Effiziente Echtzeit-Kommunikationsdienste durch Einbeziehung von Kontexten

Vom Fachbereich 18
Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt
zur Erlangung des Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
genehmigte

Dissertationsschrift

von

Dipl.-Ing. Manuel Görtz

geboren am 19.03.1973 in Frankfurt am Main

Erstreferent:	Prof. Dr.-Ing. Ralf Steinmetz
Korreferent:	Univ.-Prof. Dr.-Ing. Jörg Eberspächer

Tag der Einreichung:	12.05.2005
Tag der Disputation:	11.07.2005

D 17
Darmstädter Dissertationen

Vorwort

Der Weg zum Ziel beginnt an dem Tag, an dem du die hundertprozentige Verantwortung für dein Tun übernimmst.

Dante Alighieri

Die vorliegende Dissertation entstand während meiner Arbeit als wissenschaftlicher Mitarbeiter am Lehrstuhl *Multimedia Kommunikation – KOM* an der Technischen Universität Darmstadt.

Während der Prozess des Schreibens ein oftmals sehr einsamer ist, haben doch eine ganze Reihe Menschen auf unterschiedliche Weise zum Gelingen der vorliegenden Arbeit beigetragen. Denen, die mich auf dem Weg ganz oder teilweise begleitet haben, möchte ich hiermit danken.

Meinem Doktorvater Prof. Dr.-Ing. Ralf Steinmetz gilt ein ganz besonderer Dank für sein Vertrauen und die Möglichkeit, meine Promotion an seinem Lehrstuhl durchzuführen. Er hat bei KOM ein sehr spannendes, kreatives und hilfsbereites Umfeld geschaffen. Durch seine Diskussion, seine konstruktiven Anmerkungen und Unterstützung hat er zum erfolgreichen Abschluss der Arbeit beigetragen.

Für die Übernahme des Korreferats möchte ich Prof. Dr.-Ing. Jörg Eberspächer vom Lehrstuhl für Kommunikationsnetze der Technischen Universität München herzlich danken.

Ein solch komplexes System aus vielen Soft- und Hardwarekomponenten umzusetzen, bedarf einer Menge fleißiger Hände und vor allem pfiffiger Köpfe. All meinen Studenten, die in ihrer Studien- /Diplomarbeit oder als HiWi das Gesamtsystem durch ihre eigenen Beiträgen mit Leben gefüllt haben, möchte ich danken. Insbesondere mein Studienkollege und Freund Marc Weissmann, die Spanische Armada und Johannes Schmitt, mein jetziger Zimmerkollege, seien hier besonders genannt.

Ebenfalls zu Dank verpflichtet bin ich den Korrekturlesern dieser Dissertation, die mit ihren wertvollen Hinweisen und Kommentaren zur Verbesserung der Lese-Qualität beigetragen haben. Diese sind namentlich: ɐsɐɹbɯAAndreas[1], Johannes, Marc, Peter,

[1] Idee von Vasilis „geklaut“

Stephan, Michael, Utz. Ein spezieller Dank geht an rac & matthias[r1] für ihre jederzeit kritischen Anmerkungen – zu Allem.

Besonderen Dank an Karin und Harald, die sich durch viele, viele Seiten gequält haben und unzählige Schreibfehler gefunden haben, ohne das Geschriebe zu „verstehen“.

Dank auch an meine beiden Mitstreiter Vasilis und Stefan, meinem langjährigen Freund, für die Diskussion und moralische Unterstützung während des „Nachvornepromovierens“. Auch all den hier ungenannten KOM'lern, mit denen ich mich während meiner Zeit am Lehrstuhl austauschen konnte, sei gedankt. Ich habe dabei viel gelernt.

Mein herzlicher Dank geht an meine Eltern – Evelyn und Harald – ohne die ich nicht da wäre, wo ich heute bin. Sie, wie auch meine Schwester Claudia, waren und sind immer ein Rückhalt. Danke für ihr „Du wirst es schon schaffen“-Vertrauen in mich.

Von ganzem Herzen möchte ich meiner Frau und Liebe Karin danken, die einen nicht zubenennenden Teil zum Gelingen dieser Dissertation beigesteuert hat. Insbesondere, da ich einen Großteil der Arbeit mit ins Privatleben genommen habe und meinen Laptop ihr zeitlich vorgezogen habe. Sie hat mich bewusst und oftmals unbewusst geerdet und mir die Kraft gegeben, die notwendig ist, den langen Weg gehen zu können. Dafür schulde ich dir ewig Dank.

Dank auch an die Einstürzenden Neubauten, Pearl Jam, Nirvana, Nick Cave, Bush und all den anderen musikalischen Begleitern in den Stunden des Schreibens!

Dreieich, 12. Mai 2005, 2:00

Für Karin
me ke kipona aloha

Kurzfassung

–·– ··– ·–· ––·· ··–· ·– ··· ··· ··– –· ––·

Samuel Morse

Kommunikation beschäftigt seit jeher den Menschen. Sie ermöglicht ein soziales Zusammenleben, den Austausch von Informationen und Nachrichten sowie die Bewahrung von Wissen. Mit dem Bedürfnis nach Kommunikation ging seit jeher auch die Entwicklung von Technologien zur Unterstützung der Kommunikation einher.

Heutzutage steht eine fast nicht zu überschauende Auswahl an technischen Kommunikationsgeräten und Kommunikationsdiensten zur Verfügung. So können Menschen von (fast) jedem Punkt der Erde in *Echtzeit* miteinander kommunizieren. Kommunikation ist zu einem essentiellen aber auch sehr komplexen Dienst geworden.

Der Bedarf an Kommunikation steigt seit Jahren stetig an und eine schnelle *Erreichbarkeit* von Gesprächspartnern sowie *effiziente Handhabung* der Kommunikationsbeziehungen wird immer wichtiger. Jedoch ist in den letzten Jahren eine Tendenz zur Technikzentrierung insbesondere der Endgeräte zu beobachten, die teilweise zu Geräten mit einer Funktionsüberladung führt.

In dieser Arbeit wird das Paradigma der *nutzerzentrierten Kommunikationsdienste* für eine *nahtlose Echtzeitkommunikation* verfolgt. Bei diesem stehen der Nutzen und die Bedürfnisse des Nutzers im Vordergrund. Die Bedienung und die wahrnehmbare Technik treten in den Hintergrund, damit Kommunikation trotz räumlicher Entfernung so einfach und effektiv wie eine vis-à-vis Kommunikation wird. Um dies zu erreichen, wurden aus der Beobachtung und Analyse menschlicher Kommunikation geeignete Konzepte für einen Transfer in eine technische Entsprechung abgeleitet. Die Einbeziehung des *Kontexts*, in dem die Kommunikation stattfindet, wurde als Schlüsselkonzept für eine effiziente Kommunikation identifiziert.

Es wurden drei prinzipielle Verfahren entwickelt, die zu einer Verbesserung der Kommunikationshandhabung und so aus Sicht des Nutzers zu einer Steigerung der Effizienz führt. Für diese wird in der Arbeit ein eigenes Maß definiert. Die *Kontext-bewusste Filterung* erlaubt es dem Nutzer zu spezifizieren, wie in einem bestimmten Kontext eingehende Kommunikationsanfragen behandelt werden sollen. Durch die *Kontext-Verteilung* wird dem Anrufer der Kontext des Angerufen mitgeteilt, so dass er auf Basis des eigenen und des übermittelten Kontexts entscheiden kann, ob und wie der Kommunikationsaufbau fortgesetzt werden soll. Das dritte Verfahren setzt *Kommunikationsbroker*

ein, die stellvertretend für die Nutzer mit den Brokern potenzieller Gesprächspartner Zeitspannen für eine für beide Seiten günstige Kommunikation aushandeln.

Die Umsetzung der zuvor identifizierten Konzepte und Verfahren wurde in Form von *Kontext-bewussten Echtzeitkommunikationsdiensten*, einer neuartigen Ausprägung von Diensten, erreicht. Als exemplarische Kommunikationsplattform wurde mit der IP-Telefonie ein spezieller aber sehr anspruchsvoller interaktiver Echtzeitdienst ausgewählt. IP-Telefonie repräsentiert einen Telefoniedienst, der sich zu einer etablierten Ergänzung zum bestehenden konventionellen Telefonsystem entwickelt hat und dem das Potenzial eingeräumt wird, eine teilweise Ablösung der leitungsvermittelten Telefonie insbesondere im Geschäftsumfeld zu erreichen.

Eine zentrale Annahme der Arbeit ist, dass eine Unterstützung in Form einer einfach zu erlernenden Methodik bei der Diensterstellung und -bereitstellung zu individuellen und effektiven nutzerzentrierten Diensten führt. In der vorliegenden Arbeit wird ein umfassendes *Framework* für die Unterstützung bei der Erstellung und Bereitstellung von Kontext-bewussten Kommunikationsdiensten entworfen. Hierfür wird ein *systemorientierter* Top-Down Ansatz für die Analyse-, Design- und Implementierungsphase gewählt, der alle Schichten von der Nutzerschnittstelle bis zur sensorischen und hardwarenahen Erfassung durchdringt. Im Gegensatz zu anderen Ansätzen auf diesem Gebiet beschränkt sich diese Arbeit nicht auf einzelne unvollständig integrierte Komponenten oder Teilaspekte von Kontext-bewussten Diensten.

Die Analyse von Kontextdefinitionen, ihres Gebrauchs in unterschiedlichen wissenschaftlichen Disziplinen und des Einsatzes in Echtzeitkommunikationsdiensten führt zu einer eigenen Kontextdefinition und -modellierung. Aus dieser wird methodisch ein mehrstufiges *Spiral-Modell* zur Modellierung des Prozesses der automatischen technischen Erfassung, Synthetisierung, Verteilung und Nutzung von Kontexten abgeleitet. In jeder Phase des Spiral-Modells werden alternative Ansätze identifiziert, bewertet und den eigene Vorschläge entgegengesetzt.

Ein effizientes Design sowie eine adäquate Implementierung von Erweiterungen für komplexe Echtzeitkommunikationssysteme sowie ein Kontexterfassungsnetzwerk sind äußerst anspruchsvolle Aufgaben, die ohne Einbeziehung von Software Engineering-Techniken sowie Berücksichtigung der Echtzeitanforderungen der avisierten Dienste nicht zu einer tragfähigen Lösung führen würden. Die Arbeit kategorisiert und bewertet unterschiedliche Architekturen zur Unterstützung von Kontext-bewussten Anwendungen sowie verschiedene Kommunikationsparadigmen.

Für das Gesamtsystem wird eine verteilte dreigeteilte Architektur vorgeschlagen, die aus einem *Kontextaggregationsnetz*, einer Integrationskomponente und den *Kontext-bewussten Diensten* besteht. Ein wesentliches Merkmal dieser Architektur ist die *Trennung* zwischen der Kontextgewinnung und der Kontextnutzung. Für die Verteilung von Kontexten zwischen dem ContextServer und den Kontext-bewussten Diensten wird eine *ereignisbasierte Publish/Subscribe Kommunikationsinfrastruktur* vorgeschlagen und entwickelt. Zusätzlich wird ein Design für einen *Peer-to-Peer-Mechanismus* für den Austausch von Kontexten zwischen erweiterten Kommunikationsendsystemen vorgeschlagen.

Zur Generierung und Administration des Kontextaggregationsnetzes, das Sensordaten der physischen Welt in eine digitale Repräsentation eines Kontexts überführt, wird eine eigene Sprache entwickelt. Die Topologie und Funktionalität der Netzoperatoren können mit dieser festgelegt und parametrisiert werden. Für den Austausch der Daten wird ein geeignetes Datenformat zur Repräsentation der Kontexte und Sensordaten entworfen. Aus der Zielsetzung einer Nutzerzentrierung werden Methoden zur Kontextkomposition systematisch untersucht und mit einem Fuzzy Logik Ansatz ein eigenständiges und tragendes Konzept entworfen. Dies erlaubt eine nahezu natürlichsprachliche Spezifikation, wie der Kontext aus seinen Merkmalen bestimmt werden kann, wodurch nicht nur Experten in der Lage sind, die Vorschriften zu erstellen.

Die vorliegende Arbeit liefert einen wissenschaftlichen Beitrag, der sich aus einem praktischen und einen methodischen Teil zusammensetzt. Der praktische Teil der Arbeit erfüllt den in der Community für Kontext-bewusste und ubiquitäre Anwendungen etablierten Nachweis durch eine prototypische Implementierung. Die aufgezeigten Methodiken und Erfahrungen bei der Umsetzung liefern Referenzmodelle, die sowohl die Methodik als auch Umsetzungserfahrungen für die Konzeptionen und Realisierungen von weiteren ähnlichen Kontext-bewussten Diensten und Anwendungen bereitstellen. Die graphische Möglichkeit zur Erzeugung von Kontext-bewussten Kommunikationdiensten ist ein Alleinstellungsmerkmal, das zusammen mit der entworfenen Infrastruktur die nutzerzentrierte Diensterstellung erlaubt. Aus den theoretischen Erkenntnissen ist eine neue Definition und Modellierung von Kontexten sowie ein prozessorientiertes Spiral-Modell zur Gewinnung, Verteilung und Anwendung von Kontexten entstanden, das nicht auf Kommunikationsdienste beschränkt ist, sondern auf weitere Kontext-bewusste Anwendungen übertragen werden kann.

Abstract

·– –··· ··· – ·–· ·– –·–· –

Samuel Morse

Communication as always been essential to humans. Nowadays people are able to communication in *real-time* at (nearly) every place in the world. However communication has become technically a very complex service. A trend towards techno-centric end-systems can be observed in recent years. This leads often to overloaded devices making communication complicated. A *user-centric communication services* approach is the basis of this thesis. In this thesis user-oriented communication services for seamless real-time communication is the guiding paradigm. It places the benefit and needs of the user at the center.

The users requirements moves into focus. Human communication was observed and analyzed in order to derive technical concepts. The utilization of *context* has been identified as the key concept for efficient communication. Three principle methods have been developed which provide increasing user *efficiency* in handling communication. The provision of an easy to use methodology for creating and running communication services will lead to *user-centric communication services*. These context-aware communication services form an important building block towards the provision of *seamless communication*.

An efficient design and implementation of an extended communication system and a context aggregation network are very challenging tasks. Therefore, software engineering concepts have been applied to develop an comprehensive *framework*. Additionally, the real-time constraints have been obeyed in order to achieve a sustainable solution. An important feature of the software architecture is the *separation* of context acquisition and context usage.

The developed methodologies and experiences for designed efficient real-time communication services provide reference models for realization of similar services. The possibility to create and deploy context-aware communication services together with the framework provides a unique solution for efficient user-centric services. The results of this thesis can be transferred into the domain of general *context-aware services*.

Inhaltsverzeichnis

Kapitel 1

Einführung

Jedem Anfang wohnt
ein Zauber inne.

Stufen
Hermann Hesse

Kommunikation ist seit jeher ein essentielles Grundbedürfnis des Menschens. Sie dient in erster Linie der Übermittlung von Informationen und ist ein integraler Bestandteil des sozialen und kulturellen Lebens. Der Mensch benutzte anfangs Gestik und Laute zur Kommunikation. Die Möglichkeit der Aufzeichnung in Form von Höhlenmalerei und später der Schrift eröffnete eine weitere Art der Nachrichtenübertragung. Trotz der Weiterentwicklung der Kommunikationskanäle haben sich zwei wesentliche Prinzipien nicht geändert. Informationen können auf *synchrone* und *asynchrone* Weise übertragen werden.

Der heutige Arbeitsalltag ist bestimmt von der allgegenwärtigen Notwendigkeit, Informationen auszutauschen. Für den Austausch stehen eine Vielzahl an unterschiedlichen Geräten und Systemen, wie z. B. Telefone, Mobiltelefone, Faxgeräte oder E-Mail zur Verfügung. Die Anzahl der geführten Telefonanrufe und versendeten E-Mails – zwei gewichtige Vertreter von synchroner und asynchroner Kommunikation – nimmt stetig zu[1]. Die Zahl (mobiler) Endgeräte und die Verbreitung breitbandiger Netzanschlüsse wachsen gleichzeitig kontinuierlich an[2].

Die wachsende Mobilität der Nutzer hat zu einer hohen Anzahl an mobilen Kommunikationsgeräten geführt. Jederzeit und an jedem Ort kommunizieren zu können, ist zu einem Credo für viele Menschen geworden. Immer und überall erreichbar zu sein, weckt teilweise die hohe Erwartung, ein Anzurufender möge ebenfalls permanent verfügbar sein. Vielmehr hat der Nutzer jedoch ein legitimes Recht und Bedürfnis, sein Kommunikationsverhalten adäquat kontrollieren und steuern zu können. In Abhängigkeit seines gegenwärtigen Kontexts möchte z. B. der Angerufene bestimmte Kommunikationskanäle für bestimmte Anrufer(gruppen) offen oder geschlossen halten.

Für den Anrufer führt die gestiegene Anzahl an Kommunikationsgeräten und den damit verbundenen Adressen des Anzurufenden nicht zwangsläufig zu einer Verbesserung

[1] 35 Mrd. E-Mails/Tag (Quelle: IDC, 2003)
[2] 1 Mrd. Mobiltelefone (Quelle: GSM-World), 100 Mio. Digital Subscriber Line (DSL)-Anschlüsse (Quelle: IDC, 2003)

der Erreichbarkeit des gewünschten Gesprächspartners. Die Notwendigkeit, eine Reihe von Versuchen durchzuführen und nacheinander Verbindungen zu verschiedenen dem Anrufer bekannten Adressen herzustellen, ist ein häufig wiederkehrendes und zeitraubendes Muster beim Aufbau einer Kommunikation. Klassische Bürozeiten sind nomadischen Arbeitsanforderungen gewichen, was eine Vorhersage des möglichen Kontexts des Anzurufenden erschwert. Die durchaus flexible Erfahrung über den gewöhnlichen Tagesablauf eines Nutzers stellt hierbei, anstelle eines strikten Zeitrasters, einen guten Indikator zur Bestimmung eines geeigneten Startpunktes für die Kommunikation dar.

1.1 Motivation

Gegenwärtige Dienste unterstützen den Angerufenen nicht ausreichend bei der Steuerung der eingehenden Kommunikationsanfragen. Ebenso wird dem Anrufer nicht bei einem schnellen und erfolgreichen Kommunikationsaufbau geholfen. Dies liegt insbesondere darin begründet, dass die vom Anbieter (*provider*) zur Verfügung gestellten Kommunikationsdienste nur minimal zu parametrisieren sind und sich damit nur marginal und nicht ausreichend an die Bedürfnisse der Nutzer anpassen lassen. Die durch technische und gesellschaftliche Entwicklung veränderten Rahmenbedingungen müssen in geeigneter Weise berücksichtigt werden, um den Anforderungen der heutigen Kommunikationsgesellschaft Rechnung tragen zu können. Es entstehen andernfalls durch erfolglose Anrufversuche nicht zu unterschätzende Zeitaufwände und monetäre Kosten.

Eine Untersuchung zur Nutzungshäufigkeit von Mehrwertdiensten sowie die Bewertung von deren Wichtigkeit wurde in [Bri00] durchgeführt. Die Ergebnisse sind in Tabelle 1.1 aufgeführt. Sie belegen, dass die Nutzer den Dienst *Anrufumleiten* als am häufigsten genutzten Dienst und *Automatischer Rückruf* als wichtigsten Dienst ansehen. Der Dienst *Wahlwiederholung* ist ebenfalls in beiden Bewertungskriterien hoch eingestuft. Dies lässt den Schluss zu, dass es Nutzern von Telefonieanwendungen zum einen wichtig ist, eingehende Anrufe umzuleiten und zum anderen Dienste zu nutzen, die zur Unterstützung der Erreichbarkeit von Gesprächspartnern dienen.

1.2 Zielsetzung

Die Bereitstellung von *Kommunikationsdiensten*, die den Nutzer bei seinen täglich anfallenden Kommunikationsanforderungen unterstützten, stellt einen Lösungsansatz für die beschriebenen Probleme dar. Die vorliegende Arbeit hat mehrere Schwerpunkte, die sowohl theoretisch als auch praktisch betrachtet werden, und für die Lösungsvorschläge erarbeitet werden.

Das Hauptziel der vorliegenden Arbeit ist die Steigerung der für den Nutzer wahrnehmbaren Effizienz bei der Steuerung eingehender Kommunikationsanfragen sowie bei dem Aufbau von Kommunikationsbeziehungen. Hierzu wird eine umfassende und systemorientierte Analyse des Problemraums sowie der dafür zu entwickelnden Methodiken

Tabelle 1.1: Untersuchung über die Nutzung und Relevanz von Mehrwertdiensten

Dienstnutzung		**Wichtigkeit**	
Dienst	**Nutzung (in %)**	**Dienst**	**Bewertung (1.0 bis 5.0)**
call forwarding	68	call completion	4.3
re-dial	67	speed dial key	4.1
access to voice mail	53	re-dial	3.9
call completion	36	call forwarding	3.5
hunting group	36	hunting group call pickup	3.2
open listening	34	call transfer	3.1
speed dial key	15	consultation	2.8
conference	14	short dial	2.3
consultation	14	toggle	2.1
call transfer	13	call waiting	2.0
toggle	11	conference	1.9
store number (notepad)	11	date reminder	1.7

eingesetzt. Existierende Ansätze und Verfahren werden betrachtet und es werden für die Lösung relevante Ergebnisse identifiziert. Eigene Beiträge werden erarbeitet, aus denen Referenzmodelle entwickelt werden, die zur Lösung von allgemeineren Problemen genutzt werden können.

Eine weitere wichtige Komponente des systemorientierten Ansatzes ist der Entwurf von konkreten Konzepten aus den entwickelten Modellen zu Teillösungen sowie der Nachweis der Tragfähigkeit des eigenen Lösungsansatzes anhand der Evaluation implementierter Algorithmen, Verfahren und Prototypen.

1.2.1 Effiziente Kommunikation

Das Ziel, das mit dieser Arbeit verfolgt wird, ist die Steigerung der *Effizienz* bei einer *Mensch-Mensch-Kommunikation*. Unter Effizienz wird hierbei das Aufwand-Nutzen-Verhältnis bei dem Aufbau einer Kommunikation verstanden. Dieser Nutzen soll größer sein als der Nutzen, der sich ohne den vorgeschlagenen Lösungsansatz ergibt. Vorteile, die sich aus den Erkenntnissen der Beobachtung direkter, lokaler Kommunikation ergeben, sollen mit geeigneten Mitteln technisch nachgeahmt werden. Die starren Dienstanpassungsmöglichkeiten existierender Systeme sollen aufgebrochen werden und dem Nutzer Methoden zur Verfügung gestellt werden, mit denen Kontext-bewusste Dienste erstellt werden können. Die Methoden sollen aus der Sicht des Providers als sicher für den Betrieb des Netzes angesehen werden können.

