe *Serie Piper*. Piper, München, Dezember 2004.

[Rya99] RYAN, NICK: *ConteXtML: Exchanging Contextual Information between a Mobile Client and the FieldNote Server*. Webdokument, University of Kent at Canterbury, UK, August 1999. www.cs.kent.ac.uk/projects/mobicomp/fnc/ConteXtML.html.

[Sam02] SAMULOWITZ, MICHAEL: *Kontextadaptive Dienstnutzung in Ubiquitous Computing Umgebungen*. Doktorarbeit, Ludwig-Maximilians-Universität, München, Juni 2002.

[Sat90] SATYANARAYANAN, MAHADEV: *Scalable, Secure, and Highly Available Distributed File Access*. IEEE Computer, 23(5):9–18,20–21, Mai 1990.

[SAT+99] SCHMIDT, ALBRECHT, KOFI ASANTE AIDO, ANTTI TAKALUOMA, URPO TUOMELA, KRISTOF VAN LAERHOVEN und WALTER VAN DE VELDE: *Advanced Interaction in Context*. In: *First International Symposium on Handheld and Ubiquitous Computing (HUC99)*, Nummer 1707 in *Lecture Notes on Computer Science*, Seiten 89–101, Karlsruhe, September 1999. Springer Verlag Berlin Heidelberg.

[SBG99] SCHMIDT, ALBRECHT, MICHAEL BEIGL und HANS-W. GELLERSEN: *There is more to context than location*. Computers and Graphics, 23(6):893–901, 1999.

[Sch95] SCHÖNING, UWE: *Logik für Informatiker*. Reihe Informatik. Spektrum Akademischer Verlag, Heidelberg, 4 Auflage, 1995.

[Sch02] SCHMIDT, ALBRECHT: *Ubiquitous Computing – Computing in Context*. Doktorarbeit, Lancaster University, England, November 2002.

[Sch04] SCHMUDE, ANTHONY NORMAN: *Ontologiebasierte Suche und Navigation in webbasierten Informationssystemen*. Diplomarbeit, Universität Hamburg, Deutschland, Februar 2004.

[SLL05] SALVADOR, ZIGOR, ALBERTO LAFUENTE und MIKEL LARREA: *Jini as a platform for ubiquitous computing*. In: *Proceedings of the Simposio sobre Computación Ubicua e Inteligencia Ambiental (UCAmI 2005)*, Seiten 251–257, Granada, Spanien, September 2005.

[SLP04] STRANG, THOMAS und CLAUDIA LINNHOFF-POPIEN: *A Context Modeling Survey*. In: *Proceedings of First International Workshop on Advanced Context Modelling, Reasoning and Management*, Nottingham, England, September 2004.

[SLPF03a] STRANG, THOMAS, CLAUDIA LINNHOFF-POPIEN und KORBINIAN FRANK: *CoOL: A Context Ontology Language to enable Contextual Interoperability*. In: STEFANI, J.-B., I. DEMEURE und D. HAGIMONT (Herausgeber): *4th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS)*, Nummer 2893 in *Lecture Notes in Computer Science*, Seiten 236–247. International Federation for Information Processing (IFIP), Springer, 2003.

[SLPF03b] STRANG, THOMAS, CLAUDIA LINNHOFF-POPIEN und KORBINIAN FRANK: *Proceedings of International Conference on Software, Telecommunications and Computer Networks (SoftCom2003)*. In: BEGUSIC, DINKO und NIKOLA ROZIC (Herausgeber): *Applications of a Context Ontology Language*, Seiten 14–18. Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, University of Split, Croatia, Oktober 2003.

[SPTH05] STAMOU, GIORGOS, JEFF Z. PAN, VASSILIS TZOUVARAS und IAN HORROCKS: *A Fuzzy Extension to SWRL*. In: *Proceedings of W3C Workshop on Rule Languages for Interoperability*, Washington D.C., USA, April 2005.

[ST94] SCHILIT, BILL und MARVIN THEIMER: *Disseminating Active Map Information to Mobile Hosts*. IEEE Network, 8(5):22–32, 1994.

[Str91] STRUSS, PETER: *Wissensrepräsentation*, Kapitel Wissensrepräsentation – ein zentrales Problem der Künstlichen Intelligenz, Seiten 1–8. Oldenbourg Verlag, München, 1991.

[Str03] STRANG, THOMAS: *Service-Interoperabilität in Ubiquitous Computing Umgebungen*. Doktorarbeit, Ludwig-Maximilians-Universität München, 2003.

[SuKPS04] SRINIVASAN, NAVEEN und MASSIMO PAOLUCCI UND KATIA P. SYCARA: *An Efficient Algorithm for OWL-S Based Semantic Search in UDDI*. In: *Semantic Web Services and Web Process Composition (SWSWPC2004)*, Nummer 3387 in *Lecture Notes in Computer Science*, Seiten 96–110, San Diego, USA, Juli 2004.

[Sun06a] SUN MICROSYSTEMS, INC.: *Java Message Service (JMS)*, 2006. java.sun.com/products/jms/.

[Sun06b] SUN MICROSYSTEMS, INC.: *JavaTM Platform, Enterprise Edition 5 API Specifications*, 2006. java.sun.com/javaee/5/docs/api/.

[SW01] SMITH, BARRY und CHRISTOPHER WELTY: *Ontology: Towards a new Synthesis*. In: *Proceedings of the international conference on Formal ontology in Information Systems (FOIS)*, Ogunquit, Maine, USA, Oktober 2001. ACM Press.