Zur Steigerung der Effizienz wird in dieser Arbeit der Ansatz *Kontext-bewusster Kommunikationsdienste* verfolgt. Diese steuern die Kommunikationsbeziehungen insbesondere beim Aufbau einer Kommunikation zwischen den Teilnehmern. Bei der Wahl des

Zeitpunkts und der Kommunikationskanäle soll der gegenwärtige Kontext beider Parteien berücksichtigt werden. Dies führt so zu einer wahrnehmbaren Verbesserung der Kommunikationssteuerung. Dabei sollen die Dienste als „unsichtbare“ Helfer aus der aktiven Wahrnehmung des Nutzers verschwinden. Der Kerngedanke hinter dem Lösungsansatz ist die Metapher eines *digitalen Assistenten für jedermann*. Die Analogie zu einem realen Assistenten für die tägliche Büroarbeit wird bewusst gewählt worden, da ein solcher mit nachweislichem Nutzen die gestellten Anforderungen bei der Steuerung der Kommunikationskanäle erfüllt.

1.3 Vorgehensweise und Methodik

Die drei Wissensbereiche, die zur Erreichung des avisierten Lösungsansatzes identifiziert und analysiert wurden, sind in Abbildung 1.1 dargestellt. Der Fokus der Arbeit liegt in der Schnittmenge der drei Gebiete und ist schraffiert dargestellt. *IP-Telefonie* wurde als repräsentative Anwendung einer interaktiven Echtzeitkommunikation gewählt. Die Eigenschaften und Potenziale dieser Technologie werden analysiert und genutzt.

Während der ersten Hype-Phase der Technologie traten Hindernisse auf, die eine rasche Ausbreitung der IP-Telefonie für den Massenmarkt behinderten. Diese waren insbesondere die unzureichende Berücksichtigung der Firewall-Problematik sowie die Unsicherheit darüber, welches der konkurrierenden Signalisierungsprotokolle (SIP, H.323, H.248) zu bevorzugen sei. Die Problematik, welche Multimedia-Protokolle, wie SIP durch die dynamische Portaushandlung erzeugen, wurde in [Roe02] systematisch analysiert und gelöst. Zur Verbindung von Netzen mit unterschiedlichen Charakteristika (wie Signalisierung über H.323 oder SIP) wurden in [Ack03] Signalisierungsgateways als Integrationskomponente aber auch als Multiplikator herausgearbeitet sowie ein umfassendes Gatewaymodell entwickelt.

Die nutzerspezifische Diensterstellung ist ein weiterer Baustein, welcher die IP-Telefonie als zukunftsträchtige Technologie ausweist. Die vorhandenen Mechanismen und Lösungen werden ausführlich im Themenblock *Dienste* untersucht und für den hier vorgestellten Ansatz geeignet adaptiert. Aus dem Gebiet des *Kontext-bewussten Rechnens* werden existierende Konzepte identifiziert und auf das in diesem Kontext noch nicht systematisch betrachtete Feld der Kommunikationsdienste übertragen.

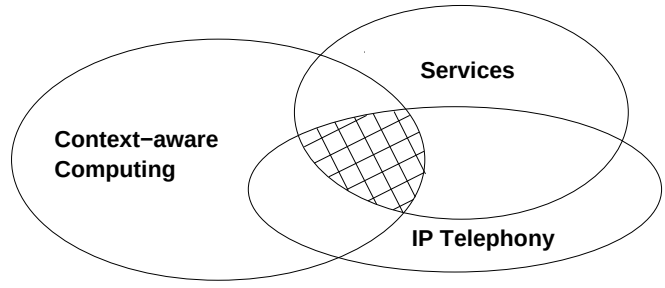


Abbildung 1.1: Fokus der Arbeit

Zur Erreichung der anvisierten Ziele werden in einem ersten Schritt die notwendigen methodischen Vorgehensweisen definiert. Anschließend werden existierende Konzepte und Ansätze aus den drei in Abbildung 1.1 dargestellten Bereichen identifiziert, die für den als Schnittmenge dargestellten Lösungsraum relevant sind. Wo immer möglich, werden diese Konzepte angepasst und integriert, statt neu entwickelt, um möglichst kompatibel zu vorhandenen Lösungen zu bleiben und um eine hohe Wiederverwendung zu erreichen.

Die entwickelten Konzepte und Algorithmen werden prototypisch umgesetzt, was in der Ubiquitous Computing Community als eine der entscheidenden Methodiken zur Erbringung eines Proof-of-Concepts angesehen wird [Dey02]. Die Erfahrungen bei der Umsetzung der Konzepte sowie die Interaktion der Nutzer mit dem erstellten umfassenden Framework zur Realisierung von Kontext-bewussten Diensten werden wichtige Erkenntnisse für die Bewertung des eigenen Lösungsansatzes liefern.

1.4 Wissenschaftliche Ergebnisse der Arbeit

Die vorliegende Arbeit identifiziert und betrachtet systematisch den neuartigen Bereich der Kontext-bewussten Echtzeitkommunikationsdienste. Hierzu wird eine Problemanalyse erstellt und aus einer methodischen Betrachtung wird der eigene Ansatz abgeleitet. Eine eigene Architektur sowie eine Reihe von Algorithmen wurden konzipiert und als Proof-of-Concept umgesetzt. Die Arbeit leistet zahlreiche wichtige Beiträg zur Realisierung des aufgezeigten Lösungsansatzes, die nachfolgend aufgeführt sind:

- Ein durchgängiger systemorientierter Ansatz zur Entwicklung, Realisierung und Bereitstellung von Kontext-bewussten Echtzeitkommunikationsdiensten wird verfolgt. Hierbei wird ein Top-Down-Ansatz gewählt, der die Unterstützung eines Nutzers bei den aufgeführten Schritten als primären Fokus hat.

- Eine methodische Untersuchung, Kategorisierung und Bewertung von Diensten, Kontextanwendungen und Verfahren für die Kontextgewinnungen, die ein einheitliches Verständnis der entworfenen Konzepte erlauben, werden erarbeitet.

- Eine Konzeption einer umfassenden und skalierbaren Architektur zur Bereitstellung von Kontexten der einzelnen Nutzer sowie der Integration dieser in eine SIP-basierte IP-Telefonieinfrastruktur wird entworfen. Die Architektur unterstützt alle Phasen des aufgezeigten Ansatzes zur Gewinnung von Kontexten aus Sensordaten.

- Eine Implementierung aller relevanten Komponenten, die zur Entwicklung, Realisierung und Bereitstellung von Kontext-bewussten Echtzeitkommunikationsdiensten notwendig sind, wird als Proof-of-Concept durchgeführt und bewertet. Die entwickelten Werkzeuge, Bibliotheken und Infrastrukturelemente verfügen über wohl-definierte Schnittstellen, die eine Wiederverwendung und Integration in alternative Systeme erlauben.

1.5 Struktur der Arbeit

Die Arbeit gliedert sich wie folgt:

Kapitel 1 motiviert die vorliegende Arbeit. Der vorgeschlagene Ansatz sowie die verwendeten Methodiken werden erläutert. Die Struktur der Arbeit und die Konventionen werden eingeführt.

Kapitel 2 beinhaltet die für das Verständnis der Arbeit relevanten Definitionen und Grundlagen zu Diensten und Kommunikationsdiensten. Die Konzepte und Architekturen von Telefonie und insbesondere IP-Telefonie werden dargelegt und stellen den Ausgangspunkt für die weitere Betrachtung der Kontext-bewussten Echtzeitkommunikationsdienste dar.

Kapitel 3 beschreibt den zur Problemlösung vorgeschlagenen Ansatz der Kontext-bewussten Kommunikationsdienste zur Effizienzsteigerung. Hierfür wird eine Effizienzdiskussion geführt, aus deren Ergebnis die zu verändernden Parameter abgeleitet werden. Drei prinzipielle Verfahren zur Verbesserung der Kommunikationseffizienz für Nutzer werden vorgestellt.

Kapitel 4 führt eine eigene Kontextdefinition und -modellierung ein. Die gewonnenen Erkenntnisse über die temporale und räumliche Gültigkeit von Kontextmerkmalen spielen für die Repräsentation von Kontexten eine Rolle. Weiterhin wird ein prozessorientiertes Kontext-Spiral-Modell entworfen, welches einen maschinellen Umgang mit Kontexten für die Verwendung für Applikationen und Diensten erlaubt.

Kapitel 5 beinhaltet den Entwurf eines Framework zur systemorientierten Handhabung von Kontexten und Kontext-bewussten Diensten. Es umfasst und unterstützt den Gesamtprozess von der automatischen Erfassung von Kontextmerkmalen über die Verteilung von Kontexten an Applikationen und Diensten, die diese nutzen. Unterschiedliche architektonische Anordnungen und verschiedene Datenverteilmechanismen werden identifiziert und bewertet. Die eigene Architektur und Kommunikationsinfrastruktur wird den untersuchten Alternativen entgegengestellt.

Kapitel 6 präsentiert eine Verfeinerung der Architektur für die Gewinnung von Kontextmerkmalen aus Sensordaten. Ein Kontextaggregationsnetzwerk, welches aus einem Graphen aus Sensordatenquellen und Operatoren, die Funktionen auf die Daten anwenden, besteht, wird vorgestellt. Eine Sprache zur Generierung und Administration der Topologie des Aggregationsnetzes wird entworfen und beschrieben. Zur Aggregation von Kontextmerkmalen zu Kontexten wird die Fuzzy Logik als Inferenzverfahren identifiziert und erläutert. Ein eigenes Format zur Repräsentation der Kontexte und deren Austausch wird dargelegt und beschrieben.

Kapitel 7 zeigt den Entwurf einer Architektur sowie die Schnittstellen einer Integrationskomponente zur persistenten Speicherung und Verteilung von Kontexten. Die Komponente stellt einen Web Service als standardisierte Schnittstelle zu Verfügung. Applikationen können auf die Funktionalitäten der Integrationskomponente

zugreifen und diese nutzen. Weiterhin wird ein In-band Signalierungsmechanismus zur Peer-to-Peer-Verteilung von Kontexten zwischen SIP-Endgeräten beschrieben. Dazu wird das Session Initiation Protocol (SIP) entsprechend erweitert.

Kapitel 8 erweitert den bisher bekannten Diensterstellungsvorgang um die Handhabung der eingeführten neuartigen Konzepte der Einbeziehung von Kontextinformationen in Kommunikationsdiensten. Dazu werden Mechanismen bereitgestellt, die es dem Nutzer erlauben, Dienste graphisch zu erstellen. Ein regelbasiertes Kommunikationskonzept, welches eine nutzerbezogene abstrakte Spezifikation der Kommunikationswünsche erlaubt, wird vorgestellt. Dazu werden Kommunikationsbroker vorgeschlagen, die dem Nutzer den Aufwand der Etablierung einer Kommunikation abnehmen.

Kapitel 9 stellt die Evaluierungen der entworfenen und umgesetzten Komponenten in Form von Messungen sowie von Realisierungsdetails vor. Es wurden aus allen Bereichen des Kontext-Spiral-Modells Verfahren ausgewählt und betrachtet. Ein System zur Erfassung von Lokationsinformationen als Kontextmerkmal sowie ein Fuzzy Logik System zur Aggregation der Kontextmerkmale wird vorgestellt. Details zum internen Aufbau der Integrationskomponente sowie zur Erweiterung von SIP-Entitäten werden erläutert.

Kapitel 10 fasst die wesentlichen Ergebnisse der Arbeit zusammen und identifiziert neue Bereiche, die auf den gewonnenen Erkenntnissen aufbauen.

1.5.1 Konventionen und Notationen in der Arbeit

In der vorliegenden Arbeit werden folgende Aspekte durch Formatierung vom Fließtext hervorgehoben.

- *Kursive Schreibweise* wird verwendet für
  - die allgemeinen Hervorhebung von Begriffen und Sachverhalten
  - Namen von Diensten und Features
- KAPITÄLCHEN werden zur Kennzeichnung von Nachnamen verwendet.
- `Schreibmaschinenschrift` wird zum Setzen von Programmelementen und Befehlen auf der Konsole verwendet.
- Serifenlose Schrift kennzeichnet
  - Nachrichtenmethoden
  - Namen von eigenen Komponenten
  - Aktoren in Szenarien
- *Schräggestellte* Begriffe bezeichnen
  - Kontexte und Kontextmerkmale

- (in Klammern) englische Fachtermini

Eine Liste, der in dieser Arbeit verwendeten Symbole und ihre Bedeutung findet sich auf Seite 317. Arbeiten im Bereich der Kommunikationstechnik kommen nicht ohne eine Vielzahl an Akronymen aus, so auch diese Arbeit. Ein Liste der verwendeten Akronyme und ihrer Bedeutung ist ab Seite 311 aufgeführt.

Kapitel 2

Dienste

Das Pferd frisst keinen Gurkensalat!

1. Telefonat in Deutschland,
24. Oktober 1861
Johann Philipp Reis

Die Einführung identifiziert die drei Bereiche in deren thematischer Schnittmenge sich diese Arbeit hauptsächlich befindet. Die Bereiche Dienste und IP-Telefonie werden in diesem Kapitel behandelt und das Themengebiet Kontext wird ausführlich in Kapitel 4 betrachtet.
Die für die Arbeit grundsätzlichen und relevanten Konzepte, Notationen und Begriffe im Bereich der Kommunikationsdienste werden nachfolgend eingeführt und erläutert. Diese sind für das weitere gemeinsame Verständnis der Ansätze in Kapitel 3 und der nachfolgend entwickelten Methoden, Konzepte und Komponenten wichtig.
Eine prototypische Umsetzung der in dieser Arbeit erarbeiteten Lösungsansätze zum Nachweis der Realisierbarkeit fußen auf einer SIP-basierten IP-Telefonie-Plattform. Im Speziellen werden die Komponenten SIP-User Agent und CPL-Engine um die notwendigen Funktionalitäten für die Erbringung von Kontext-bewussten Kommunikationsdiensten erweitert. Die Konzeption des Session Initiation Protocols (SIP) sowie ihrer Komponenten wird ausführlich dargestellt.

Kurzübersicht

Das Kapitel ist wie folgt strukturiert: Abschnitt 2.1 bildet den Kern dieses Kapitels, in welchem Kommunikationsdienste, ihre Eigenschaften und ihre Modellierung erläutert werden. Aus diesem Dienstmodell wird in Kapitel 3 das eigene erweiterte Dienstmodell abgeleitet. IP-Telefonie wird in Abschnitt 2.2 als konkrete Ausprägung von Kommunikationsdiensten ausgewählt und vorgestellt. Die später entwickelten Lösungen bauen auf diesem Kommunikationsdienst auf. Dazu werden Konzeption, Komponenten und Signalisierungsnachrichten der IP-Telefonie beschrieben und Unterschiede

zu anderen existierenden Telefoniesystemen aufgezeigt. Abschnitt 2.3 beschreibt unterschiedliche Verfahren zur Diensterstellung und Diensterbringung speziell im Umfeld der IP-Telefonie. Die Struktur und die Abhängigkeiten innerhalb des Kapitels sind in Abbildung 2.1 visualisiert.

Abbildung 2.1: Struktur des Kapitels 2

2.1 Kommunikationsdienste

Dienste sind ein wesentlicher funktionaler Bestandteil sowie ein Charakterisierungsmerkmal von Kommunikationssystemen. Dienste erbringen den Teilnehmern die vom System bereitgestellten Funktionen. Der Nutzer kann heute aus einer großen Anzahl vielgestaltiger Dienste für seine Kommunikationswünsche wählen. Zu den dominierenden Diensten zählen Telefonie, E-Mail, Fax und Instant Messaging. Die Ausprägungen der Dienste stellen in zunehmendem Maße ein *Unterscheidungsmerkmal* zwischen den konkurrierenden Anbietern dar.

Trotz der Bedeutung der Kommunikationsdienste existiert keine einheitliche Definition des Dienstbegriffs, keine umfassende Modellierung von Diensten sowie keine standardisierte Methodik zur Erstellung von Diensten. Der Schwerpunkt der Betrachtung von Kommunikationsdiensten wurde in dieser Arbeit auf *Telefonie* gelegt. Als spezielle Ausprägung innerhalb dieser Kommunikationsform wurde die *IP-Telefonie* gewählt. Die Eigenschaften der Signalisierungsprotokolle sowie der eingesetzten Komponenten in der IP-Telefonie werden in Abschnitt 2.2 vertieft.

2.1.1 Definition

Der Begriff *Dienst* (*service*) ist im Umfeld der Informations- und Kommunikationstechnik mit vielen Definitionen überbelegt. Es ist allerdings nicht möglich, eine Definition des Dienstbegriffs zu geben, die alle Anwendungsgebiete einschließt. Daher muss die Definition des Begriffs „Dienst“ für das jeweilige Themenfeld angegeben werden, um Missverständnisse ausschließen zu können. Im Umfeld der Telekommunikation wurde vom Telecommunications Information Networking Architecture (TINA)-Konsortium der Begriff „Dienst“ folgendermaßen definiert [TIN97]:

Definition 2.1 (Dienst)
*Ein **Dienst** σ ist definiert als ein sinnvoller Satz an Fähigkeiten, der durch ein existierendes oder gedachtes System für alle, die es nutzen wollen, bereitgestellt wird.*

Die Definition 2.1 ist sehr generisch gehalten und kann nur bedingt für Dienste im Bereich der Telekommunikation angewandt werden. Eine graphische Repräsentation ist in Abbildung 2.2 gegeben. Das dargestellte Modell sowie die Definition werden im Rahmen dieser Arbeit als Basis für eine weitere formalisierte Darstellung sowie für ein erweitertes Dienstmodell verwendet. Der Dienst wird dabei als „Black-Box“ dargestellt, welche die Funktionalität erfüllt. Der Dienst erhält als Eingabe n Input-Stimuli, wie Signalisierungsnachrichten, Digital Tone Multi Frequency (DTMF)-Signale oder Medienströme. Diese werden vom Dienst verarbeitet und erzeugen m Ausgabe-Ströme.

Abbildung 2.2: für multimediale Echtzeit-Kommunikationsdienste

Das eingeführte Modell erfüllt die Anforderungen an eine Repräsentation für einen *multimedialen Echtzeit-Kommunikationsdienst* (*multi-media real-time communication service*). Ein solcher Dienst ist dadurch gekennzeichnet, dass er mindestens einen Ein- oder Ausgabestrom (*stream*) mit Echtzeitanforderungen (z. B. Sprachübertragung) verarbeiteten kann. Jeder dieser Ströme kann ein Multimedia-Strom sein, der aus der Kombination heterogener statischer (Text, Bild) und dynamischer (Audio, Video) Daten besteht. Eine Definition eines Multimediasystems aus [Ste99] ist in Definition 2.2 gegeben. Der Echtzeitbetrieb eines Rechensystems ist in der DIN Norm 44300 [Deu88] definiert und in Definition 2.3 wiedergegeben.

Definition 2.2 (Multimediasystem)
*Ein **Multimediasystem** ist durch die rechnergesteuerte, integrierte Erzeugung, Manipulation, Darstellung, Speicherung und Kommunikation von unabhängigen Informationen gekennzeichnet, die mindestens in einem kontinuierlichen (zeitabhängigen) und einem diskreten (zeitunabhängigen) Medium kodiert sind.*

Definition 2.3 (Echtzeitbetrieb)
*Der **Echtzeitbetrieb** ist ein Betrieb eines Rechensystems, bei dem Programme zur Verarbeitung anfallender Daten ständig derart betriebsbereit sind, dass die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind.*

2.1.2 Modellierung

Jedes Telefoniesystem muss mindestens den *Basisdienst* (*basic call*), der in dem (*Basic Call State Model – BCSM*) beschrieben ist [YEO96], gewährleisten. Der Basisdienst stellt eine Verbindung zwischen einem festen Ausgangspunkt zu einem Ziel her, das von einem Anrufer durch eine Adresse oder einen anderen Identifikator spezifiziert ist. Nach dem Aufbau der Verbindung wird das Ziel. Über die Verbindung benachrichtigt und kann entscheiden, ob der Anruf entgegengenommen wird oder nicht. Im letzteren

Fall wird dies der auslösenden Seite mitgeteilt. Eine bestehende Verbindung kann von beiden Seiten beendet werden.

Kommunikationsdienste lassen sich konzeptionell in einen *Basisdienst* σ_B, sowie weitere Dienste und *Features* ϕ zerlegen. Die *Komposition* von Diensten ist graphisch in Abbildung 2.3 dargestellt. Formal lässt sich die Komposition wie folgt schreiben [Zav93, Kec00]:

$$\sigma_1 = \sigma_B \oplus \phi_3 \oplus \phi_5 \oplus \phi_6 \tag{2.1}$$

Der Operator $\oplus$ stellt hierbei einen beliebigen, nicht mathematisch definierten Operator zur Verknüpfung von Diensten und Features dar. Die Operation bietet eine qualitative Aussage und keine funktionale Vorschrift zum konkreten Zusammensetzen der einzelnen Dienste und Features. Eine Algebra zur Konkatenation einzelner Features zu einem Dienst in Form des aus der Interprozesskommunikation bekannten Modells des „Pipes-and-Filter“ ist in der Distributed Feature Composition (DFC)-Architektur [JZ98] beschrieben.

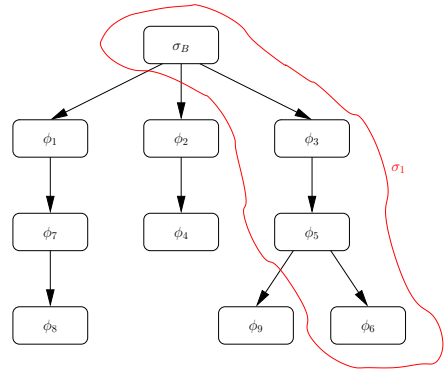


Abbildung 2.3: Komposition von Diensten aus einem Basisdienst und Features

Der Dienst *Anrufweiterleiten mit Rückfrage* (*Call Transfer with Consultational Hold*) lässt sich aus dem Basisdienst *Basisruf* sowie den Diensten *Anrufhalten* und *Anrufweiterschaltung* zusammensetzen. *Anrufumleiten bei Besetzt* (*Call Diversion on Busy*) ist ein Dienst, der aus Diensten und einem Feature komponiert werden kann. Hierbei wird der Dienst *Anrufumleiten* mit dem Feature *Besetzt* zu einem neuen Dienst kombiniert.

Die Trennschärfe zwischen Diensten und Features ist nicht sehr ausgeprägt. Im Rahmen dieser Arbeit wird die nachfolgende Featuredefinition verwendet, die eine Erweiterung der Definition aus [BDC+89] darstellt.

Definition 2.4 (Feature)
*Ein **Feature** ist eine optionale und inkrementelle Funktionalität, welche den Basisdienst erweitert und nicht eigenständig lauffähig ist.*

Jeder Dienst σ und jedes Feature ϕ erfüllen dabei gewisse gegebene Eigenschaften p_i, ausgedrückt durch

$$\sigma_i \models p_i \qquad \text{bzw.} \qquad \phi_i \models p_i$$

Eine *Dienstwechselwirkung* tritt auf, wenn

$$\mathcal{S} \oplus \sigma_i \models p_i \qquad 1 \leq i \leq n \tag{2.2}$$

und

$$\mathcal{S} \oplus \sigma_1 \oplus \sigma_2 \oplus \ldots \oplus \sigma_n \not\models p_1 \wedge p_2 \wedge \ldots \wedge p_n \tag{2.3}$$

erfüllt sind. Hierbei bezeichnet $\mathcal{S}$ das System.

Bei Problemen der Dienstwechselwirkung (*Feature Interaction Problem*) modifiziert oder stört ein hinzugenommenes Feature das Verhalten des bestehenden Gesamtsystems. Dieses Verhalten wird auch als *unerwünschte* Wechselwirkung bezeichnet, um dies von dem durchaus gewünschten Interagieren von Diensten zu unterscheiden. Das Phänomen ist im Umfeld des Public Switched Telephone Network (PSTN) und Intelligent Network (IN) seit langem bekanntes und intensiv erforschtes Problem [KK98, FD02, CMKRM03].

So sind zahlreiche Techniken wie Z [Int92b], Promela [Eti95], die Specification and Description Language (SDL) [Int99c], die Language Of Ordered Temporal Ordering Specification (LOTOS) [Int88] oder Estelle [Int99a] für Erkennung von unerwünschten Dienstwechselwirkungen vorgeschlagen worden. Neuere Ansätze umfassen die Verwendungen von Use Case Maps [Amy01], Online Konfliktauflösungen [RM02] und Hybrid Ansätze [CKM$^+$03]. In einem weit geringerem Maße wurde das Problem bisher in der IP-Telefonie identifiziert und behandelt [ZJ00, ZJ99a, ZJ99b, LS00b, Tur03, Tur02].

Die Konzeption der IP-Telefonie sieht eine Verschiebung der Intelligenz aus dem Netz hin zu der Peripherie des Netzes vor. Darüber hinaus ermöglicht die nun gegebene Offenheit der Provider den Dienstanbietern sowie dem Nutzer eine eigene Diensterstellung. Beide Faktoren haben die Problematik der Vermeidung von unerwünschten Interaktionen noch erschwert. Auch hat sich im Umfeld der IP-Telefonie noch kein ausgeprägtes Bewusstsein für die Problematik der Dienstwechselwirkung entwickelt. Die IP-Telefonie stellt potenzielle Möglichkeiten für den Nutzer zur Erstellung eigener Dienste zur Verfügung. Die Funktionalität so genannter *Custom User Service* (CUS) geht weit über die reine Parametrisierung von Diensten hinaus, wie es im PSTN/IN bekannt ist. Gerade diese Freiheit für den Nutzer kann aber zu unprofessionellen Diensten führen, die den Gesamtbetrieb des Telefonienetzes beeinträchtigen.

Andererseits bietet die ausdrucksstarke Signalisierungssemantik von SIP und H.323 neue Möglichkeiten zum Auflösen von Dienstwechselwirkungen [GAMS03]. Es sollte beachtet werden, dass Dienstwechselwirkungen eine inhärente und bestehende Eigenschaft der Dienstentwicklung, der Dienstbereitstellung und Dienstausführung sind. Konzepte und Methoden zum korrekten Umgang mit dieser Problematik sind daher notwendig.

2.1.3 Diensttypen

Es kann eine Reihe von Dienstklassen unterschieden werden. In Abbildung 2.4 ist ein Ausschnitt einer Dienst-Taxonomie mit dem Schwerpunkt auf IP-Telefonie gezeigt. Von besonderem Interesse sind die Klassen der *Rufkontrolldienste* (*call control services*), und der *nutzergenerierten Dienste* (*Custom User Service – CUS*). Erstere sind im Protokoll definierte Dienste, während letztere vom Nutzer erstellt und bereitgestellt werden können.

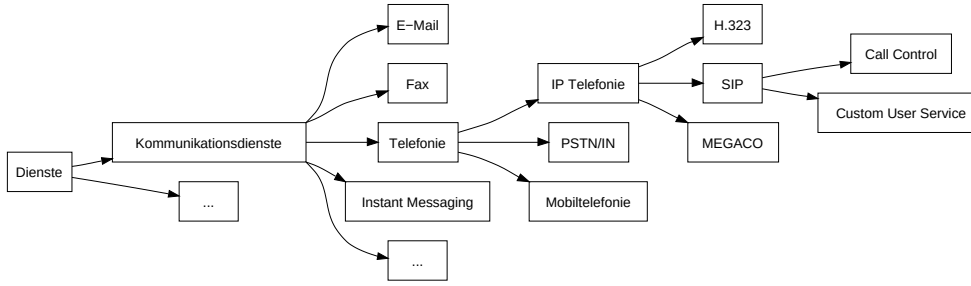


Abbildung 2.4: Taxonomie von Kommunikationsdiensten

Dienste für das traditionelle Telefonnetz wurden und werden von einer kleinen Zahl von Expertengruppen entworfen. Im Rahmen einer Standardisierung wird sichergestellt, dass diese Dienste fehlerfrei sind und eine korrekte Funktionalität auch zwischen unterschiedlichen Anbietern gewährleistet werden kann. Für das Intelligent Network sind die Rufkontrolldienste in den *Capability Sets* CS-1 [Int93b] und CS-2 [Int97a] definiert.

Dienste für IP-Telefonie-Systeme sind in Form der Protokollprimitive in den Kernstandards definiert. Mehrwertdienste sind in zusätzlichen ITU-T Recommendations sowie RFCs und Internet Drafts (IDs) spezifiziert. Insbesondere Drafts sind oftmals nicht genau spezifizierte Vorschläge von Einzelpersonen. Working Groups der IETF sorgen für eine Verifikation des Dienstablaufs von neuen Diensten für SIP. Formale Methoden finden jedoch gegenwärtig nur selten Anwendung.

Insbesondere im Umfeld des Session Initiation Protocols existiert eine Reihe von Entwicklungsmethoden, die die Erzeugung nutzergenerierter Dienste erlauben. Die einzelnen Konzepte werden im Detail in Abschnitt 2.3 beschrieben. Im Unterschied zu den Rufkontrolldiensten bietet diese Art der Dienste dem Nutzer die Möglichkeit, seine eigene Dienstfunktionalität zu realisieren. Der ambitionierte Nutzer erhält damit eine Flexibilität, die im PSTN/IN-Umfeld nicht zur Verfügung gestellt wird.

2.2 IP-Telefonie

Telefonie (von griech.: tele = fern, weit + phoné = Stimme) entwickelte sich über einen Zeitraum von über hundert Jahren und ist aus dem heutigen Alltag nicht mehr wegzudenken. Zu den Pionieren der Telefonie um 1860 zählen Antonio Meucci, Johann

Philipp Reis und Alexander Graham Bell. Das damals aufgestellte phänologische Grundprinzip der Sprachübertragung hat sich bis in die jüngste Vergangenheit kaum geändert. Zwischen beiden Gesprächspartnern wurde eine physische Leitung geschaltet, über welche die in elektrische Signale umgewandelte Sprache übertragen wird. Dies definiert auch den logischen Basisdienst „Telefonie“ bestehend aus Rufsteuerung und Medientransport.

Dem prinzipiellen Nachteil der permanenten Ressourcenbelegung, der durch die exklusive Nutzung einer Leitung auch bei Inaktivität entsteht, wird heute mittels verschiedener Verfahren begegnet. Durch Einsatz von Voice Active Recognition (VAD) können Gesprächspausen erkannt werden. Kenntnisse über Gesprächsverhalten lassen sich zu statistischen Mustern zusammenfassen. Diese Informationen erlauben es, ein *Multiplexing* durchzuführen, das die gleichzeitige Benutzung einer Leitung für mehrere Gespräche erlaubt. Heutige Time Division Multiplex (TDM)-Netze erzielen durch das (meist periodisch) zeitversetzte Senden von Daten große Einsparungen an Ressourcen.

Ein fundamentaler Unterschied zwischen traditioneller Telefonie und IP-Telefonie liegt in der Vermittlung der Informationen. IP-Telefonie nutzt hierzu ein Netzwerk, das dem Prinzip der *Paketvermittlung* folgt. Hierbei werden Signalisierungsnachrichten und Mediendaten als Nutzdaten in Pakete gepackt und einzeln über das Netzwerk übertragen. Die einzelnen Pakete können dabei unterschiedliche Routen nehmen. Abbildung 2.5 zeigt den prinzipiellen Ablauf einer Telefonieübertragung. Beginnend mit der Analog/Digital-Umwandlung des Sprachsignals werden die digitalen Daten kodiert und paketisiert. Die Pakete werden dann über heterogene Netze, die sich hinsichtlich ihrer qualifizierbaren Parameter wie Verlust (*loss*), Schwankungen (*jitter*) oder Verzögerung (*delay*) unterscheiden [Sch01] und in der Regel keine Dienstgüte (*Quality of Service – QoS*) garantieren können, übertragen. Auf der Empfängerseite werden die Pakete gegebenenfalls in die zeitlich richtige Reihenfolge gebracht und wieder in ein analoges Audiosignal umgewandelt.

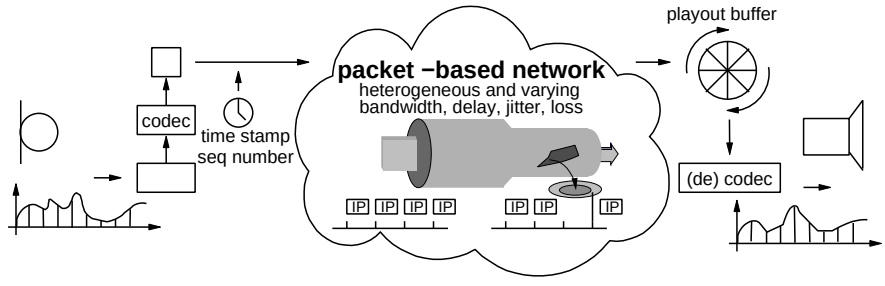


Abbildung 2.5: Prinzip einer paketbasierten Telefonie (aus [Ack03])

Weitverbreitete Computernetzwerke nutzten und nutzen paketvermittelte Netze zur Datenübertragung. Technologien hierfür sind Token Ring (IEEE 802.5) [ANS98] und Ethernet (IEEE 802.3) [ANS00]. Die Verwendung dieser existierenden Netze zur Übertragung der Sprache wurde Anfang der 80er Jahre untersucht. In den Xerox PARC Labs wurden 1983 Experimente mit dem Etherphone [SSO83] durchgeführt. In den

allgemeinen Fokus rückte die IP-Telefonie Ende der 90er Jahre. Durch die Verbreitung von breitbandigen Netzwerkzugängen in Haushalten mittels DSL oder Kabelmodems sowie die Bereitstellung von Rufnummern, die aus dem PSTN erreichbar sind, wurde und wird IP-Telefonie gegenwärtig zunehmend zu einem Massenprodukt.

Klassische Telefonie ist ein Dienst, an den sowohl Netzbetreiber als auch Kunden sehr strikte Anforderungen stellen. Heute erfüllen Telefonsysteme Anforderungen von 99,999% Verfügbarkeit. Sie gewährleisten ein korrektes Routing der Gespräche sowie eine genaue Abrechnung der geleisteten Dienste. IP-Telefoniesysteme müssen diese geforderten Anforderungen ebenfalls leisten, um eine Berechtigung neben der traditionellen Telefonie zu haben.

2.2.1 Konzeptionen und Architekturen

IP-Telefonie und das traditionelle Telefonnetzwerk (PSTN/IN) unterscheiden sich hinsichtlich ihrer Konzeption und Architektur in wesentlichen Punkten. Insbesondere die Lokation der Intelligenz, die zur Steuerung des Netzes und der Dienste notwendig ist, ist unterschiedlich gewichtet. Eine reine Aufteilung in Protokoll-basierter, zentraler, Endgeräte-basierter oder Middleware-basierter Erbringung der Intelligenz ist in der Realität nicht oder nur selten möglich.

2.2.1.1 Intelligentes Netz

Das Intelligente Netz (IN) ist aus den Ansätzen von Bellcore [II95] hervorgegangen. In den USA hat es sich zu dem Advanced Intelligent Network (AIN) entwickelt [Bel90] und in Europa wurde es von der International Telecommunication Union – Telecommunication Sector (ITU-T) standardisiert [Int]. Die Architektur des Intelligenten Netzes konzipiert die Trennung von *Dienststeuerung* und *Vermittlungsaktionen*. Die „Intelligenz“ wird durch zentral gesteuerte Dienste in einem abgeschotteten Netzwerk erbracht. Für das IN wurde eine dienstunabhängige Schnittstelle zwischen dem *Dienstzugangsknoten* und den *Dienststeuerknoten* (*Service Control Point – SCP*) definiert [Int92d].

Zur Realisierung eines Dienstes muss nur der Dienststeuerknoten mit einem neuen Programmcode verändert werden. Die Anpassung wird durch die Verwendung von dienstunabhängigen Funktionsblöcken (*Service Independent Building Block – SIB*) zur Komposition von Diensten vereinfacht. Jedoch haben Nutzer bewusst keinen Zugriff auf die Knoten Service Control Point (SCP) und Service Switching Point (SSP), um eigene Dienste anzubieten und zu nutzen. [Int92c, Int93a]. Ein Grund hierfür liegt in dem „Trust-by-Wire“-Sicherheitsparadigma, das eine physische Trennung als einen wichtigen Schutz ansieht. Darüber hinaus folgt die Konzeption des Dienstes im IN nur selten dem eines pro-Nutzer Dienstes, sondern eher dem eines Netzdienstes.

Die Übertragung der Signalisierung sowie der Medienströme erfolgt im PSTN/IN leitungsvermittelt. Für unterschiedliche Medientypen wie Daten oder Sprache wurden verschiedene Netzwerkkonzepte wie Asynchronous Transfer Mode (ATM), TDM, oder Frame-Relay (FR) entwickelt. System und Netzwerke im PSTN/IN wurden von einer

sehr kleinen Zahl von Herstellern und Anbietern entwickelt und betrieben. Die meisten Systeme sind monolithisch aufgebaut und *vertikal integriert*, d. h. für jedes dieser Entitäten wurde der gesamte Funktionsstack vom Transport über das Switching und Controlling bis hin zur Anwendungssignalisierung definiert und umgesetzt. Eine graphische Darstellung der vertikalen Integration der Netze ist in Abbildung 2.6 dargestellt. Die Realisierung integrierter Dienste für beispielsweise ein *Unified Messaging System* wird dadurch erschwert.

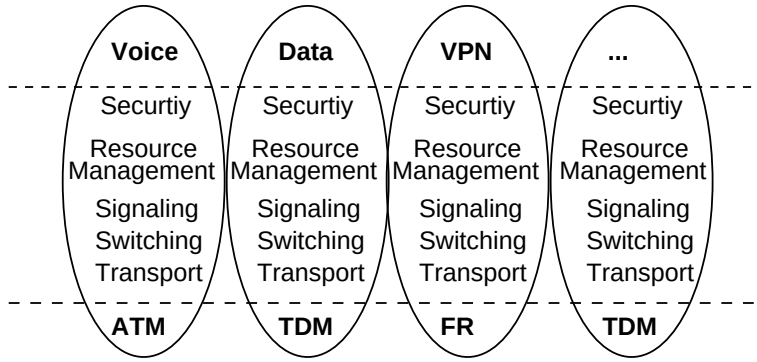


Abbildung 2.6: Vertikale Integration von Netzen

2.2.1.2 IP-Telefonie

Das Next Generation Network (NGN) besteht aus einem IP-basierten Kernnetz, auf das andere Netze mittels Gateways als Zugangsnetze fungieren. Es erlaubt eine *horizontale Integration* der einzelnen Schichten. Jede Schicht (z. B. Transportschicht, Signalisierungsschicht oder Anwendungsschicht) bietet ihre Funktionalität *Ende-zu-Ende* an. Dies vereinfacht die Bereitstellung von nahtlos verzahnten Diensten (*seamless services*). Eine Darstellung der horizontalen Integration ist in Abbildung 2.7 zu sehen. Das heterogene IP-Netzwerk stellt hierbei die gemeinsame Plattform dar, auf der die Schichten aufsetzen. Das IP-Netzwerk stellt dabei den darüber liegenden Schichten eine transparente Konnektivität (*transparent connectivity*) zwischen den Netzentitäten zur Verfügung. Dies ist ein signifikanter Unterschied zum PSTN, in welchem die Kommunikation zwischen direkt verbundenen Entitäten stattfindet.

IP-Telefonie bildet eine Applikation eines solchen NGN. Die Erbringung der Dienste erfolgt bei einer IP-Telefonie-Architektur in den *Endsystemen* oder in *Signalisierungsservern*, die in der Nähe der Endsysteme lokalisiert sind. Diese Anordnung folgt konzeptionell den Komponenten *Host* und *Router* auf Schicht 3 des IP Netzwerks. Endsysteme steuern durch Versenden von Signalisierungsnachrichten die Multimediasitzungen. Die Signalisierungsserver stellen die notwendige Routingfunktionalität auf der Anwendungsschicht bereit.

Im Unterschied zum Intelligenten Netz ist der Markt für IP-Telefonie-Lösungen von Anfang an dereguliert gewesen. Eine Vielzahl von kleineren und größeren Anbietern

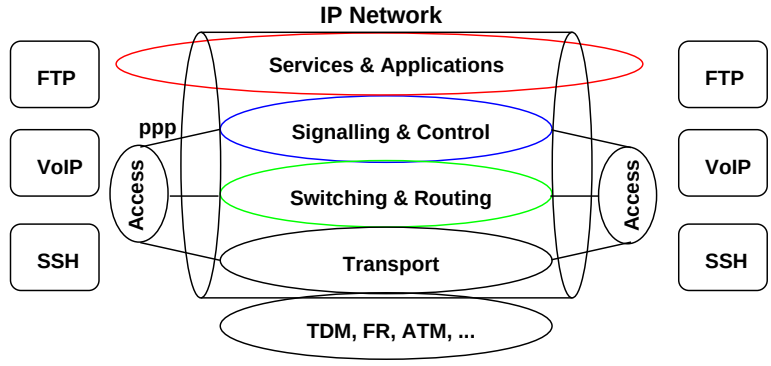


Abbildung 2.7: Horizontale Integration von Netzen

ist am Markt vertreten und kann dem Nutzer Dienste zur Verfügung stellen. Auch können Nutzer über Mechanismen wie die Call Processing Language (CPL) (Unterabschnitt 2.3.2.1) oder Telefonie-APIs (Unterabschnitt 2.3.1) eigene Dienste erstellen und nutzen. Der Transport der Daten und die Diensterbringung können von verschiedenen Providern realisiert werden. Soll eine Sicherung der Signalisierungs- oder Medienpakete erfolgen, so muss diese mit kryptographischen Mechanismen wie Transport Layer Security (TLS) oder Secure RTP (S/RTP) erfolgen.

2.2.2 IP-Telefonie-Signalisierung

Zur Bereitstellung der Telefoniefunktionalität sind zwei verschiedene Schichten notwendig. Zum einen *Signalisierung* und zum anderen *Medientransport*. Signalisierungsnachrichten dienen der Steuerung von Telefoniesitzungen[1]. Die Kontrollnachrichten steuern den Aufbau, die Modifizierung der bestehenden Sitzung sowie den Abbau der Sitzung. Der Medientransport stellt die Übermittlung der Mediendaten zwischen beiden Gesprächspartnern sicher. Die Medienpakete können dabei einen anderen Pfad entlang des Netzwerks nehmen als die Signalisierungsnachrichten. Generell besitzen die Signalisierungsprotokolle in der IP-Telefonie Protokollelemente, welche ausdrucksstärkere Informationen enthalten können, als dies im PSTN der Fall ist. Zusätzlich ist die Signalisierung insbesondere bei SIP einfacher zu erweitern, bei gleichzeitig beibehaltener Interoperabilität.

Das Real-time Transport Protocol (RTP) [SCFJ96] hat sich als de-facto Standard für den Transport von verzögerungssensitiven Mediendaten in interaktiven Anwendungen mit Echtzeitanforderungen etabliert. Alle zur Zeit standardisierten Signalisierungsprotokolle für IP-Telefonie-Anwendungen nutzen RTP zur Übertragung der Echtzeitmediendaten. Für die Signalisierung von IP-Telefonie Sitzungen habe sich drei Signalisierungsprotokolle etabliert: H.323, SIP, und Megaco/H.248. MEGACO/H.248 (MEdia GAteway COntrol (MEGACO)) [CGR+00, Int00b] ist eine gemeinsame Spezifikation

[1] Da in der IP-Telefonie in der Regel ein verbindungsloser Transport der Daten verwendet wird, wird der Begriff „Sitzung“ anstelle von „Verbindung“ verwendet.

der IETF und der ITU eins Media Gateway Cntrol Protocol (MGCP) [AF03] und verfolgt einen anderen Ansatz als H.323 und SIP. Das Protokoll wird zur Steuerung von Mediengateways durch eine andere Entität eingesetzt. H.323 und SIP werden in den folgenden Abschnitten näher betrachtet. Für die weitere Arbeit wird SIP als Signalisierungsprotokoll verwendet und erweitert. Die hierbei entstandenen prinzipiellen Erkenntnisse lassen sich allerdings auch mit entsprechendem Aufwand auf H.323 übertragen, da beide Protokolle, die von der konzeptionellen Ausrichtung ähnlich sind.

2.2.2.1 H.323 Protokoll-Suite

Die Study Group 16 der ITU-T veröffentlichte im Dezember 1996 die Recommendation H.323 [Int96b] mit dem Titel „Visual telephone systems and equipment for local area networks which provide a non-guaranteed quality of service“. Der Standard ist auf Echtzeit-Videokonferenzen über lokale Netze, die keine Dienstgüte (*Quality of Service – QoS*) garantieren, fokussiert. H.323 wird kontinuierlich weiterentwickelt und liegt mittlerweile in der Version 5 [Int03b] vor. Die aktuelle Fassung unterstützt insbesondere besser die Eigenschaften von Weitverkehrsnetzen (*Wide Area Network – WAN*) und stellt Methoden bereit, die den Sitzungsaufbau beschleunigen.

H.323 stellt dabei ein Rahmenprotokoll dar, das die notwendigen Netzelemente definiert und eine Reihe weiteres Substandards bestimmt, welche die eigentliche Signalisierung definieren. H.323 wurde dabei unter der speziellen Maßgabe einer umfassenden Zusammenarbeit mit Integrated Services Digital Network (ISDN) entworfen. Die Protokollelemente der Substandards folgen daher dem Q.931-Protokoll [Int98c].

Die Steuerung der Multimedia-Sitzungen wird im Standard H.225.0 [Int03a] festgelegt. Die ITU-Recommendation H.245 [Int00a] beschreibt die Aushandlung der Medienparameter. Für die Übertragung der audio-visuellen Daten wird die Verwendung von RTP [SCFJ96] und das dazugehörige Kontrollprotokoll Real-time Transport Control Protocol (RTCP) festgelegt. H.323 umfasst im wesentlichen den einfachen Basisdienst (*Basic Call*). *Mehrwertdienste* (*Supplementary Services*) wie *Anrufweiterleitung* (*Call Forwarding*), *Rückruf bei Besetzt* (*Call Completion on Busy*) sind in der H.450.*n* Protokollserie auf Basis des Frameworks H.450.1 [Int98a] definiert. Jeder Dienst wird dabei durch eine vom Basisdienst unabhängige Zustandsmaschine (*Finite State Machine – FSM*) definiert, wie dies in Abbildung 2.8 dargestellt ist.