[Tan00] TANENBAUM, ANDREW S.: *Computernetzwerke*. 3. Addison-Wesley, November 2000.

[Tay03] TAYLOR, JOHN G.: *Attention Control and the Disappearing Computer*. Technischer Bericht, King's College, Strand, London, UK, 2003.

[uCB02] CHRISTOPH BUSSLER, DIETER FENSEL UND: *The Web Service Modeling Framework WSMF*. Electronic Commerce Research and Applications, 1(2):113–137, 2002.

[udd04] *Introduction to UDDI: Important Features and Functional Concepts*. Technischer Bericht, Organization for the Advancement of Structured Information Standards, Oktober 2004.

[UG96] USCHOLD, MIKE und MICHAEL GRUNINGER: *Ontologies: Principles, Methods and Applications*. Knowledge Engineering Review, 11(2), Juni 1996.

[UmK95] USCHOLD, MIKE und MARTIN KING: *Towards a Methodology for Building Ontologies*. In: *Proceedings of the workshop on Basic Ontological Issues in Knowledge Sharing*, Juli 1995.

[UPIG01] URI PLANNING INTEREST GROUP, W3C/IETF: *URIs, URLs, and URNs: Clarifications and Recommendations 1.0*. W3C Note, The W3 Consortium, September 2001.

[upn00a] *Understanding Universal Plug and Play*. White Paper, Microsoft Co., Juni 2000.

[upn00b] *Universal Plug and Play Device Architecture Version 1.0*. Standard, UPnP Forum, Juni 2000.

[Vol01] VOLZ, RAPHAEL: *Eine kleine Einführung in Ontologien*. Beitrag zum Workshop Begriffliche Formalisierung von Prozessen und Systemen, November 2001.

[Wei91] WEISER, MARK: *The Computer for the Twenty-First Century*. Scientific American, 265(3):94–104, September 91.

[WG02] WEBER, JOW und DARREN GOVONI: *Java Web Services Unleashed*, Kapitel What are Web Services?, Seiten 7–20. Sams Publishing, Indianapolis, USA, April 2002.

[WGZP04] WANG, XIAO HANG, TAO GU, DA QING ZHANG und HUNG KENG PUNG: *Ontology Based Context Modeling and Reasoning using OWL*. In: *Proceedings of Workshop on Context Modeling and Reaonign (CoMoRea 2004)*, Orlando, Florida, USA, März 2004.

[WHFG92] WANT, ROY, ANDY HOPPER, VERONICA FALCAO und JONATHAN GIBBONS: *The Active Badge Location System*. ACM Transactions on Information Systems (TOIS), 10(1):91–102, Januar 1992.

[Wil05] WILDER, THOMAS: *Implementierung eines kontextsensitiven Beispieldienstes für Mobilfunknetze*. Fortgeschrittenenpraktikum, Institut für Informatik, Ludwig-Maximilians-Universität München, 2005.

[Win01] WINOGRAD, TERRY: *Architectures for Context*. Human-Computer Interaction (HCI) Journal, 16(2):401–419, 2001.

[WPT03] WANT, ROY, TREVOR PERING und DAVID TENNENHOUSE: *Comparing autonomic and proactive computing*. IBM Systems Journal, 42(1):129–135, 2003.

[WSA+95] WANT, ROY, BILL N. SCHILIT, NORMAN I. ADAMS, RICH GOLD, KARIN PETERSEN, DAVID GOLDBERG, JOHN R. ELLIS und MARK WEISER: *An Overview of the ParcTab Ubiquitous Computing Experiment*. IEEE Personal Communications, 2(6):28–43, Dezember 1995.

[wsm04] *Mission Statement – Web Service Modeling Ontology Project*. DERI Working Draft, WSML working group, Januar 2004.

[Wut02a] WUTKA, MARK: *Java Web Services Unleashed*, Kapitel UDDI, Seiten 215–225. Sams Publishing, Indianapolis, USA, April 2002.

[Wut02b] WUTKA, MARK: *Java Web Services Unleashed*, Kapitel UDDI in Depth, Seiten 227–250. Sams Publishing, Indianapolis, USA, April 2002.

[WVV01] WACHE, HOLGER, THOMAS VÖGELE und UBBO VISSER: *Ontology-based Integration of Informations - A Survey of Existing Approaches*. In: STUCKENSCHMIDT, HEINER (Herausgeber): *IJCAI-01 Workshop: Ontologies and Informations Sharing*, Seiten 108–117, Seattle, USA, August 2001.

[Xer99] XEROX PALO ALTO RESEARCH CENTER (PARC): *Mark D. Weiser: July 23, 1952 - April 27, 1999*. www2.parc.com/csl/members/weiser/mwbkgd.htm, 1999.

[xqu05] *XQuery 1.0: An XML Query Language*. W3C Candidate Recommendation, World Wide Web Consortium, November 2005.

[YCB+04] YERGEAU, FRANÇOIS, JOHN COWAN, TIM BRAY, JEAN PAOLI, C. M. SPERBERG-MCQUEEN und EVE MALER: *Extensible Markup Language (XML) 1.1*. W3C Recomendation, The W3 Consortium, April 2004.

[Zim93] ZIMMERMANN, HANS-JÜRGEN: *Prinzipien der Fuzzy Logik*. Spektrum der Wissenschaft, 3, März 1993.