2.2.2.2 Session Initiation Protocol – SIP

Ein alternativer Ansatz zur Steuerung von Multimediasitzungen (*Multimedia Sessions*) wurde im März 1999 von der *Multiparty Multimedia Session Control Working Group* (MMUSIC) innerhalb der IETF entwickelt. Dieser wurde unter dem Titel “Session Initiation Protocol (SIP)“ als Request For Comments (RFC) 2543 [HSSR99] veröffentlicht. SIP folgt der Philosophie der Internet Engineering Task Force (IETF) und definiert nur die notwendigen Bausteine, die nicht bereits durch andere existierende

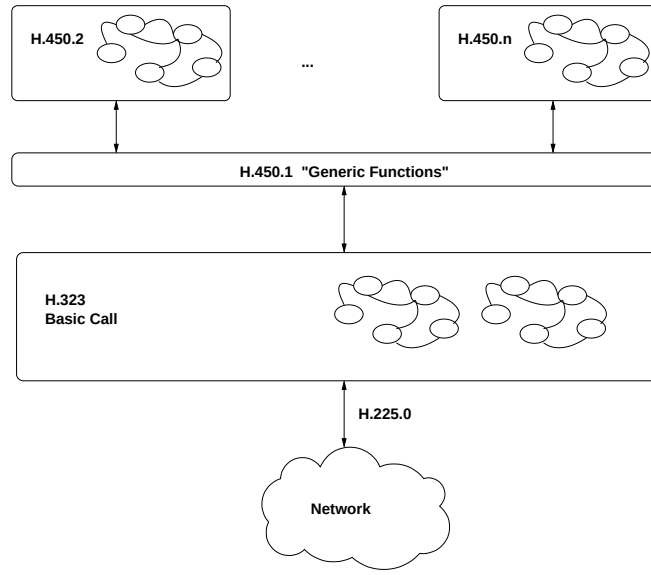


Abbildung 2.8: H.323 Mehrwertdienste nach H.450.1

Protokolle abgedeckt sind. Abbildung 2.9 zeigt das Zusammenspiel von SIP mit anderen Protokollen.

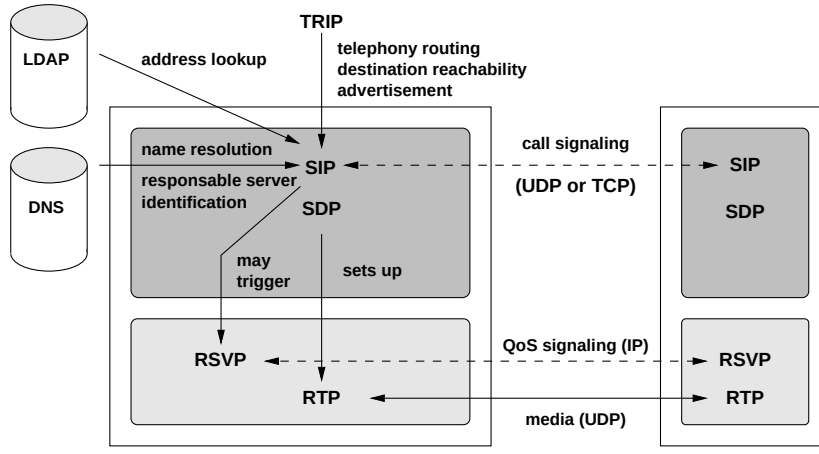


Abbildung 2.9: Zusammenspiel der Protokolle in SIP-Umgebungen

Mit dem RFC 3261 [RSC+02] wurde 2002 eine neue Version des Kernstandards vorgelegt. Wesentliche Ergänzungen und Verbesserungen liegen in der deutlicheren Darstellung unter Verwendung einer Augmented Backus-Naur-Form (ABNF) [CO97] sowie einer deutlichereren Beschreibung der Signalisierungsabläufe. Weiterhin beschreiben zahlreiche IETF Drafts (IEFT-ID) sowie RFCs Erweiterungen zum Session Initiation Protocol. Die Koordination darüber übernimmt die IEFT Working Group SIP [www23].

Zusätzliche Dienste müssen in Konformität mit dem *Call Control Framework* [Cam01] definiert werden, um Interoperabilitätsprobleme zu vermeiden.

Der SIP Kernstandard definiert ein Client/Server-*Anwendungsschichtprotokoll* (*application layer protocol*) für den endgerätebasierten *Sitzungsaufbau*, die *Modifikation* von bestehenden Sitzungen sowie den abschließenden *Sitzungsabbau*. Dazu definiert der Standard die Syntax von *Signalisierungsnachrichten* (*messages*) sowie deren Semantik. Hierbei zeigt sich eine große Verwandtschaft im Aufbau mit dem *HyperText Transport Protocol* (HTTP) [FGM+99]. Die Unterscheidung zwischen Client und Server bezieht sich auf die Prozesse und muss nicht zwangsläufig auch eine bestimmte Entität meinen.

Signalisierungsnachrichten in SIP sind in ASCII kodiert und bestehen aus einem Nachrichtenkopfteil (*header*) mit Steuerungsinformationen sowie dem eigentlichen Nachrichtenkörper (*body*). Die Struktur einer SIP-Nachricht ist in Abbildung 2.10 abgebildet. Der Type des SIP-Nachrichtenkörpers wird als MIME-Type [FB96] in der Headerzeile `Content-Type` definiert. Im Standard wird die Unterstützung des Typs "application/sdp" für SDP-Medienbeschreibungen als stets zu erfüllende Minimalanforderung festgelegt. Die transportierten Informationen beschreiben üblicherweise die Sitzung zwischen den Teilnehmern.

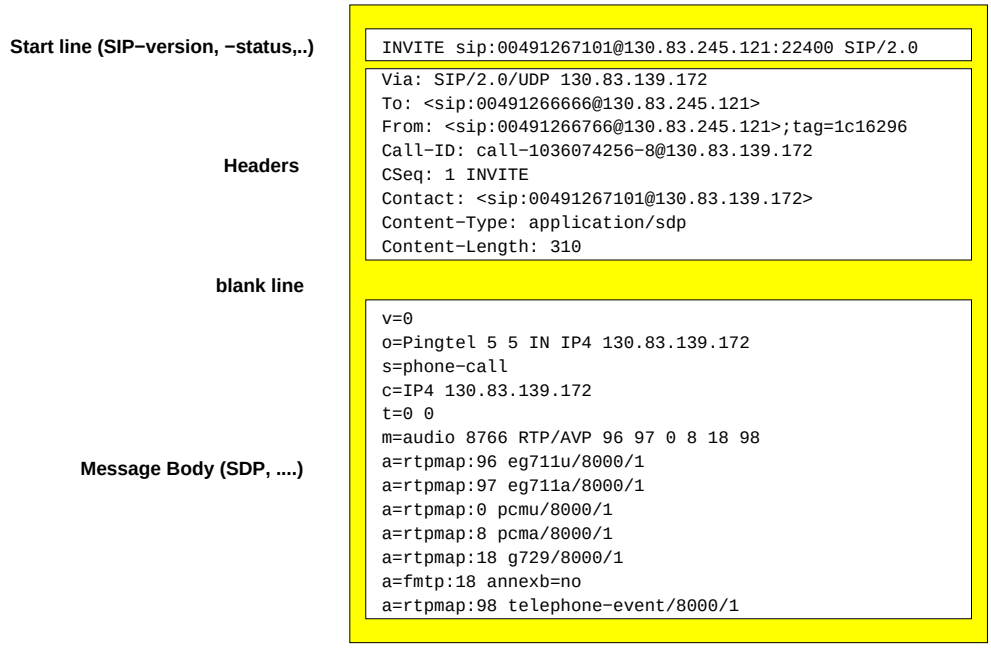


Abbildung 2.10: Aufbau einer SIP-Nachricht

Zur Beschreibung und Aushandlung der zu verwendenden Mediencharakteristika einer SIP-Sitzung wird von der IETF die Verwendung das *Session Description Protocol* (SDP) [HJ98] empfohlen. Es handelt sich dabei nicht um ein Protokoll, sondern um eine strukturierte Beschreibung von Medien durch Angabe der verwendbaren Codecs sowie

Tabelle 2.1: Auflistung der im RFC 3261 standardisierten SIP Methoden und ihre Bedeutungen

Methode	Bedeutung
REGISTER	Anmelden der Entität an einem SIP-Server (Registrar) durch Bekanntgabe der Kontaktinformationen
INVITE	Transport von Informationen zur Initierung und Modifikation von SIP-Sitzungen
ACK	Abschließende Operation im 3-Wege Handshake in SIP
CANCEL	Abbruch einer offenen Anfrage
BYE	Terminierung einer Sitzungen
OPTIONS	Abfragen des Leistungsumfangs (*capabilities*) anderer Teilnehmer

Start- und Stoppzeiten. Zusätzlich enthält die Beschreibung Informationen zu den zugehörigen Verbindungen wie z. B. Portnummern und IP-Adressen. Die Verarbeitung von SDP ist in der Spezifikation [RS02] als formaler Angebot/Antwort-Prozess (*offer/answer exchange process*) für SIP beschrieben, der die Limitierungen, die sich aus der Herkunft von SDP aus dem Multicast-Umfeld ergeben, für IP-Telefonie-Anwendungen aufhebt.

Zur Diensterbringung wurde im Standard für SIP das Anfrage/Antwort-Transaktionsmodell (*request/response transaction model*) spezifiziert. Jede Transaktion besteht aus einer *Anfrage* (*request*), die eine Methode aufruft und mindestens einer dazugehörigen *Antwort* (*response*). Der prinzipielle Ablauf zwischen zwei SIP-Endgeräten, *User Agent* (UA) genannt, ist dabei der Folgende: Der *User Agent Client* (UAC) initiiert, beispielsweise als Reaktion auf eine Nutzerinteraktion, eine Anfrage, welche im SIP-Kontext als Methoden (*methods*) bezeichnet wird. Diese wird an den *User Agent Server* (UAS) geschickt, der diese Nachricht verarbeitet und Aktionen, gegebenenfalls mit Nutzerinteraktion, ausführt. Abschließend sendet der UAS eine oder mehrere Antworten an den Initiator zurück.

Gegenwärtig sind im SIP Kernstandard [RSC+02] 6 Methoden definiert. Diese sind mit ihrer Bedeutung in Tabelle 2.1 aufgeführt. Als Reaktion auf eine eingehende Methode antwortet der Server-Prozess des Empfängers mit einer Response-Nachricht. Diese identifizieren sich über einen von HyperText Transport Protocol (HTTP) übernommenen und erweiterten Nummerncode. Hierbei zeigt die erste Ziffer die Klasse des Responses an. So zeigt beispielsweise $1xx$ eine informelle Nachricht, $2xx$ eine erfolgreiche Antwort und $5xx$ eine Fehlermeldungen an. Eine vollständige Liste aller Fehlercodes sowie erweiterter SIP-Methoden befinden sich in Anhang F.1.

Den Funktionalitätssumfang erweiternde Methoden und Protokolle sind in einer Reihe von RFCs und Internet Drafts definiert. So wird beispielsweise der Mehrwertdienst *Anrufweiterleiten* mittels der REFER-Methode [Spa03] realisiert. In [Roa02] wird ein

Tabelle 2.2: Gegenüberstellung von Entitäten in SIP, H.323 und PSTN/IN

Entität	SIP	H.323	PSTN/IN
Endsystem	User Agent	Terminal	Telefon
Signalling server	SIP Proxy	(Gatekeeper)	Service Control Point (SCP), Service Switching Point (SSP)
Router	Router	Router	Service Transfer Point (STP)
Gateway		Signalisierungsgateway	
Conferencing	Individuelle Lösungen	Multiparty Control Unit (MCU)	Individuelle Lösungen

subskriptionsbasierter Eventmechanismus durch die Methoden SUBSCRIBE und NOTIFY bereitgestellt. Die Verwendung von SIP für Instant Messaging Systeme wird in [CRS+02a] beschrieben.

2.2.2.3 SIP Netzwerkentitäten

Im Unterschied zum IN verfolgt SIP die Maxime „fast in core, smart at the edges“. Daraus folgt, dass ein substantieller Anteil der Intelligenz in diesem verteilten System in den Endgeräten bzw. in deren Nähe lokalisiert ist. Zwei prinzipielle Netzwerkentitäten können unterschieden werden: zum einen Endsysteme (ES) und zum anderen Intermediate-Systeme (IS). Die nachfolgende Betrachtung bezieht sich auf Netzwerkentitäten, wie sie im Umfeld des Session Initiation Protocols definiert worden sind. Eine Gegenüberstellung von äquivalenten Entitäten in SIP, H.323 und im PSTN/IN findet sich in Tabelle 2.2.

SIP User Agent

Der SIP *User Agent* (UA) ist ein Endsystem, welches einem Telefon (im PSTN) und dem Terminal (in H.323) entspricht. Ein solches Endsystem kommt im Kontext der IP-Telefonie in vielen Ausprägungen vor. Ein IP *Soft-phone* bezeichnet dabei eine Softwarelösung, die auf dem PC oder einem Personal Digital Assistant (PDA) [AGKS01] ausgeführt wird. Mittels einer Kombination aus Kopfhörer und Mikrophon (*headset*) kann der Nutzer telefonieren. Spezielle Audio-Hardware bietet eine Sprachqualität, die je nach eingesetztem Codec mit der Qualität der klassischen Telefonie bzw. Mobilkommunikation vergleichbar ist.

Die zweite Klasse der IP Telefone stellen dedizierte Endgeräte dar, die der Optik und Bedienung klassischer Telefone ähneln. Üblicherweise bieten diese Endgeräte eine Reihe von zusätzlichen Leistungsmerkmalen an. Diese nutzen die Rechenleistung des Prozessorkerns (CPU) sowie die erweiterten Ein- und Ausgabeschnittstellen (z. B. Liquid Crystal Display (LCD), Bluetooth oder Web Interface). Oftmals stehen dem Nutzer

Programmierschnittstellen (*Application Programming Interface – API*) für eigene Erweiterungen zur Verfügung.

Abstrahiert betrachtet besteht, der UA intern aus einem Client-Teil, dem User Agent Client (UAC) und einem Server-Teil, dem User Agent Server (UAS). Der User Agent (UA) ist ein vollständiger Internet Host, wie er in [Bra89] beschrieben ist. Darüber hinaus verwaltet der UA die kompletten Zustände (*call states*) aller initiierten Sitzungen. Der Signalisierungspfad und der Medienpfad haben im User Agent (UA) ihren Start- oder Endpunkt.

Der Rufaufbau wird in SIP generell nur von SIP-Endsystemen ausgelöst. Dienste, die eine Nutzerinteraktion erfordern, werden im User Agent (UA) ausgeführt. Obwohl im Signalisierungspfad lokalisiert, wird der Back-2-Back User Agent (B2BUA) als Endgerät kategorisiert. In seinen Ausprägungen als *Gateway* [Ack03] terminiert er die Signalisierung in der jeweiligen Domäne und initiiert in der anderen Domäne die zweite Sitzungshälfte (*call leg*).

SIP Server

Im Standard werden eine Reihe von *SIP Servern* (Proxy Server, Redirect Server und Registrar) und deren Funktionalität beschrieben. Die einzelnen Server sind dabei logische Einheiten, die nicht notwendigerweise auf physikalisch verschiedenen Systemen laufen müssen. SIP Server sind Intermediate-Systeme, die auf der Anwendungsschicht (*application-level*) das Routing der Signalisierungsnachrichten kontrollieren. Sie sind in ihrer Funktionalität dem Service Control Point (SCP) aus dem Intelligent Network (IN) ähnlich.

Proxy Server: Diese Komponente leitet eingehende Request-Nachrichten an ein oder mehrere Endsysteme bzw. SIP-Server weiter. Dabei kann der Proxy zustandslos (*stateless*) oder zustandsbehaftet (*stateful*) arbeiten. Response-Nachrichten werden zum Sender des Requests zurückgeleitet.

Redirect Server: Dieser SIP-Server erwidert eingehende SIP-Requests mit einer Response-Nachricht, die die neue temporäre oder permanente Adresse des Zielsystems enthält.

Registrars: Dieser verbindet die symbolischen Kontaktinformationen aus den REGISTER-Nachrichten mit einer physisch erreichbaren Adresse. Diese Bindungen (*bindings*) werden als Lokalisationsdienst (*location service*) SIP-Proxies zur Verfügung gestellt.

2.3 Ausgewählte Architekturen, Konzepte und Methoden zur Diensterstellung und Diensterbringung

Die *Diensterstellung* ist ein Prozess, der die Entwicklung von Diensten und Systemen, der Einrichtung von Diensten sowie das gesamten Management umfasst. Es hat sich

hierfür auch der Begriff *Service Engineering* etabliert, der die Anlehnung an die Disziplin des Software Engineerings [Ree95] verdeutlicht. Eine *Dienstarchitektur* (*service architecture*) lässt sich daher auch als Softwarearchitektur auffassen, da sie, wie auch die Dienste in der Regel vollständig in Software realisiert ist. Eine Definition einer Dienstarchitektur ist in [Kel02] beschrieben und nachfolgend angegeben.

Definition 2.5 (Dienstarchitektur)
*Eine **Dienstarchitektur** ist ein Satz an Modellierungskonzepten, Prinzipien und Bedingungen für den Entwurf von Dienstsystemen. Zur Dienstarchitektur gehören die Struktur der Komponenten, aus denen sie besteht und die Art der Wechselwirkung zwischen den Komponenten.*

Die *Diensterbringung* in heterogenen Netzen kann durch unterschiedliche Ansätze erfolgen. So bietet das Next Generation Network (NGN) ein *all-IP-Netz* als Backbone an, an den sich andere Netze als Zugangsnetze ankoppeln und über Gateways interoperabel bleiben. Die Endsysteme übernehmen bei dieser Ausprägung die Intelligenz und die Dienststeuerung. Die Verwendung einer Kopplung auf Dienstebene über offene Schnittstellen (APIs), wie dies mit Parlay [www18] unternommen wird, ermöglicht es, die Netzinfrastruktur weitgehend unverändert zu lassen. Eine einheitliche Middleware, die als Signalisierungsebene zwischen der Netz- und der Dienstschicht ihre Funktionalität anbietet, ist eine weitere Alternative. Das Session Initiation Protocol (SIP) stellt eine solche Signalisierungsmiddleware dar.

Die erwähnten Ansätze erfüllen die Anforderungen, welche die *Dienststeuerung*, das Herzstück eines Kommunikationssystems, erfordert. Die Aufgaben lassen sich in die folgenden Gruppen unterteilen [Kel02]: Teilnehmer-bezogene Aufgaben (z. B. Registrierung, Authentisierung, Management), Dienst-bezogene Aufgaben (z. B. Dienstlogik, Dienstablauf, Dienstlebenszyklus), Kommunikationsressourcen-bezogene Aufgaben (z. B. Steuerung der Medien, Aushandlung der Gerätefähigkeiten, Auswahl der Komponenten) und allgemeine Aufgaben (z. B. Logging, Management).

Es existiert eine Reihe von unterschiedlichen Architekturen, wie die standardisierten Ansätze TINA, PSTN, IN, aber auch interessante Forschungsergebnisse wie die flexible Signalisierungsarchitektur Advance Multimedia Signaling Architecture (AMSA) [Mül96] oder die Server Architecture for network independent Multimedia Service cOntrol in heterogenous communication Networks (SAMSON) [Kel02].

Im Rahmen dieser Arbeit fokussiert sich die Betrachtung von Dienststeuerung und -erstellung auf in der IP-Telefonie verwendete Konzepte. Insbesondere API-basierte Verfahren werden nachfolgend analysiert. Neben den vorgestellten Ansätzen sind weitere Verfahren vorgeschlagen worden. Dazu zählen SIP-Servlets [KBK00, Jav02] oder VoiceXML [MBC+04], die hier nicht weiter betrachtet werden, da sie keine große Relevanz für die anvisierten Kommunikationsdienste haben.

2.3.1 API-basierte Diensterstellung

Eine Programmierschnittstelle (*Application Programming Interface (API)*) bietet eine horizontale Schnittstelle, die in der Regel eine Abstraktion der darunter liegenden Schicht zur Verfügung stellt. Zur Erstellung von Kommunikationsdiensten werden diese TAPI (*Telephony Application Programming Interface*) genannt. Diese werden von Kommunikationsanbietern als Trennung zwischen der Dienststeuerung und der Kommunikationssteuerung angeboten.

Mehrere Gruppierungen versuchen eine standardisierte und modularisierte API zur Erstellung von Kommunikationsdiensten zu etablieren. Zu den bekannten und verbreiteten Ansätzen zählen *Parlay* und *JAIN*. Beide Konzepte bieten jeweils eine einheitliche Schnittstelle, die die Entwicklung von Diensten für unterschiedliche und heterogene Netze vereinfachen soll.

2.3.1.1 Parlay

Die Parlay Group [www18] verfolgt das Ziel einer offenen und gesicherten Programmierschnittstelle. Diese Schnittstelle soll externen Dienstanbietern einen geordneten Zugriff auf die netzinternen Informationen geben. Der Zugriff auf die Netzressourcen wird über das *Parlay-Gateway* realisiert. Die gesicherten Steuerungsfunktionen für diese Ressourcen werden durch *Service Interfaces* zur Verfügung gestellt. Die angebotenen Dienste werden in die fünf Gruppen *Call Control Service*, *User Interaction Services*, *Mobility Services*, *Messaging Services* und *Connectivity Services* unterteilt.

Dienste geben ihre Steuerungsbefehle (z. B. Rufaufbau) über die Schnittstelle an das Netz weiter, umgekehrt werden Ereignisse der Netzressourcen (z. B. Teilnehmer belegt) wieder an die Dienste gereicht. Die Schnittstellen bieten generische Rufmodelle von unterschiedlicher Komplexität, die vom einfachen *Basic Call* bis hin zum *Multiparty Multimedia Conferencing* reichen. Die Schnittstellen selbst sind in der Unified Modelling Language (UML) beschrieben, und unabhängig von der eigentlichen Implementierung.

2.3.1.2 JAIN

Das Framework *JAVA APIs for Integrated Networks* (JAIN) wurde 1998 von Sun Microsystems [www26] ursprünglich als *JAVA Advanced Intelligent Network* eingeführt [BBD00]. Die Intention von JAIN ist die Bereitstellung von Schnittstellen, die es in JAVA geschriebenen Programmen erlauben, Telefoniedienste zu steuern. JAIN stellt dabei im Wesentlichen eine JAVA-Spezifikation einer Parlay-API dar. Im verwendeten Rufmodell (*call model*) und der adressierten Netzarchitektur mit Switchen zur Rufsteuerung folgt das JAVA APIs for Integrated Networks (JAIN)-Konzept den Prinzipien des INs. Als Dienste werden ausgelöste Ereignisse (*events*) bezeichnet, die Rufzustände bestehender Rufe verändern können.

JAIN definiert die Kommunikationstechniken zwischen der Client-Applikation und den Parlay-Interfaces. Die JAIN-API besteht aus den protokollunabhängigen APIs für die Rufsteuerung (JAIN Call Control (JCC)) und für die Rufverarbeitung (JAIN Coordination and Transaction (JCAT)). Diese unterteilen sich weiter in Kontrollstrukturen für *intelligente Endsysteme* (*intelligent peripherals*) und in Schnittstellen für die Kontrolle von Netzwerk-Switches [JAMS00]. Zusammen mit der Managementschnittstelle (*JSLEE*) bietet JAIN Drittanbietern von Diensten eine JAVA-basierte Parlay-API an.

2.3.1.3 SIP-CGI

Das in [LRS01] vorgeschlagene *SIP Common Gateway Interface* (SIP-CGI) bietet eine von der eingesetzten Programmiersprache unabhängige Methode zur Realisierung von Diensten. Es ist daher sowohl für Hochsprachen wie C, C++ oder Java aber auch für Skriptsprachen wie Tcl/Tk, Perl oder Phyton, die für ein schnelles Prototyping (*rapid protoyping*) eingesetzt werden können, interessant. Dabei stellt SIP-CGI eine administrative low-level Schnittstelle zur Steuerung der Dienste dar. Die primären Entitäten, auf denen SIP-CGI ausgeführt wird, sind SIP-Server wie Proxy, Redirect oder Registrar Server. Auf einem User Agent wird in der Regel kein SIP-CGI laufen.

SIP-CGI hat viele Ähnlichkeiten zum Common Gateway Interface (CGI), wie dieses für HTTP existiert [CR99]. Diese liegt nicht zuletzt an der konzeptionellen Verwandtschaft (z. B. RFC822-basierte [Cro82] Syntax) von SIP und HTTP. Diese Ähnlichkeit macht die Verwendung von SIP Common Gateway Interface (SIP-CGI) für Programmierer, die bereits mit den Mechanismen des HTTP-CGI vertraut sind, attraktiv. Der Einarbeitungsaufwand kann dadurch niedrig gehalten werden. Das SIP-CGI-Modell ist in Abbildung 2.11 dargestellt. Der linke Teil mit dem SIP-Client dem SIP-CGI-Server entspricht dabei dem Modell von HTTP-CGI, der rechte Teil ist SIP-CGI-spezifisch.

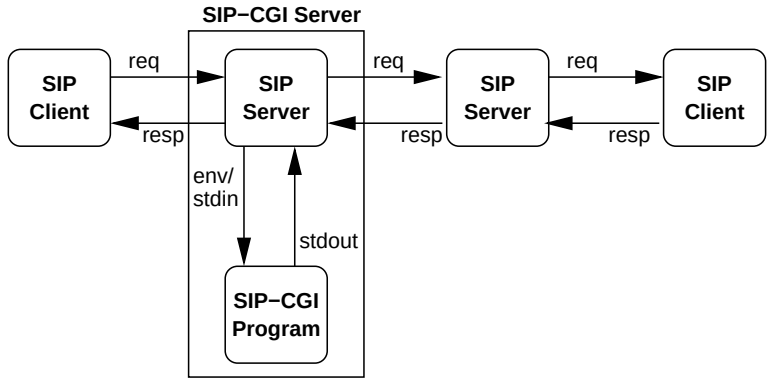


Abbildung 2.11: Das SIP-CGI Modell (aus [Len04])

Ein wesentlicher Unterschied zwischen beiden CGI-Varianten ist das *Persistenzmodell* (*persistence model*). Dieses erlaubt dem Dienst, die Kontrolle über einen persistenten Status über den Austausch einer Reihe von Nachrichten zu behalten. In HTTP-CGI

wird jedes CGI-Skript einmalig für jede Anfrage gestartet. Es generiert nach der Verarbeitung eine Antwort und das Skript beendet sich anschließend. Für den jeweiligen Vorgang bzw. den jeweiligen Nutzer wird kein Zustand gehalten. Dadurch kann keine Zuordnung von zwei verschiedenen Anfragen zum selben Nutzer erfolgen. Da es in HTTP-CGI keinen inhärenten Mechanismus zur zustandsbehafteten Ausführung der Transaktionen gibt, müssen komplexere Skripte auf die Verwendung von clientseitigen Cookies [KM00] oder auf speziell erzeugte Uniform Resource Locators (URL) zurückgreifen.

Jeder SIP-Request kann weitere neue und von Proxys generierte Requests erzeugen, die ihrerseits Antworten auslösen. Ein CGI-Skript muss die Zusammenhänge und Zustände aller dieser Ereignisse erfassen und verwalten können. Der Standard von SIP-CGI sieht zur Aufrechterhaltung der Zustände ein *Zustands-Token* (*state token*) vor. Dieses Token ist für den Server nicht einsehbar (*opaque token*) und wird vom CGI-Skript mittels eines Meta-Headers an den Server übermittelt. Wenn das CGI-Skript für eine über mehrere Schritte fortdauernde Transaktion erneut aufgerufen wird, so wird das Token über Umgebungsvariablen (*environment variables*) vom Server weitergereicht. Wird die finale Antwort-Nachricht an den Client gesendet, so wird danach das Token zerstört. Bei einer neuen Transaktionsanfrage wird initial kein Token übertragen.

2.3.2 Skript-basierte Diensterstellung

Die Verwendung von *Skript-Sprachen* [Ous98, SN99] stellt einen sehr interessanten Ansatz zur Erstellung von Diensten dar. Der Einsatz bietet insbesondere beim *Rapid-Prototyping* große Vorteile. In der Regel sind Skriptsprachen einfacher zu erlernen als andere Programmiersprachen, besitzen jedoch meist einen geringeren Funktionsumfang. Zur Erstellung von Nutzer-zentrierten Diensten hat sich die Call Processing Language (CPL) als weitverbreiteter Mechanismus etabliert.

2.3.2.1 Call Processing Language – CPL

Die high-level Skriptsprache *Call Processing Language* (CPL) stellt einen Ansatz dar, der sich primär auf die sichere Diensterstellung für externe Entwickler und ungeübte Nutzer richtet [LS00a, LWS04]. Die Sprache ist unabhängig von einer Signalisierungsarchitektur oder den verwendeten Signalisierungsprotokollen. Gegenwärtig ist CPL im Wesentlichen im SIP-Umfeld verbreitet. Eine einfache Diensterstellung ist ein weiteres Ziel der Sprache. Die Konzeption von CPL erlaubt die Unterstützung durch einen Editor zur graphischen Repräsentation und Modifikation bestehender Dienstbeschreibungen. Dennoch lässt sich ein solches CPL-Skript vollständig und umfangreich von erfahrenen Programmierern direkt lesen und editieren.

In seiner Intention folgt es dem Dienstmodell aus dem Intelligent Network (IN), welches eine formalisierte und strikte Trennung zwischen den Switches und der Dienstlogik besitzt. IN-Dienste bestehen aus der Verbindung von vordefinierten Service Independent

Building Blocks (SIB). Die einzelnen SIB können in Form eines gerichteten azyklischen Graphen (*Directed Acyclic Graph – DAG*) dargestellt werden.

Auch CPL verwendet eine Entscheidungsgraph-basierte Technik in Form eines Directed Acyclic Graph (DAG), der den Fluss der Programmausführung beschreibt. Der Umfang der Sprache ist bewusst limitiert worden. So gibt es keine Schleifenkonstrukte, Rekursionen, Variablen und Möglichkeiten externe Programme auszuführen. CPL ist daher keine Turing-vollständige Sprache. Dadurch ist es möglich, Worst-Case Abschätzungen zu machen und eine begrenzte Ressourcenbelegung sowie eine wohldefinierte Terminierung des Programms zu garantieren. Diese konzeptionelle Ausrichtung wurde von der E-Mail Filtersprache *Sieve* [Sho01] übernommen.

Für den Netzwerk- oder Kommunikationsprovider bietet die Laufzeitgarantie sowie die Einschränkung des Funktionsumfangs der Sprache eine geeignete Methode, seine Ressourcen für Personen freizugeben, zu denen kein oder nur ein eingeschränktes Vertrauensverhältnis besteht. Nutzer können weder willentlich noch unbeabsichtigt Schaden am System anrichten. Durch eine Überprüfung der Korrektheit lassen sich Dienste schon vor der Ausführung testen.

Eine CPL-Dienstbeschreibung besteht aus einer baumartigen Anordnung von *Knoten* (*nodes*). Jeder Knoten stellt eine auszuführende *Aktion* (*action*) dar. Ein Knoten besitzt optional Ausgänge (*outputs*), die in Abhängigkeit vom Ergebnis der Aktion oder der Bedingung der Verzweigung verfolgt werden und zum nächsten Knoten führen. Damit entspricht ein Knoten dem in Abbildung 2.2 dargestellten Modell. Die Ausführung des Pfades endet mit einem Knoten, der keine weiteren Ausgänge mehr besitzt. Parameter können den Knoten und sein Verhalten näher spezifizieren. Im CPL-Standard sind vier Knotenarten definiert worden. Es sind Verzweigungen (*switches*), Lokationsmodifikatoren (*location modifiers*), Signalisierungsoperationen (*signalling operations*) und Nicht-Signalisierungsoperationen (*non-signalling operations*).

Zwei verschiedene Arten von Rufverarbeitungsaktionen (*call processing actions*) werden unterschieden. *Top-Level Actions* werden ausgelöst, wenn ein Signalisierungsereignis (*event*) in einem Telefonieserver eintritt. Ein solches Ereignis kann eine eingehende bzw. eine ausgehende Rufaufbaunachricht, welche am Server ankommt, sein. Es wird jeweils das Skript ausgeführt, dessen Besitzer das Ziel bzw. der Initiator des Rufereignisses ist. *Subactions* sind Aktionen, die indirekt von anderen aufgerufen werden können. Ein Beispiel für ein CPL-Skript ist in Listing 2.1 dargestellt. Die graphische Repräsentation des Skriptes ist in Abbildung 2.12 gegeben.

Listing 2.1: Textuelle Repräsentation eines CPL-Skripts einer Voicemail-Anwendung. Das Skript verzweigt in Abhängigkeit der Adresse des Anrufers zu verschiedenen Aktionen

```
<?xml version="1.0" encoding="UTF-8"?>
<cpl xmlns="urn:ietf:params:xml:ns:cpl"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:cpl␣cpl.xsd␣">
  <incoming>
    <address-switch field="origin" subfield="host">
      <address subdomain-of="kom.de">
```

```
        <location url="sip:mgoertz@kom.eu">
          <proxy timeout="10">
            <busy> <sub ref="voicemail" /> </busy>
            <noanswer> <sub ref="voicemail" /> </noanswer>
            <failure> <sub ref="voicemail" /> </failure>
          </proxy>
        </location>
      </address>
      <otherwise>
        <sub ref="voicemail" />
      </otherwise>
    </address-switch>
  </incoming>
  <subaction id="voicemail">
    <location url="sip:mgoertz@voicemail.kom.eu">
      <redirect />
    </location>
  </subaction>
</cpl>
```

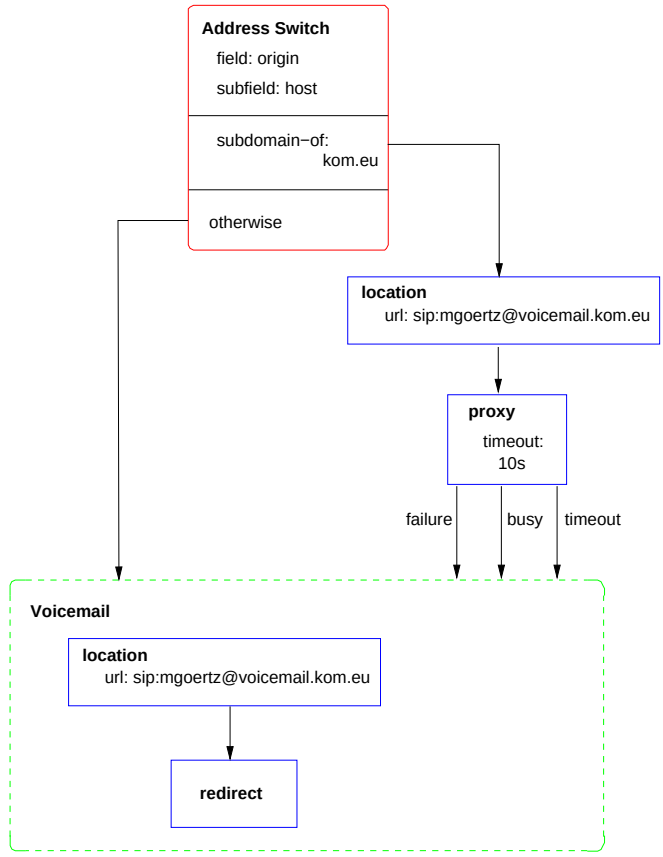


Abbildung 2.12: Graphische Repräsentation eines CPL-Skripts (aus Listing 2.1 mit einer Voicemail-Anwendung

CPL verwendet XML (eXtensible Markup Language) [BPSM00] zur Notation der Sprachsyntax. XML ist eine Untermenge der Markup-Sprache Standard Generalized Markup Language (SGML) [Int86]. Ein XML-Dokument besteht aus einer hierarchischen Struktur von *Tags*. Jedes Tag besitzt eine Reihe an Parametern. Ein XML-Dokument beschreibt nur die *strukturelle* Anordnung. Eine semantische Zuordnung der einzelnen Tags wird erst durch die Definition eines *XML-Schemas* [Fal00] möglich. Die Verwendung von XML zur Notation der CPL-Skripte erlaubt die Verwendung einer großen Zahl von verfügbaren Werkzeugen für die Erstellung, Bearbeitung und Validierung von XML-Dokumenten.

Die Syntax der Call Processing Language wurde ursprünglich durch eine XML Document Type Definintion (DTD) beschrieben, die mittlerweile durch ein XML-Schema ersetzt worden ist. Hierin sind die Knoten und Ausgänge eines CPL-Skripts als XML-Tags definiert. Die Parameter, welche diese näher beschreiben, werden als Tag-Attribute repräsentiert.

Die Reihenfolge und die Beziehung zwischen den einzelnen Knoten wird durch die Schachtelung der einzelnen Knoten-Tags ausgedrückt. Dabei umfasst der Knoten, der näher an der Wurzel des Graphens ist, seine Nachfolgeknoten. Besitzt ein Knoten mehrere Eingänge, so muss dieser als Makro in Form einer *Subaction* definiert werden. Jede Subaction besteht aus einem Tag zur Definition (`<subaction id=''label''>`) und einem Tag, mit dem diese referenziert (`sub ref=''label''`) werden kann. Dieser `sub`-Tag ist ein *Pseudoknoten*, der anstelle eines echten Knotens im Graphen plaziert werden kann. Der Output des Knotens ist innerhalb der Subaction definiert. Um Rekursionen und Schleifen zu verhindern, darf aus einer Subaction keine Top-Level Action aufgerufen werden. Darüber hinaus dürfen nur Subactions referenziert werden, die vorher definiert worden sind.

2.4 Zusammenfassung

Der Themenkomplex der Dienste wurde eingeführt und in einer Taxonomie gegliedert. Der Schwerpunkt des Kapitels wurde auf Kommunikationsdienste gelegt. Für diese wurde ein generisches Modell entworfen und die Eigenschaften der Diensterstellung betrachtet. Anschließend wurde auf Telefonie als ein technisch und ökonomisch anspruchsvoller Echtzeit-Kommunikationsdienst fokussiert.

IP-Telefonie wurde als junge Technologie für eine Realisierung eines Telefoniedienstes für IP-Netze ausgewählt. Die Konzepte, Signalisierungsprotokolle und Komponenten wurden detailliert vorgestellt. Die weitere Betrachtung wurde auf des Signalisierungsprotokoll SIP beschränkt. Für SIP wurden verschiedene Verfahren im Bereich des Service Engineering identifiziert und betrachtet. Diese stellen die Ausgangsbasis für die Erstellung von Kontext-bewussten Kommunikationsdiensten dar.

Kapitel 3

Effiziente Echtzeitkommunikation

Ein Problem lösen heißt, sich vom Problem lösen.

Johann Wolfgang v. Goethe

In Kapitel 1 wurde bereits die Problematik heutiger Kommunikationsanforderungen identifiziert und beschrieben. Dieses Kapitel beschreibt nun den gewählten Ansatz zur Erstellung von nutzerzentrierten Kommunikationsdiensten, wie sie in Kapitel 2 eingeführt worden sind, zur Verbesserung der Effizienz hinsichtlich der Kommunikationskontrolle und Erreichbarkeit. Eine Betrachtung und Definition der Effizienz in diesem Kontext wird durchgeführt. Die sich daraus ergebenden Möglichkeiten zur Effizienzsteigerung werden analysiert und zur weiteren Ausarbeitung des Ansatzes einbezogen.
Drei prinzipielle Verfahren zur Steigerung der Effizienz werden diskutiert. Als Schlussfolgerung aus den gewonnen Erkenntnissen wird die Verwendung von Kontexten zur Parametrisierung von Diensten eingeführt. Aus bisherigen Dienstmodellen wird ein erweitertes Dienstmodell für Kontext-bewusste Kommunikationsdienste abgeleitet. Eine notwendige Unterstützung zur Erstellung dieser Dienste wird in den nachfolgenden Kapiteln ausführlich entworfen.

Kurzübersicht

Die Struktur des vorliegende Kapitels ist wie folgt: Aus der Motivation und Problemanalyse in Abschnitt 3.1 wird in Abschnitt 3.2 die Definition von Effizienz, wie sie in dieser Arbeit verstanden wird, eingeführt. Aus den Faktoren, die diese limitieren, werden diejenigen identifiziert, die sich mit technischen Mitteln verändern lassen.

Drei grundlegende Verfahren, welche eine wahrnehmbare Steigerung der Effizienz durch die Verlagerung des Aufwands vom Nutzer zum Dienst erreichen, werden in Abschnitt 3.3 detailliert behandelt. Als prinzipieller Ansatz wird aus der Erkenntnis, dass Kontext

in der menschlichen Kommunikation eine überaus wichtige Rolle spielt, ein technisch erweitertes Modell für Kontext-bewusste Kommunikationsdienste in Abschnitt 3.4 hergeleitet. Abbildung 3.1 visualisiert die Struktur und die Beziehung zwischen den Abschnitten des Kapitels.

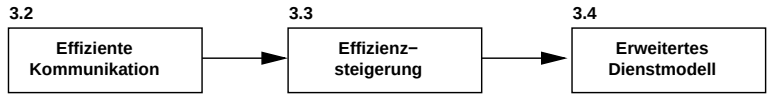
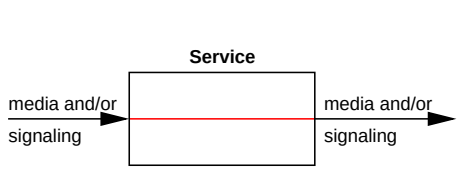


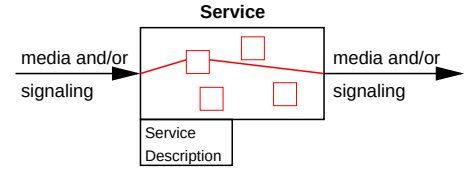
Abbildung 3.1: Strukturübersicht von Kapitel 3

3.1 Problemanalyse

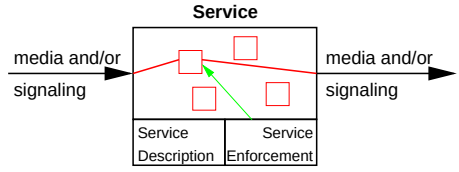
Der stetig wachsende Bedarf nach Kommunikation zu jeder Zeit, an jedem Ort und mit unterschiedlichen Medien führte und führt zu einer fortwährenden Verbesserung der Kommunikationsgeräte und -dienste. Anhand des *Telefoniedienstes* wird nachfolgend die Entwicklung von Kommunikationsdiensten (in Abbildung 3.2(a) dargestellt) aufgezeigt. Zur Verdeutlichung wird auf das Dienstmodell aus Abbildung 2.2 zurückgegriffen und dieses erweitert. Die Trennung zwischen den einzelnen Gruppen ist jedoch nicht strikt. Funktionalitäten aus einer zeitlich jüngeren Dienstgruppe sind teilweise auch für ältere Gruppen verfügbar.



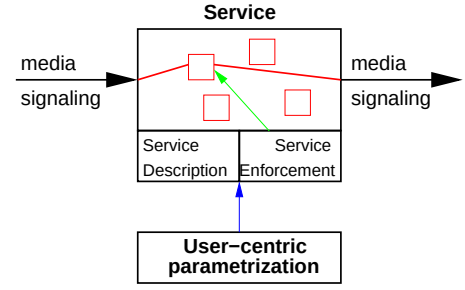
(a) POTS Dienst



(b) Intelligent Network Dienst



(c) Mobilkommunikationsdienst



(d) Nutzerzentrierter Dienst

Abbildung 3.2: Evolution von Kommunikationsdiensten anhand von Telefoniediensten

Abbildung 3.2(a) zeigt den klassischen Telefoniedienst auch *Plain Old Telephone Service* (POTS) genannt. Dieser Dienst erfüllt die Funktionalität des Aufbaus einer Verbindung zwischen zwei Gesprächspartnern. Ein Signalisierungs- und Medienpfad wird geschaltet – symbolisiert durch die durchgezogene Linie. Die Interaktion zwischen dem Endgerät und dem Nutzer beschränkt sich auf ein akustisches Signal bei einem Rufeingang. Bei diesem Dienst hat der Nutzer die Möglichkeit, den Ruf anzunehmen (*off-hook*) oder nicht anzunehmen. Eine bestehende Verbindung kann mittels Auflegen (*on-hook*) beendet werden.

Dienste aus dem ISDN oder IN sind schematisch in Abbildung 3.2(b) dargestellt. Eine Reihe von Mehrwertdiensten steht zur Auswahl (hier als Boxen dargestellt). Der Signalisierungspfad und eventuell auch der Medienpfad können einen solchen Dienst traversieren. Über die Anzeige von Rufnummern bzw. eines dazugehörigen im geräteeigenen Telefonbuch eingespeicherten Namens kann der Nutzer entscheiden, ob er den Anruf entgegennehmen möchte oder nicht. Die Dienstbeschreibung (*service description*) spezifiziert jeden zugelassenen Mehrwertdienst in einer formalen Notation. Der Nutzer hat die Wahl, einen Mehrwertdienst ein- oder auszuschalten. Eine minimale Parametrisierung, wie die Eingabe des Ziels einer Anrufumleitung erlaubt es, den Dienst an den Nutzer anzupassen.

Mobiltelefone verfügen in der Regel über ein eingebautes Telefon- und Adressbuch. Darüber hinaus lassen sich die Einträge verschiedenen Anrufgruppen zuordnen. Diese Zuordnung kann zur Parametrisierung der auf einem Mobiltelefon vorhandenen Dienste genutzt werden. So können in Abhängigkeit der Zugehörigkeit eines Anrufers in eine Anrufgruppe verschiedene Ruftöne abgespielt werden. Auch können zur selben Zeit mehrere Dienste parallel aktiv sein, deren Ausführung abhängig von einem Anrufer ist. Dies wird in Abbildung 3.2(c) mit Dienstforcierung (*service enforcement*) bezeichnet. Zusätzlich lassen sich Mobiltelefone und die Dienste mittels Profilen parametrisieren.

Einen wesentlichen Nachteil besitzen alle drei bisher aufgeführten Klassen von Kommunikationsdiensten. Jeder Dienst bzw. Mehrwertdienst muss *manuell* und explizit konfiguriert, aktiviert und deaktiviert werden. Eine Anpassung an die gegenwärtige Umgebung und den Kontext des Nutzers findet nicht statt. Als nächste Stufe der Kommunikationsdienste wird daher eine weitere Zentrierung auf den Nutzer erwartet. Die Dienste sollen und müssen sich stärker an dem tatsächlichen Bedarf des Nutzers anpassen lassen. Eine solche *nutzerzentrierte* Parametrisierung (*user-centric parametrization*) ist als Funktionsblock in Abbildung 3.2(d) dargestellt. Die Parametrisierung beeinflusst sowohl den Ablauf des Dienstes als auch die Bedingungen, wann ein bestimmter Dienst ausgeführt werden soll. Die nachfolgend vorgeschlagenen Kontext-bewussten Kommunikationsdienste stellen eine mögliche Ausprägung eines nutzerzentrierten Dienstes dar. Diese Dienste repräsentieren eine neuartige Dienstklasse, die als unsichtbare und nicht mehr im aktiven Wahrnehmungsbereich des Nutzers befindliche Helfer bei der effektiven Bewältigung von Kommunikationsaufgaben fungieren.

3.2 Effiziente Kommunikation durch Einbeziehung von Nutzerkontexten

Die in der Motivation beschriebene gegenwärtige Situation bei der Kommunikation erfordert eine Steigerung der *Effizienz* in Bezug auf die *Erreichbarkeit* eines Gesprächspartners in einer interaktiven Echtzeitkommunikation.

3.2.1 Effizienzbetrachtung

Nach [Deu00] wird Effizienz wie folgt definiert:

Definition 3.1 (Effizienz)
*Die **Effizienz** ist das Verhältnis zwischen dem erreichten Ergebnis und den eingesetzten Ressourcen.*

Im Rahmen der Arbeit wird die Effizienz η als Ratio zwischen dem Aufwand für erfolgreiche Anrufversuche im Verhältnis zum Gesamtaufwand an Anrufversuchen verstanden.

$$\eta = \frac{A_{\text{erfolgreich}}}{A_{\text{erfolgreich}} + A_{\text{nicht-erfolgreich}}} \tag{3.1}$$

Der Aufbau der Formel (3.1) ähnelt der vom dänischen Mathematiker Agner Krarup Erlang Anfang des 20. Jahrhunderts aufgestellten Erlang-C Formel. Diese beschreibt eine spezielle Form der Erlang-Verteilung, die die Wahrscheinlichkeiten von Wartezeiten bei der Vermittlung von Telefongespräch beschreibt. Die Erlang-C Formel in ihrer generischen Form ist in Gleichung (3.2) dargestellt.

$$a = \frac{N \cdot \bar{t}_{\text{bearb.}}}{t} \tag{3.2}$$

wobei a das Arbeitsvolumen, N die Anzahl der Anrufe und $\bar{t}_{\text{bearb.}}$ die durchschnittliche Bearbeitungszeit bezeichnen.

Jeder der Anrufversuche N, die ein Nutzer zur Erreichung seines Gesprächspartners aufbringt, ist mit einem bestimmten *Aufwand* verbunden. Unter diesem subsumieren sich unterschiedliche Faktoren wie Zeitspannen, Ressourcen und Kosten. Der Aufwand A kann aufgeteilt werden in einen Nutzeranteil und einen Systemanteil, geschrieben als

$$A = A_{\text{User}} + A_{\text{System}} \tag{3.3}$$

Der Anteil A_{User} beschreibt den vom Nutzer *wahrnehmbaren* Aufwand, den er aufbringen muss. Dies kann beispielsweise das Suchen, das Eingeben der Nummer oder das Warten auf eine Antwort vom Angerufenen sein. Der Nutzeraufwand kann noch weiter unterteilt werden in die Anteile der bei der Kommunikationssteuerung beteiligten Personen, wie dies in Gleichung (3.4) gezeigt ist.

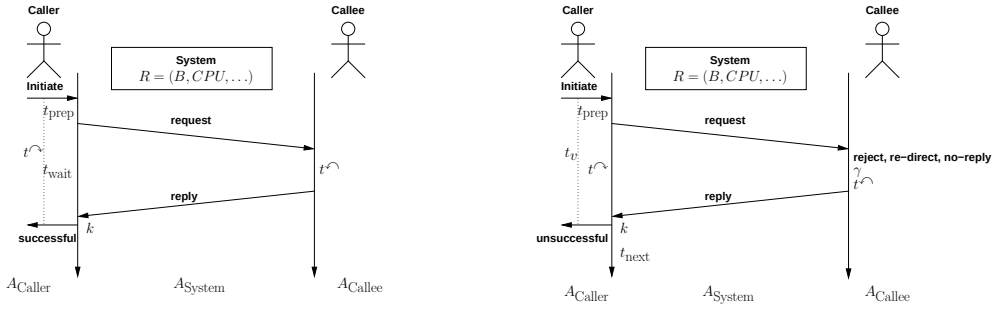
$$A_{\text{User}} = A_{\text{Caller}} + A_{\text{Callee}} \tag{3.4}$$

Dadurch kann ausgedrückt werden, dass auch der Angerufene in den erfolgreichen Aufbau der Kommunikationsbeziehung eingebunden ist. So zum Beispiel durch von ihm getätigte Rückrufe aufgrund einer Anzeige in seiner „Entgangene Anrufe“-Liste. Weiterer zeitlicher Aufwand und eventuell auch Kosten entsteht dem Angerufenen, wenn er nicht der eigentliche Ansprechpartner für das Gespräch ist, und dieses weiterleiten muss. Der Anruf kann den Angerufenen auch stören, wenn dieser zu einem ungünstigen Zeitpunkt erfolgt. Diese Störung wird mit dem *Störfaktor* (*disturbance factor*) γ bezeichnet.

Der Aufwand A_{System}, der vom System erbracht wird, ist zum Beispiel der des Routing der Signalisierung und der Bereitstellung von Diensten wie *Wahlwiederholung* oder *Rückruf bei Besetzt*. Dies schlägt sich in verbrauchten Ressourcen wie belegter Bandbreite oder erbrachter Prozessorleistung wieder.

$$A_{\text{System}} = R(B, C, \ldots) \tag{3.5}$$

Die Abbildungen 3.3(a) und 3.3(b) zeigen für die Fälle „erfolgreicher“ und „nicht-erfolgreicher“ Anruf die Verteilung der Aufwände und die einzelnen Faktoren, aus denen sie zusammengesetzt sind.



(a) Erfolgreicher Verbindungsaufbau

(b) Nicht-erfolgreicher Verbindungsaufbau

Abbildung 3.3: Aufwände für den Verbindungsaufbau

Die Zeitspanne $t^{\curvearrowright}$ ist definiert als die Zeit ab dem Vorgang der Initiierung eines Anrufs und dem Zeitpunkt eines erfolgreichen Verbindungsaufbaus oder der Terminierung des Verbindungsaufbaus. Die Zeitspanne $t^{\curvearrowright}$ ist in die beiden Zeiten t_{prep} für die Vorbereitungsphase (*preparation*), bevor die Signalisierung beginnt, und in die Zeit t_{wait} , in welcher der Nutzer auf eine Antwort vom Angerufenen wartet, unterteilt. In der Vorbereitungsphase wird die Zeit erfasst, in der der Nutzer beispielsweise nach der Nummer sucht oder den Wählvorgang abschließt. Die Zeit, die auf der angerufenen Seite vergeht, bevor eine Antwort an den Anrufer zurückgesendet wird, ist mit $t^{\curvearrowleft}$ bezeichnet.

Die Zeitspanne der einzelnen Anrufversuche variiert zwischen den einzelnen Versuchen und ist abhängig vom Verhalten der beiden Teilnehmer, der Dringlichkeit des Anrufs

und den den Nutzern zur Verfügung stehenden Diensten. Aus diesem Grund wird die Zeitspanne als Mittelwert aller Versuche mit $\bar{t}_v$ angegeben.

Zusätzlich werden, je nach Verlauf, für den Gesprächsaufbauversuch Kosten in Form von Gesprächsgebühren oder für die Nutzung eines bestimmten Dienstes fällig. So erreicht man unter Umständen nicht den gewünschten Teilnehmer, sondern einen Kollegen oder die Zentrale. Da auch die Kosten variieren, wird dieser Faktor ebenfalls als Mittelwert $\bar{k}_v$ angegeben. Werden „reine“ IP-Telefonie-Gespräche betrachtet, so fällt der Kostenpunkt weniger ins Gewicht, da hierbei meist eine andere Tarifierung vorliegt, in der das Volumen des Verkehrs schon enthalten ist, wie z. B. Flatrate-Tarife.

So ergibt sich der Aufwand aus den Kosten und der eingesetzten Zeit, wobei der Operator '+' in Gleichung (3.6) keine nummerische Operation darstellt[1]. Es handelt sich hierbei um eine qualitative Betrachtung der Aufwände und nicht um eine Berechnungsvorschrift.

$$A = A_{\text{System}} + A_{\text{Caller}} + A_{\text{Callee}} \tag{3.6}$$

$$= \underbrace{r + (t^{\curvearrowright} + k) + t^{\curvearrowleft}}_{\text{erfolgreich}} + \underbrace{\sum_{i}^{N-1} r_i + (t_i^{\curvearrowright} + k_i) + (t_i^{\curvearrowleft} + \gamma_i)}_{\text{nicht-erfolgreich}} \tag{3.7}$$

$$\approx N \cdot (\bar{r} + \bar{t} + \bar{k} + \bar{\gamma}) \tag{3.8}$$

$$= N \cdot (R + T + K + \Gamma) \tag{3.9}$$

Ein Ziel dieser Arbeit ist es, die vom Nutzer *wahrnehmbaren Aufwände* für die Kontrolle der Kommunikation so zu senken, dass sich für den Nutzer eine Steigerung der Effizienz in Bezug auf die Kommunikationsbeziehung ergibt.

Um die Ratio η aus Gleichung (3.1) zu verbessern, können die Parameter N, T, K und Γ jeweils verändert werden. Welche Auswirkungen die Veränderung auf den Wert η hat, ist nachfolgend beschrieben. Eine weitere Reduktion kann durch eine Verschiebung von A_{User} zu A_{System} erreicht werden.

Kosten: Eine Reduktion der Telefonkosten ist prinzipiell möglich, beispielsweise durch Wahl eines anderen Providers oder eines anderen Tarifmodells. Jedoch sind die zu erwartenden Einsparungen marginal, insbesondere bei IP-Telefoniegesprächen.

Zeitspanne: Die benötigte Zeitspanne für den Versuch eines Verbindungsaufbaus variiert stark und lässt sich im Allgemeinen nur schwer reduzieren. Ein Nutzer könnte schon nach wenigen Freizeichen auflegen, was allerdings die Chancen auf ein erfolgreiches Erreichen des Gesprächspartners verringert.

Anzahl der Versuche: Eine Verbesserung der Ratio kann durch die Verringerung der nicht-erfolgreichen Versuche oder durch die Erhöhung der erfolgreichen Rufaufbauversuche erreicht werden. Dies gilt insbesondere, wenn nur die vom Nutzer

[1] Durch Einführen von Faktoren können die unterschiedlichen Einheiten in einander umgerechnet werden. So können monetäre Kosten in Zeit und umgekehrt überführt und additiv verrechnet werden.

manuell durchgeführten Versuche gezählt werden. Durch den Einsatz von Kommunikationsdiensten kann N reduziert werden.

Verschiebung zum Systemaufwand: Zur Steigerung der Effizienz wird der Ansatz der Reduktion der manuell durchzuführenden Anrufversuche verfolgt. Es findet eine Verschiebung der wahrnehmbaren Aufwände vom Nutzer zum System statt. Das System erbringt die Funktionalität, die sonst vom Nutzer geleistet werden müsste.

Verringerung der störenden Anrufe: Eine wahrnehmbare Verbesserung der Kommunikation kann erreicht werden, wenn die Zahl der zwar vermittelten, aber letztlich störenden Anrufe vermindert werden kann. Dies führt insbesondere beim Angerufenen zu einer Entlastung der Kommunikationssteuerung. Aber auch für den Anrufer bedeutet es eine Verbesserung, wenn er effektiver die Personen erreicht, die er erreichen möchte.

Drei aus der Beobachtung des menschlichen Verhaltens abgeleitete Konzepte werden im Detail in Abschnitt 3.3 behandelt. Jedes dieser Verfahren verändert besonders einen der aufgeführten Parameter. Zur Realisierung wurden in dieser Arbeit Kommunikationsdienste und notwendige weitere Verfahren entwickelt, mittels derer eine solche Verlagerung und damit Steigerung der Effizienz erreicht werden kann.

3.3 Verfahren zur Steigerung der Effizienz der Kommunikationsbeziehungen

Zwei prinzipielle Ansätze sind im Rahmen dieser Arbeit identifiziert und entwickelt worden. Zum einen die *Filterung* von Kommunikationsanfragen und zum anderen die *Verteilung des Kontexts* zur Vermeidung von Kommunikationsanfragen. Darüber hinaus wurde ein Verfahren erarbeitet und implementiert, das die Erreichbarkeit durch Auffinden von gemeinsamen Zeitschlitzen steigert. Hierzu werden *Kommunikationsbroker* eingesetzt, welche die Aushandlung im Namen der Nutzer durchführen. Diese Konzepte werden nachfolgend dargestellt.

3.3.1 Kontextinformationen zur Filterung von eingehenden Kommunikationsanfragen

Eine Vielzahl an eingehenden Kommunikationswünschen müssen heute von Nutzern entgegengenommen und entsprechend behandelt werden. Ein wichtiger Dienst ist, wie auch Tabelle 1.1 zu entnehmen ist, die Anrufweiterleitung und Anrufumleitung. Der Nutzer muss hierbei den Dienst explizit aktivieren und das Umleitungsziel festlegen. Wird der Dienst nicht mehr benötigt, so muss er z. B. nach Ende einer Besprechung wieder explizit und manuell deaktiviert werden. Eine Verbesserung ergibt sich, wenn die Schritte der Aktivierung, Parametrisierung und Deaktivierung der Dienste in einem automatisierten und dem Kontext des Nutzers angepassten Vorgang erfolgen können.

Einen Beitrag zur Lösung des Problems der Steuerung eingehender Kommunikationsanfragen liefert der nachfolgend beschriebene Ansatz der *Filterung* unter Einbeziehung von Kontextinformationen.

Ein repräsentatives Szenario ist in Abbildung 3.4 abgebildet. Ein Nutzer möchte während einer wichtigen Besprechung nicht gestört werden. Eingehende Anrufe (z. B. von Caller X), die während der Besprechung an seinem Telefonsystem ankommen, sollen an seinen Kollegen, der nicht an der Besprechung teilnimmt, weitergeleitet werden. Ausnahmen von dieser Regel sind Anrufe von seinem Vorgesetzten (z. B. Boss) und wichtigen, die Besprechung betreffenden Kunden. Anrufe von diesen Anrufern sollen auf sein Mobiltelefon umgeleitet werden. Nach Beendigung der Besprechung sollen alle eingehenden Anrufe wieder auf seinem Arbeitsplatz-Telefon ankommen. Der entsprechende Dienst der hierfür eingesetzt werden kann, ist die Erweiterung des Dienstes *Anrufumleitung* zum Dienst *Kontext-bewusstes Anrufumleiten*.

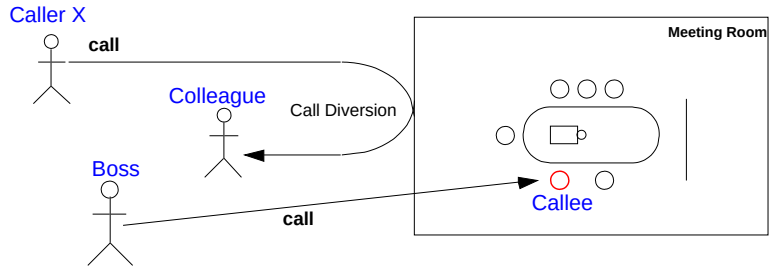


Abbildung 3.4: Kontext-bewusster Anrufumleiten-Dienst

Die zwei wesentlichen Merkmale des *Kontext-bewussten Anrufumleiten* Dienstes sind die automatisierte Feststellung des aktuellen Kontexts des Nutzers und ein Regelsystem, das es dem Nutzer erlaubt, seine Kommunikationsabläufe zu spezifizieren. Für die Bestimmung des Kontexts kann die in Kapitel 7 beschriebene Kontextaggregationsarchitektur verwendet werden. Diese liefert den aktuellen Kontext durch Einbeziehung und anschließender Aggregation von Lokationsdaten, Kalenderinformationen und symbolische Informationen zu Objekten (z. B. durch Infrared-Baken). Zur Spezifizierung der Kommunikationsabläufe kann der Nutzer auf einen speziellen Editor, der in Abschnitt 8.3 beschrieben ist, zurückgreifen. Dieser ermöglicht die Erstellung von CPL-Skripten, die den Kontext des Nutzers als Bedingung für die weitere Verzweigung des Kommunikationsablaufs nutzen.

Eine Alternative zur Verwendung eines CPL-Servers sowie von CPL-Skripten ist die Änderung des Anrufumleitungsdienstes im Endgerät. Hierzu wird der interne Ablauf des Dienstes derart geändert, dass eine Kontextquelle abgefragt werden und der erhaltene Kontext als zusätzliche Bedingung für die Weiterleitung einbezogen werden kann. Die Details für die Umsetzung des Konzeptes in einem SIP-basierten User Agents sind in Unterabschnitt 9.5.1 beschrieben. Die notwendigen Erweiterungen der CPL-Engine, des CPL-Editors und der CPL-Syntax sind in Abschnitt 8.2 ausgeführt.

3.3.2 Verteilung von Kontextinformationen zur Vermeidung unnötiger Kommunikation

Die Kommunikation zwischen Menschen kann unterschieden werden in *direkte* Kommunikation von Angesicht zu Angesicht (*face-to-face*) und Kommunikation zwischen *entfernten* Gesprächspartnern (*distant*). Die Kommunikationstheorie zeigt, dass Kontext ein essentieller und immanenter Bestandteil sprachlicher Interaktion zwischen Menschen ist. Kontext bezieht sich hierbei sowohl auf den situativen Umgebungskontext als auch auf den inhaltlichen Kontext des Gesprächs. Während eines Gesprächs wird ein solcher Kontext aufgebaut und benutzt, um Mehrdeutigkeiten aufzulösen[Aue86]. Auch kann der äußere Umgebungskontext einen Einfluss auf die Art und den Inhalt der Konversation haben.

Neben der näheren Erklärung des Inhalts spielt der Kontext eine wichtige Rolle beim Start einer vis-à-vis Kommunikation. Ein Gesprächspartner beobachtet den äußeren Kontext eines potenziellen Gesprächspartners und zieht daraus Schlüsse für die Konversation. Eine Vielzahl an Kriterien wird hierfür berücksichtigt. Die sozialen Rollen der Gesprächspartner, der Sinn und Inhalt der angestrebten Konversation und die gegenwärtige Situation sind repräsentative Beispiele dieser Kriterien. Das Verhalten, den Kontext zu erfassen, ist ein Teil der Erziehung und wird von Kindheit an trainiert. Regeln, die aufgrund des wahrgenommenen Kontexts befolgt werden, sind in den gesellschaftlichen Kontext eingebettet und werden dementsprechend befolgt. Mittels dieser „Technik“ besitzen Menschen in der Regel ein gutes Gespür, wann es geeignet erscheint, die Konversation zu beginnen.

Bei einer zwischenmenschlichen Kommunikation zwischen entfernten Gesprächspartnern kann diese Methode nicht angewendet werden. Gegenwärtig existiert keine technische Umsetzung, die eine Beobachtung des Kontexts eines adressierten Gesprächspartners erlaubt. Ein Anrufer versucht aus seiner Erfahrung und der Einschätzung des Gesprächspartners auf dessen Kontext zu schließen. Während früher der Tagesablauf eines Angestellten im Büroumfeld relativ statisch war, ist er heute nomadischen Arbeitsanforderungen gewichen. Dementsprechend ist auch die Antizipation des Kontexts des Gegenübers schwieriger geworden. Dennoch wird in der Regel vor jedem Gespräch implizit versucht, diesen zu bestimmen.

Eine genaue Kenntnis des Kontexts würde unnötige Kommunikationsanfragen verringern und damit eine wahrnehmbare Verbesserung und Effizienzsteigerung der Kommunikation bedeuten. Die vorgeschlagene Methode der *Verteilung von Kontextinformationen* (*context sharing*) zwischen Gesprächspartnern stellt ein Konzept dar, das die existierende Lücke zwischen der direkten und entfernten Kommunikation hinsichtlich der Kenntnis des Kontexts des Gesprächspartners zu schließen sucht. Dabei erhebt das entwickelte und umgesetzte System keinen Anspruch, die menschlichen über viele Jahrhunderte entwickelten Fähigkeiten vollständig abzubilden. Es wurden die für die Kommunikation in einem Büroumfeld notwendigen und wichtigen Mechanismen identifiziert und als Annäherung an ein möglichst gutes technisches Äquivalent umgesetzt.

Der prinzipielle Ablauf ist in Abbildung 3.5 für den Fall, dass der Gesprächspartner

in einer Besprechung (*meeting*) ist, dargestellt. Der Anrufer Caller möchte einen Gesprächspartner Callee anrufen. Während der Rufaufbauphase (*call setup*) wird der Kontext des Gesprächspartners mittels Signalisierungsnachrichten erfragt und übermittelt. Dabei wird keine explizit wahrnehmbare Anrufsignalisierung auf Seiten des Angerufenen ausgelöst. Dies ist wichtig, da der Angerufene unter Umständen nicht gestört werden möchte, eine akustische Signalisierung aber stören würde. Eine Alternative zu der automatischen Übertragung der Kontextinformationen ist die textuelle Anzeige einer Anfrage einer Kontextabfrage. Der Angerufene kann dann entscheiden, ob er die Informationen mitteilen möchte.

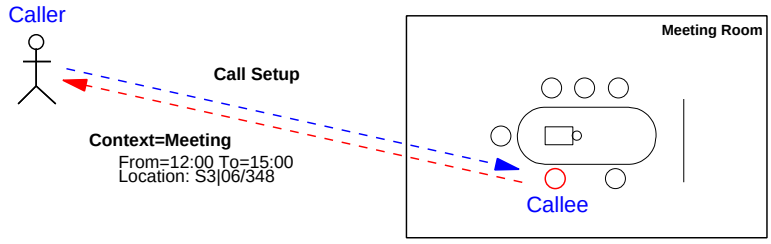


Abbildung 3.5: Mitteilen von Kontext zwischen Angerufenem und Anrufer

Auf Grund der übertragenen Kontextinformation kann der Anrufer den weiteren Kommunikationsablauf steuern. Hierbei wird die Verantwortung und Kontrolle vom Angerufenen auf den Anrufer verlagert. Es werden dabei die oben beschriebenen sozialen und kulturellen Verhaltensweisen vorausgesetzt, die ein für beide Parteien zufrieden stellendes weiteres Vorgehen sicherstellen sollen. Der Anrufer kann die Kontextinformationen nutzen, um diese entsprechend seiner Kommunikationserfahrungen, seiner Kenntnisse des Angerufenen sowie des Sinns und Inhalts des Anrufs zu interpretieren.

Verschiedene Varianten des Konzepts der Verteilung von Kontexten sind denkbar. Die in dieser Arbeit identifizierten und betrachteten Varianten unterstützen jeweils den Anrufer und den Angerufenen.

Vom Angerufenen initiiert: Der Nutzer initiiert den Sitzungsaufbau wie gewohnt, jedoch möchte der Angerufene seinen aktuellen Kontext dem Anrufer mitteilen. Dazu werden die Informationen in einer Antwort-Nachricht übermittelt. Diese werden dem Nutzer angezeigt, worauf dieser weitere Aktionen auswählt. Alternativ können sinnvolle weitere Aktionen auch vom Endgerät vorgeschlagen bzw. ausgeführt werden. Bei einem weiteren Sitzungsaufbau muss die Semantik des bisherigen Verlaufs erhalten bleiben, da sonst erneut die Kontexte mitgeteilt werden würden.

Vom Anrufer initiiert: Der Anrufer möchte vor einem Anruf wissen, in welchem Kontext sich der potentielle Gesprächspartner befindet, um entscheiden zu können, ob sich ein Anruf lohnt oder ob ein späterer Zeitpunkt oder ein anderes Kommunikationsmedium erfolgversprechender wären. Hierzu signalisiert der Anrufer eine Kontextabfrage, die auf der rufenden Seite keine Interaktion mit dem Nutzer

oder eine Signalisierung eines Anrufs auslöst. Ein weiterer Sitzungsaufbau soll zu dieser Sitzung zugeordnet werden können.

3.3.2.1 Verwandte Arbeiten

Ein initialer Ansatz zum Austausch von Kontextinformationen zwischen Mobiltelefonen mittels Wireless Application Protocol (WAP) wurde in [STM00] vorgestellt. Die wesentliche Limitierung des Ansatzes liegt in der Beschränkung auf eine kleine Anzahl von vordefinierten Antworten, die den Zustand des Angerufenen beschreiben. Die Zustände des Nutzers müssen in jeden Fall manuell eingegeben werden. Bei einer Änderung des Kontexts muss der Nutzer aktiv und explizit eingreifen. Ein Kontext-bewusster Dienst soll genau dies automatisch und für den Nutzer nicht wahrnehmbar erledigen. Die übermittelten Informationen werden dem Anrufer angezeigt und können nicht zu einer weiteren automatischen Steuerung des Kommunikationsablaufs verwendet werden. Auch hier muss der Nutzer immer aktiv handeln und die Eingaben tätigen.

3.3.3 Kommunikationsbroker zur Steigerung der Erreichbarkeit

Die vorgestellte Methode der Kontext-bewussten Filterung entlastet den Angerufenen von eingehenden Kommunikationsanfragen und leitet die Anfrage an einen (aus der Sicht des Angerufenen) Kommunikationsendpunkt weiter. Die Verteilung des Kontexts erlaubt es dem Anrufer, die Situation des entfernten Kommunikationspartners besser zu bewerten und die Anrufentscheidung darauf zu basieren. Beide Verfahren verringern die Anzahl an Aktionen, die notwendig sind, um den Gesprächspartner zu erreichen.

Eine deutliche Reduktion des Aufwands für den Nutzer ergibt sich im Alltag durch Delegation des Kommunikationsaufbaus an ein Sekretariat. Dies ist ein Vorgehen, das vielfach mit nachweisbarem Nutzen von Geschäftsleuten praktiziert wird. Hierbei äußert der Nutzer seinen Wunsch, einen bestimmten Gesprächspartner zu sprechen und wird bei einer erfolgreichen Verbindung darüber informiert, ob und unter welchen Umständen dies möglich ist. Für das weitere Vorgehen werden der aktuelle Kontext des Nutzers, seine Präferenzen und Erfahrungen einbezogen. Eventuell bestehen bereits Informationen über den Gesprächspartner und seine Präferenzen. Oftmals können diese Informationen vom Sekretariat des Gesprächspartners erfragt werden. An dieses kann die Kommunikationsanfrage weitergeleitet werden. Die beiden eigentlichen Gesprächspartner sind somit von der Bearbeitung vollständig entlastet und werden erst bei einem Gespräch wieder involviert.

Im Rahmen dieser Arbeit wird ein *Kommunikationsbroker* vorgeschlagen, der ein technisch äquivalentes Verfahren zur Verfügung stellen soll. Der Broker agiert in diesem Fall äquivalent zu einem Mitarbeiter im Sekretariat als Stellvertreter (*proxy*) für den Nutzer und vermittelt günstige Zeitfenster für eine Kommunikation. Dazu spezifizieren die Nutzer die Kommunikationsmedien und die Zeiten, in denen sie über diese erreichbar

sind. Dieser Zeitschlitz repräsentiert das Übereinkommen der beiden Kommunikationspartner und der gewählten Kommunikationsmedien. Die Wahrscheinlichkeit ist danach sehr hoch, dass eine Kommunikation in diesem Zeitschlitz zustande kommt.

Ein Szenario mit zwei Kommunikationsbrokern ist in Abbildung 3.6 dargestellt. Es zeigt, dass der Broker in den Kommunikationspfad des Nutzers eingebunden ist und damit Einfluss auf den Pfad nehmen kann. Der Nutzer hat zu verschiedenen Zeiten und an verschiedenen Orten einen unterschiedlichen Satz an Kommunikationsgeräten zur Verfügung; analog ändert sich sein Status einer Kommunikation gegenüber seinem Gesprächspartner. Diese Begebenheiten muss der Broker bei der Aushandlung einer Kommunikation in Betracht ziehen.

Zur Entscheidungsfindung hat der Broker Zugriff auf die Informationsquellen des Nutzers wie beispielsweise auf den elektronischen Kalender, ContextServer oder Präferenzen. Zusätzlich können Kommunikationsentitäten wie *Gateways* zur Umwandlung von Signalisierungsnachrichten oder *Splitter* zum Aufteilen der Kommunikationsanfrage auf unterschiedliche Endgeräte des Anzurufenden mit einbezogen werden.

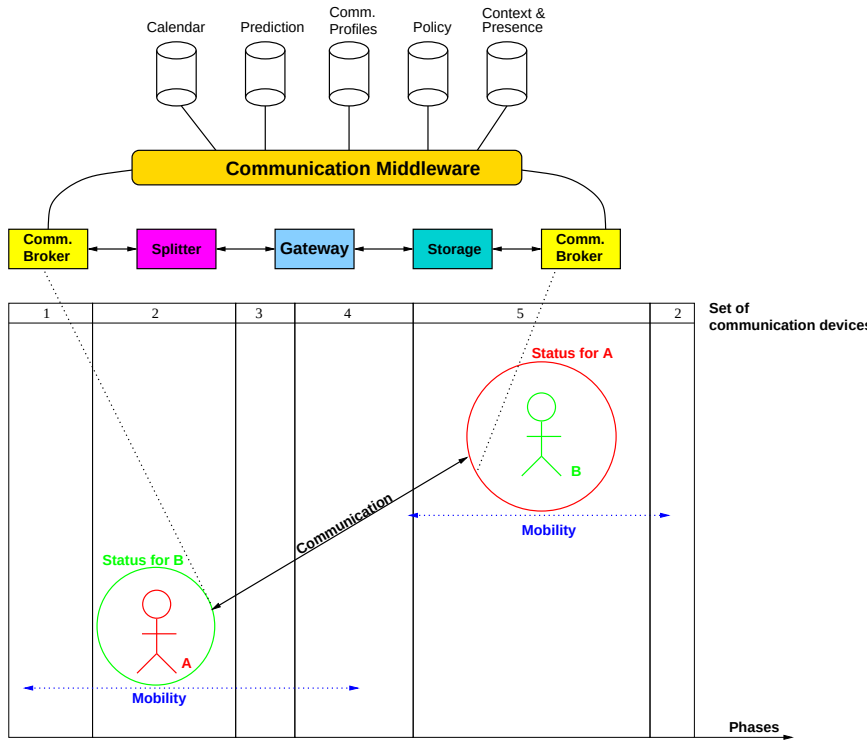


Abbildung 3.6: Kommunikationsbroker und sich ändernde Verfügbarkeit von Kommunikationsgeräten

Eine Realisierung eines hier vorgestellten Kommunikationsbrokers ist der in Unterabschnitt 8.4.1 eingeführten Digital Call Assistant. Dieser schlägt dem Nutzer Zeitpunkte für eine Kommunikation vor. Dazu spezifiziert der Nutzer mittels eines elektronischen Kalenders seine Zeitschlitze, in denen er für Kommunikation verfügbar ist sowie die

dazugehörigen Kommunikationstypen. Als zusätzliche Informationsquelle wird der aktuelle Kontext der Nutzer in die Bewertung der Situation mit einbezogen. Die Digital Call Assistants der beiden Gesprächspartner kommunizieren miteinander und tauschen dabei die Daten aus, die zur Findung gemeinsamer Zeitschlitze notwendig sind. Ein von beiden Seiten als günstig bewerteter Zeitpunkt führt zu einer starken Verringerung des Störungsfaktors γ.

3.3.4 Sicherheitsbetrachtungen

Die während der Erfassung von Kontextmerkmalen anfallenden Daten, wie sie zur Bestimmung des Kontexts des Nutzers benötigt werden, sind bereits sensible und oftmals auch personenbezogene Informationen. Der Kontext ist in der Regel noch aussagekräftiger und daher als noch sensibler einzustufen. Daher haben Nutzer ein berechtigtes Interesse, diese Daten vor der Einsicht von Fremden zu schützen. Datenschutzrechlichte Fragen wie z. B. wer wann welche Daten wo speichern kann, müssen geklärt sein, damit die vorgestellten Verfahren Akzeptanz finden.

Die Bereitstellung von Mechanismen zur Gewährleistung der *Sicherheit* der Daten sowie dem Schutz der *Privatsphäre* in einem dynamischen Umfeld mit vielen (auch unbekannten) Parteien stellen eine wissenschaftliche Herausforderung und unabdingbare Vorbedingung für den Einsatz in Unternehmen und in einer breiten Nutzerschicht dar.

Für den Austausch der Daten sollen kryptographische Verfahren zur Verschlüsselung und zur Integritätssicherung verwendet werden. Eine Authorisierung der Partner, die Kontexte abfragen oder an die Kontexte verteilt werden, stellt ein weiteres Sicherheitskonzept dar. Zusätzlich soll in einem *Stufenmodell* der Nutzer die Möglichkeit an die Hand bekommen, die Kontexte in abgestuften Detailgraden an andere Teilnehmer zu versenden. So können authorisierte Nutzer Information mit mehr Details bekommen (z. B. „Meeting, von 13:00-15:00 Uhr, in Raum S3|06/348“). Andere Nutzer hingegen erhalten nur die Basisinformationen, wie „Nicht verfügbar“.

3.4 Erweitertes Dienstmodell für Kontext-bewusste Kommunikationsdienste

Die in Unterabschnitt 3.3.1 und Unterabschnitt 3.3.2 beschriebenen Methoden zur Adaptierung der Kommunikation an den gegenwärtigen Kontext der Nutzer erfordert eine Veränderung der beteiligten Kommunikationsdienste. Diese neue Klasse an Kommunikationsdiensten werden in dieser Arbeit *Kontext-bewusste Kommunikationsdienste* genannt und folgendermaßen definiert:

Definition 3.2 (Kontext-bewusster Kommunikationsdienst)
*Ein **Kontext-bewusster Kommunikationsdienst** ist ein Kommunikationsdienst, dessen funktionaler Ablauf durch die Einbeziehung von Kontexten aktiv oder passiv*

verändert wird. Dies erfolgt mit dem Ziel, die erbrachte Funktionalität an den Kontext (des Nutzer) anzupassen oder einen neuen Dienst auszulösen.

Eine schematische Darstellung eines Kontext-bewussten Dienstes aus Definition 3.2 ist in Abbildung 3.7 dargestellt. Diese Dienstklasse ist eine Ausprägung des *Nutzerzentrierten* (*User-centric*) Dienstes aus Abbildung 3.2(d). Der Kontext wird in diesem Modell als Parameter angesehen, der den Dienst derart verändern kann, dass seine Funktionsweise an die Wünsche des Nutzers adaptiert werden kann.

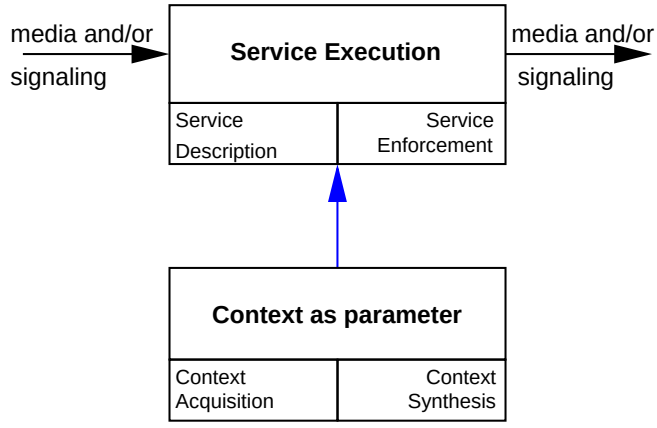


Abbildung 3.7: Erweitertes Dienstmodell mit Kontextinformationen zur Parametrisierung des Dienstes

Aus der Beschreibung gehen einige Anforderungen an einen Kontext-bewussten Kommunikationsdienst hervor, die nachfolgend aufgeführt sind.

- Der Kontext-bewusste Kommunikationsdienst soll den Nutzen für den Nutzer erhöhen. Diese Anforderung kann im Gegensatz zu den Forderungen stehen, die ein Provider erreichen möchte. Die Verbesserung des Nutzens wird aus der Sicht des Nutzers betrachtet und nicht vom Standpunkt des Providers.

- Der Dienst muss über geeignete Methoden verfügen, einen Kontext zu erfassen oder zu erfragen.

- Der Ablauf des Kommunikationsdienstes muss sich durch die Einbeziehung des Kontexts verändern lassen. Zwei prinzipielle Modi können unterschieden werden:

  - Die bereitgestellte Funktionalität des Dienstes adaptiert sich an den Kontext und führt zu einem angepassten Ergebnis.

  - Der Dienst aktiviert auf Grund der Bewertung des Kontexts einen weiteren Dienst.

- Zwei Arten von Kommunikationsdiensten sollen zur Erbringung von Kontext-bewussten Kommunikationsdiensten verwendet werden können:

- *Endgerätedienste* erbringen ihre Funktionalität in der Entität, die der Nutzer zur Kommunikation verwendet. Dies können Telefone, Softwareapplikationen oder PDAs sein. Insbesondere Dienste, die die Interaktion mit dem Nutzer erfordern (z. B. Anrufweiterleiter) oder Zugriff auf den Medienstrom haben müssen, werden üblicherweise als Endgerätedienste ausgeführt.
- *Proxydienste* werden auf Servern oder anderen intermediären Netzwerkentitäten ausgeführt. Diese Dienste bieten Anrufsteuerdienste, die unabhängig von der Lokation des Nutzers sind. Beispielsweise ist dies das Weiterleiten von Anrufen, wenn das Telefon des Nutzers nicht verfügbar oder erreichbar ist. Hierzu zählen Dienste wie sie insbesondere von CPL-Skripten erfüllt werden.

3.4.1 Kontext zur Parametrisierung der Steuerungsabläufe von Kommunikationsdiensten

Die Funktionalität eines Kommunikationsdienstes kann formal durch Systembeschreibungssprachen wie die Specification and Description Language beschrieben werden. In SDL lässt sich das Verhalten von Diensten unter anderem durch *erweiterte endliche Automaten* (*Extended Finite State Machine – EFSM*) darstellen. Eine EFSM erweitert eine Finite State Machine (FSM) durch Variablen mit endlichem Wertebereich, die als Vorbedingung für Transitionen ausgewertet werden können. Zusätzlich kann eine *zeitliche* Komponente modelliert werden. Ein Dienst in SDL kann in Prozesse zerlegt werden, die mit Signalen über Kanäle miteinander kommunizieren.

Der funktionale Ablauf eines Kommunikationsdienstes ist kein atomarer Prozess, daher kann ein Kontext-bewusster Kommunikationsdienst durch *Modifikation* eines bestehenden Dienstes realisiert werden. Hierfür müssen im Ablauf Verankerungspunkte (*hooks*) integriert werden, in die die zusätzlichen Prozessblöcke für die Erweiterungen integriert werden können.

3.5 Fazit

Drei konzeptionelle Verfahren zur Steigerung der Effizienz bei der Handhabung von eingehenden Kommunikationsanfragen sowie beim Aufbau von Kommunikationen wurden vorgeschlagen. Dazu wurde eine Effizienzdefinition gegeben und die effizienzbestimmenden Parameter wurden identifiziert. In einer anschließenden Diskussion wurden die Parameter, die verändert werden können bestimmt und in die Verfahren einbezogen.

Zur Steigerung der Effizienz kann die Anzahl der nicht-zielführenden Kommunikationsversuche reduziert werden. Hierzu werden eine Kontext-bewusste Filterung der eingehenden Kommunikation sowie die Verteilung von Kontexten zur Vermeidung unnötiger Kommunikation vorgeschlagen. Zum anderen wird eine Reduktion des Aufwands des Nutzers durch Verlagerung auf das System erreicht. Hierfür wird der Ansatz eines Kommunikationsbrokers beschrieben, der günstige Zeitpunkte für eine Kommunikation in

Abhängigkeit der Kontexte der Teilnehmer und ihrer Kommunikationstypen aushandelt.

Kapitel 4

Kontext

Nam et ipsa scientia potestas est.
(*Denn schon das Wissen an sich ist Macht.*)

De Haeresibus
Francis Bacon

Aus der Beobachtung, dass der Kontext eine essentielle und implizit genutzte Rolle in der menschlichen vis-á-vis Kommunikation spielt, wurde im vorherigen Kapitel die Verwendung eines technischen Äquivalents für Kommunikationsdienste vorgeschlagen. Hierfür ist eine formale Modellierung des Prozesses seiner Erfassung notwendig.
In diesem Kapitel werden geläufige Definitionen von Kontexten vorgestellt und eine eigene Definition gegeben, die aus der Analyse der Kontexteigenschaften abgeleitet worden ist. Ein eigenes Kontext-Spiral-Modell beschreibt die Abbildung der theoretischen Erkenntnisse auf Prozesse für die automatische und maschinelle Erfassung von Kontexten. Aus den identifizierten Phasen der Kontexterfassung werden in den nachfolgenden Kapiteln die notwendigen Komponenten sowie eine geeignete Architektur und Kommunikationsinfrastruktur für die Umsetzung eines Frameworks entworfen.

Kurzübersicht

Die Struktur des vorliegenden Kapitels ist wie folgt: Eine historische Einführung, Definitionen und Eigenschaften zum Themenkomplex „Kontext“ sind in Abschnitt 4.1 ausgeführt. Abschnitt 4.2 formalisiert und definiert den Begriff „Kontext“ und seine Eigenschaften. Aus diesen werden in Abschnitt 4.3 Konzepte für den Prozess der Kontexterfassung und Nutzung für Kommunikationssysteme im Spiral-Modell abgeleitet. Abbildung 4.1 visualisiert die Struktur und die Beziehungen der Abschnitte des Kapitels.

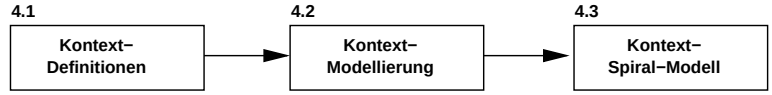


Abbildung 4.1: Struktur des Kapitels 4

4.1 Kontext: Konzept, Definitionen und Anwendungsgebiete

Der deutsche Philosoph Gotthard GÜNTHER führte die Ansätze von Georg Wilhelm Friedrich HEGEL zur Abkehr von der Aristotelischen Logik fort. Er kombinierte die *Polykontexturalitätstheorie* mit einem *mehrwertigen Logikkalkül* [Gün79]. Die entstehende Kernaussage ist, dass es keine *einzig gültige* Sicht auf die Welt gibt. Vielmehr befindet sich jeder Beobachter in einem unterschiedlichen *Kontext* und kommt daher zu einer anderen Sicht der Welt. Dieser Kontext ist bestimmt durch das soziale, kulturelle und ethische System, in dem der Beobachter lebt. Ein Transfer von Wissen zwischen zwei Ontologien unterschiedlicher Kontexte ist daher nicht zwingend möglich.

MCCARTHY formulierte in [MB97] ein *Regelsystem*, mit dem ein *Anheben* (*lifting*) des Wissens aus einem begrenzten Kontext in einen weitgefassteren Kontext möglich ist. Die Schwierigkeit solcher Systeme liegt in der Bestimmung einer Anhebe-Funktion. Darüber hinaus kann in diesem System Wissen von einem Kontext in einen anderen Kontext *transzendieren* (*transcend*). Das dafür notwendige Regelsystem muss jedoch von Experten erstellt werden.

Die Verwendung von Kontexten ist eine dem Menschen intuitiv innewohnende Fähigkeit. So besitzen Menschen die Fähigkeit, in mehreren Kontexten zu denken und so Informationen mit anderen Menschen auszutauschen. Darüber hinaus teilen Menschen einen gemeinsamen Erfahrungskontext und können zur Beschreibung Metaphern verwenden. Diese Fähigkeiten versetzen den Menschen in die Lage, zwischen Kontexten zu transzendieren, was eine Vorbedingung für die Schaffung von Wissen ist.

Generellere Kontexte sind *mächtig*e Objekte (*rich objects*) einer Domäne in der Hinsicht, dass sie eine Vielzahl an Annahmen haben, die in der Regel nicht alle aufgeführt werden können [Guh91]. Ein Kontext kann daher nicht komplett beschrieben werden, was dazu führt, dass Kontexte nicht vollständig durch ein Logiksystem ausgedrückt werden können. Für ein *Künstliche Intelligenz* (KI)-System (*Artificial Intelligence – AI*) wurden in [MB97] eine Algebra und dazugehörige Operationen entworfen. Der $ist(c, p)$ Operator wurde eingeführt, um auszudrücken, dass eine Aussage (*statement*) p im Kontext c wahr ist. Diese Aussage ist in einem den Kontext umgebenden *äußeren* Kontext c' gültig. Es kann keine Aussage innerhalb des Kontexts, in dem man sich befindet, getroffen werden.

$$c' : ist(c, p) \tag{4.1}$$

Mit p wird dabei eine *Aussage* in einem Kontext bezeichnet. Die Idee dabei ist, in einen inneren Kontext zu wechseln, dort ein Problem zu lösen und eine gewonnene Lösung

auf den äußeren Kontext zu übertragen, was im Allgemeinen einer *Generalisierung* der Lösung entspricht. Auch gibt es eine wechselseitige Beeinflussung des inneren und des äußeren Kontexts. Die Schachtelung der Kontexte ist in Abbildung 4.2 dargestellt. Ein reales Beispiel ist folgendes: So kann der Nutzer in einem bestimmten Kontext die Anwendung anweisen eine Netzverbindung aufzubauen. Dies führt dazu, dass dem Nutzer Hotspots angezeigt werden und er sich zu diesen bewegen muss, was wiederum seinen Ortskontext ändert. Ein Kontext ist nicht statisch, sondern sehr dynamisch und muss immer wieder neu erfasst werden.

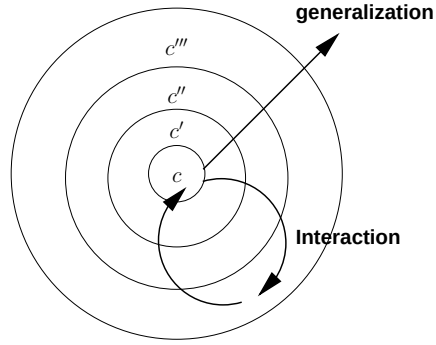


Abbildung 4.2: Zusammenhang zwischen inneren und äußeren Kontexten

4.1.1 Definition des Begriffs „Kontext“

Eine Vielzahl an Definitionen des Begriffs *Kontext* werden in der Literatur gebraucht, um die Bedeutung von Wörtern oder Sätzen in einen Sinnzusammenhang zu stellen. In der Informatik findet sich der Begriff mit unterschiedlichen Bedeutungen in den Gebieten der Künstliche Intelligenz (KI), der Nutzerinterfaces, oder Betriebssystemen oder auch der Grammatiken. Die in mittlerweile fast allen Software-Anwendungen zu findenden Kontextmenüs sind eine Ausprägung hiervon. Es zeigt sich, dass es keine Definition von Kontext gibt, die global für alle Bereiche gültig ist [MB97]. Vielmehr existiert eine Reihe von Definitionen, die in ihrer jeweiligen Wissensdomäne, in der sie verwendet werden, gültig sind.

Die jeweiligen Kontexte können sich in verschiedene Kontextklassen kategorisieren lassen. So können diese beispielsweise in situative, inhaltliche, aber auch in Umgebungs-, Anwendungs- und Nutzer-Kontexte unterschieden werden. So werden in diesem Abschnitt auch nur eine sehr allgemeine Definition sowie eine akzeptierte Definition aus dem Umfeld des ubiquitäteren Rechnens (*Ubiquitous Computing*) und der Wissensrepräsentation gegeben.

Im Duden [Bib01] ist folgende Definition zu „Kontext“ zu finden:

Definition 4.1 (Sprachlicher Kontext)
Kontext <lat.> der; -[e]s, -e: 1. (Sprachw.) a) der umgebende Text einer gesprochenen oder geschriebenen sprachlichen Einheit; b) (relativ selbstständiges) Text- od. Redestück;

c) der umgebende inhaltliche (Gedanken-, Sinn)zusammenhang, in dem eine Äußerung steht, u. der Sach-, u. Situationszusammenhang, aus dem heraus sie verstanden werden muss; vgl. Kontext. 2. umgebender Zusammenhang.

Die Verwendung des Konzepts Kontext ist in unterschiedlichen wissenschaftlichen Disziplinen wie Sprachwissenschaften, Psychologie oder Philosophie bekannt und untersucht worden. Definitionen in diesem Umfeld unterscheiden sich vom Begriff Kontext in der Informatik. Dort wird dieser insbesondere im Bereich von Human Computer Interaction (HCI) und der KI (*Artificial Intelligence – AI*) [MP95, BBM95, Bré99] gebraucht. Die Anstrengungen in der HCI beziehen sich auf die Einbeziehung des Kontexts zur Verbesserung der Kommunikation zwischen dem Nutzer und der Anwendung, wie dies bei Nutzerschnittstellen der Fall ist.

Eine ähnliche Ansicht wie in der KI wird in der algebraischen Wissensrepräsentation verwendet, die sich Eigenschaften der Begriffsintentionen mittels formaler Begriffsanalyse [Wil92] zu eigen macht. Hier wird ein *formaler Kontext* definiert als:

Definition 4.2 (Formaler Kontext)
*Seien G und M Mengen und I eine Relation zwischen G und M, das heißt $I \subseteq (G \times M)$. Ein **formaler Kontext** ist eine Mengenstruktur $K := (G, M, I)$. Die Elemente von G nennt man Gegenstände, die Elemente von M Merkmale und gIm steht dafür, dass der Gegenstand g das Merkmal m besitzt.*

Auch Definition 4.2 legt nicht fest, welche Mengen von Merkmalen und Gegenständen betrachtet werden müssen. Aus jeder Menge kann jedoch eine Gegenstandsmenge und eine Merkmalsmenge entstehen, wenn die Relation I existiert. Die Betrachtung des formalen Kontexts bezieht sich auf die Generierung von formalen Begriffen. Jedoch kann das interessante Relationskonzept auch auf die angestrebte Verwendung von Kontexten für die Beeinflussung von Kommunikationsbeziehungen übertragen werden. Auch in diesem Anwendungsfeld ergibt nicht jeder Dienst für jeden Nutzer zu jedem Zeitpunkt einen Sinn. Wichtig ist jedoch die Beziehung zwischen den beteiligten Entitäten.

4.1.1.1 Kontextdefinitionen und Einsatz von Kontext im Umfeld des Ubiquitous Computing

Ein neues Anwendungsfeld ist das *Ubiquitäre Rechnen* (*ubiquitous computing*), welches seinen Anfang in dem wegweisenden Aufsatz „The Computer for the 21st Century“ von Mark Weiser [Wei91] hat. Gebräuchlich sind auch die Begriffe *invisible computing* [Nor98], *calm computing* [WB98], *pervasive computing* [BHH+01] oder *disappearing computer* [Wej00]. Hierbei weben sich Computer als unsichtbare Helfer in die natürliche Umgebung der Nutzer ein – sie verschwinden aus der *Wahrnehmung* als implizit zu nutzende Technik.

Für die Einbeziehung von Kontexten, die die Ausführung von Anwendungen beeinflussen wurde in [SAW94] der Begriff *Context awareness* geprägt und damit das Gebiet des *Context-aware Computing* benannt, was eine Schnittmenge des Ubiquitous Computing

darstellt. Im Deutschen existiert eine Reihe von Übersetzungen dieser Begriffe. So sind „Kontext-abhängig“, “Kontext-sensitiv“, „Kontext-bezogen“ oder „Kontext-bewusst“ gebräuchlich. Für diese Arbeit ist der Begriff *Kontext-bewusst* gewählt worden, da dieser nicht strikt von einer Abhängigkeit der Anwendung vom Kontext spricht. Vielmehr sind die Anwendungen (im übertragenen Sinn) ihrem Kontext *bewusst* und *können* ihn einbeziehen.

Eine Reihe interessanter Kontext-bewusster Anwendungen ergibt sich aus der Erweiterung von existierenden Anwendungen und der Einbeziehung von Kontexten. Ein häufig verwendetes Feature im Mensch-Maschine-Dialog sind Kontextmenüs in Softwareanwendungen. Weiterhin sind Haussteuerungen, elektronische Reiseführer oder Informationsaufbereitungssysteme zu nennen. Neuere Ansätze beziehen Kontexte zur Bewertung von Sicherheitsbeziehungen oder zur Zugangskontrolle [Mos02] ein.

Im Umfeld von HCI und Ubiquitous Computing findet sich eine Reihe von Ansätzen für eine Definition des Begriffs Kontext. Einige Definitionen verwenden *Beispiele* wie Lokation, Identität der Nutzer, Zeit oder Umgebung für die Umschreibung des Begriffs [BBC97, RPM98, Dey98]. Als eine weitere Form von Definitionen findet man in der Literatur die Aufzählung von *Synonymen* wie Situation, Zustände und Umgebung [Bro96, FF98, WJH97, RCDD98, HNBR97]. Formale Ansätze durch objektorientierte Klassifikationen finden sich in [SAT+99, SDA99, CK00, SAW94]. Eine Übersicht zu Kontext-bewussten Anwendungen ist in [Dey00] aufgeführt, worin auch eine häufig zitierte anwendungszentrierte Definition von Kontext im Bereich des Ubiquitous Computing gegeben ist.

Definition 4.3 (Kontext)
Kontext ist jede Information, die verwendet werden kann, um die Situation einer Entität zu beschreiben. Entitäten sind Personen, Plätze oder Objekte, die als relevant für den Nutzer, die Applikation und die Interaktion zwischen dem Nutzer und der Applikation erachtet werden.

Der Kerngedanke der im Ubiquitous Computing häufig verwendeten Definition ist die Betrachtung von Kontext als Repräsentation der umgebenden Situation. Diesem auf die Entität bezogenen *situativen Kontext* wird auch in dieser Arbeit gefolgt. Der Schwerpunkt dieser Arbeit liegt auf der Interaktion zwischen Menschen und weniger auf der Mensch-Maschinen Interaktion (*Human Computer Interaction – HCI*), wie er in den meisten Arbeiten im Umfeld von *context-aware computing* verfolgt wird.

4.1.2 Kontext-bewusste Anwendungen

Eine erste Klassifikation Kontext-bewusster Anwendungen wurde in [SAW94] vorgenommen. Diese teilt die Anwendungen in zwei Dimensionen auf. Dies sind automatische oder manuelle Kontexterfassung und Informationsbereitstellung oder Dienstausführung. Die vier Bereiche werden als *naheliegende Auswahl* (*proximate selection*), *automatische Kontext-bewusste Rekonfiguration* (*automatic contextual reconfiguration*),

Kontext-bewusste Informationen und Aktionen (*contextual information and commands*) sowie *Kontext-ausgelöste Aktionen* (*context-triggered actions*) bezeichnet.

Ein ähnlicher Ansatz wird auch in [Sal00] vorgestellt. Es werden die drei Kategorien *Anzeigen von Informationen und Diensten* (*presenting information and services*), *automatische Ausführung von Diensten* (*automatically executing a service*) und *Annotieren von Informationen mit Kontext für spätere Verarbeitung* (*Attaching context information for later retrieval*) unterschieden. Im Wesentlichen wird hier nur die Annotation von Informationen mit Kontexten hinzugefügt. Aus dieser soll bei einer späteren Verarbeitung der Erfassungsprozess besser nachvollzogen werden können.

Eine Unterteilung in 6 Typen, die eine Erweiterung der zuvor vorgestellten Kategorien sind, wurde in [BBL+00] vorgenommen:

Type 1: Proaktive Auslösung (*proactive triggering*): Auf Basis des aktuellen Nutzerkontexts werden relevante Informationen und Dienste bereitgestellt.

Type 2: Verbesserte Interaktion (*streamlining interaction*): Die eingesetzten Geräte erfassen ihren Kontext und bieten passende Dienste zur gemeinsamen Mensch-zu-Mensch-Kommunikation an.

Type 3: Speicherung für spätere Ereignisse (*memory for past events*): Jeder erfasste Kontext wird für einen späteren Gebrauch gespeichert und anderen Anwendungen verfügbar gemacht.

Type 4: Erinnerung zukünftiger Kontexte (*reminders for future contexts*): Bei Anwendungen diesen Typs werden Aktionen spezifiziert, die bei einem Auftreten eines Kontexts ausgelöst werden sollen. Tritt der spezifizierte Kontext ein, so wird die entsprechende Aktion ausgelöst.

Type 5: Optimierung von Verhaltensmustern (*optimizing the patterns of behavior*): Bei diesem Typ werden der Kontext und das dazugehörige Nutzer- oder Anwendungsverhalten aufgezeichnet, um es später zu analysieren. Aus der Analyse sollen Verbesserungsvorschläge für die nächsten Aktionen in einem vergleichbaren Kontext abgeleitet werden.

Type 6: Gemeinsame Erfahrungen (*sharing experiences*): Der Kontext wird zur Unterstützung von Gruppenarbeiten erfasst und den Teilnehmern zur Verfügung gestellt.

Die aufgelisteten Typen finden sich in fast allen Ausprägungen von Kontext-bewussten Anwendungen und Systemen. Die aufgeführten Typen werden später für die Beschreibung des eigenen Kontext-bewussten Systems verwendet und in Tabelle 5.1 aufgeführt.

4.2 Modellierung von Kontext und seinen Eigenschaften

Eine Reihe von Definitionen des Begriffs „Kontext“ ist in Unterabschnitt 4.1.1 gegeben worden. Es ist offensichtlich, dass der Begriff überbelegt ist und in unterschiedlichen

Domänen spezifische Bedeutungen hat. Eine eigene Definition für Kontext und seine Eigenschaften wird nachfolgend gegeben. Die Definitionen beziehen die zuvor angegebenen Definitionen mit ein und übertragen sie auf die Domäne der Kommunikationsdienste.

Aus der gegebenen Definition und begrifflichen Modellierung von Kontext und seiner Eigenschaften werden anschließend (in Abschnitt 4.3) die notwendigen Modelle und Prozesse für eine Verwendung innerhalb eines Computersystems entwickelt und die Systemarchitektur bzw. die notwendigen Komponenten abgeleitet (in Abschnitt 5.1).

In dieser Arbeit wird folgende eigene Definition von *Kontext* verwendet.

Definition 4.4 (Kontext)
*Ein **Kontext** ξ ist eine abstrakte, bedeutungstragende Beschreibung einer Aktion eines Objektes und ihrer Beziehung zu anderen Objekten. Ein Objekt kann dabei eine Situation, eine Entität oder eine Umgebung sein.*

Kontexte sind beispielsweise [1]:

$$\Xi = \{\underline{\text{Meeting}}, \text{Unterhaltung}, \text{Verwunderung}, \ldots\}$$

Zusätzlich werden folgende zwei Annahmen getroffen, die für die Erfassung und der Bestimmung des Kontexts aus messbaren Bestandteilen notwendig sind.

Annahme 1 *Jeder Kontext kann durch eine **charakteristische Funktion** approximiert werden.*

Annahme 2 *Ein Kontext ξ ist ein Objekt, das aus einer großen Anzahl von Kontextmerkmalen ζ besteht, geschrieben*

$$\xi = \{\zeta_1, \zeta_2, \ldots, \zeta_n\} \tag{4.2}$$

Ein Kontextmerkmal wird dabei wie folgt definiert:

Definition 4.5 (Kontextmerkmal)
*Ein **Kontextmerkmal** oder eine Kontextinformation ζ ist jede abstrakte relevante Information, die zur Charakterisierung eines Kontexts in einem bestimmten Gültigkeitsbereich verwendet werden kann.*

Kontextmerkmale sind z. B.:

$$Z = \{\underline{\text{Lokation}}, \underline{\text{Zeit}}, \text{Temperatur}, \text{Druck}, \underline{\text{Lautstärke}}, \ldots\}$$

Die abstrakten Kontextmerkmale können von Maschinen oder Menschen erfasst werden, wodurch sie *instanziert* werden und folgende weitere Eigenschaft besitzen.

[1] Unterstrichene Begriffe sind für das weitere Beispiel von Bedeutung.

Annahme 3 *Jedes Kontextmerkmal ist ab seiner Instanzierung abhängig von der Zeit t und der Umgebung ε, in der es erfasst wird.*

Ein instanziertes Kontextmerkmal hat einen Wert und eine dazugehörige Einheit. Instanzierte Kontextmerkmale des obigen Beispiels sind:

$$Z = \{\underline{\text{S3|06/448}}, \underline{\text{13:41:21Z}}, 22\ ^\circ\text{C}, 1050\ \text{hPa}, \underline{4\ \text{sone}}, \ldots\}$$

Ein Merkmal kann dabei direkt erfasst werden oder es wird aus Sensorwerten gewonnen. Dies kann zum Beispiel mittels *Merkmalsextraktion* aus Sensorwerten s oder der Kombination dieser erfolgen. Die surjektive Vorschrift

$$\varphi\colon \vec{s} \rightarrow \zeta \tag{4.3}$$

bildet dabei einen Sensorwertevektor, geschrieben als

$$\vec{s} = [s_1, \ldots, s_m]^T \tag{4.4}$$

auf ein Merkmal ζ ab.

Sensordaten zu den unterstrichenen Kontextmerkmalen des Beispiels könnten wie folgt sein:

$$S = \left\{ \begin{pmatrix} -30\ \text{dB} \\ -12\ \text{dB} \\ -48\ \text{dB} \end{pmatrix}, 123423804\ \text{Ticks}, \begin{pmatrix} 3.7\ \text{sone} \\ 4.2\ \text{sone} \\ 3.9\ \text{sone} \\ 4.1\ \text{sone} \end{pmatrix}, \ldots \right\}$$

Dieser Prozess kann konstruktiv angewandt werden, wobei mehrere Merkmale zu einem neuen Merkmal zusammengefasst werden können. Somit kann ein Merkmal aus Einzelwerten bestehen oder aber aus mehreren Werten zusammengesetzt sein und als Vektor

$$\vec{\zeta} = [\zeta_1, \ldots, \zeta_k]^T \tag{4.5}$$

repräsentiert werden.

Jedes Kontextmerkmal ζ ist abhängig von der Zeit t und der Umgebung ε, in der es erfasst worden ist. Daraus folgt, dass jedes Kontextmerkmal und damit jeder Kontext inhärent *orts- und zeitabhängig* ist [SG01, BSGT03]. Innerhalb dieser beiden Attribute bestimmt sich die *Gültigkeit* bzw. die *Relevanz* der Information. Für die weitere Verarbeitung im System ist auch auf eine Zeitsynchronisierung zu achten. Die Dimensionen der Gültigkeit können jedoch stark variieren. Dieses Konzept ist intuitiv erfassbar und entspricht den vom Menschen wahrgenommenen Eigenschaften von Ereignissen. Implizit wird die Abhängigkeit von Raum und Zeit bei der Bewertung von Ereignissen angewendet.

Die maximale Gültigkeit hat ein Merkmal zu dem Zeitpunkt, an dem es erfasst wurde (t_δ) und an dem Ort bzw. in der Umgebung, an dem die Erfassung des Phänomens

stattgefunden hat (ε_δ). Dies wird als *Ursprung* δ des Phänomens bezeichnet.

$$\delta = (t_\delta, \varepsilon_\delta) \tag{4.6}$$

Eine *Abklingfunktion* (*decay function*) g kann verwendet werden, um die Eigenschaft der Orts- und Zeitabhängigkeit darzustellen. Jede Funktion g, die reellen Werte, die der zeitlichen oder räumlichen Differenz vom Ursprung δ des Phänomens entsprechen, auf den normierten Bereich $[0, 1]$ abbildet, kann verwendet werden.

$$g\colon x \mapsto [0, 1] \tag{4.7}$$

Da die Gültigkeit mit wachsendem Abstand abnehmen soll, wird als Abklingfunktion g eine beliebige monoton fallende Funktion gefordert. Die Gültigkeit von zukünftigen Ereignissen wie Kalendereinträgen oder von Ergebnissen einer Kontextvorhersage lassen sich ebenfalls modellieren. In diesem Fall liegt t_δ vor der aktuellen Zeit.

Mit zunehmender Entfernung vom Punkt δ nimmt die Gültigkeit einer gemessenen Lautstärke für den Prozess, der an dem Wert an Punkt δ interessiert ist, ab. Eine Abklingfunktion für die Gültigkeit in Abhängigkeit der Entfernung ist in Abbildung 4.3 dargestellt. Als Funktion wurde in diesem Beispiel

$$g(x, y) = \frac{4}{4 + x^2 + y^2} \tag{4.8}$$

gewählt.

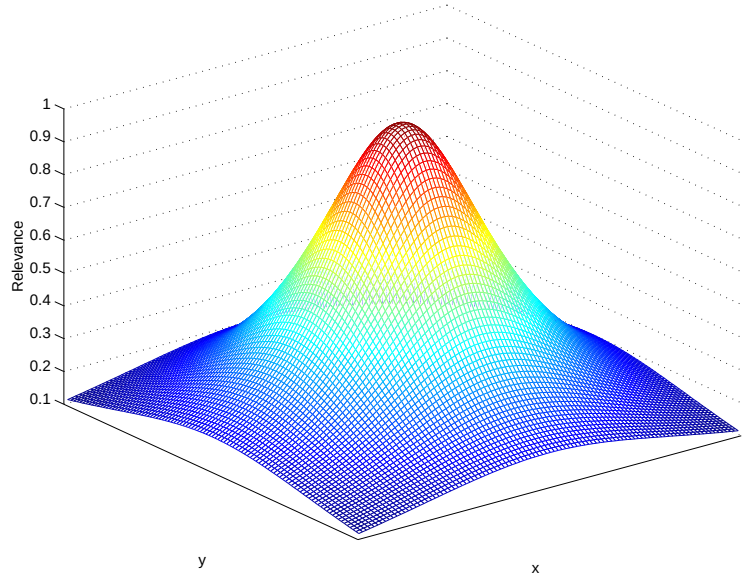
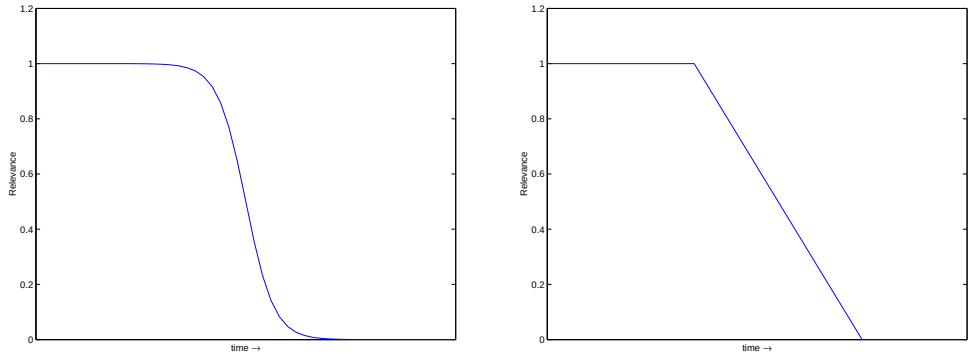


Abbildung 4.3: Abnahme der räumliche Relevanz in x- und y-Richtung

Der gemessene Wert der Lautstärke nimmt zusätzlich mit der seit der Messung vergangenen Zeit ab. Zu einem Zeitpunkt t_z ist die messbare Lautstärke Null bzw. nahe Null. Der Wert hat damit für die Einbeziehung des Merkmals keine Relevanz mehr. Zwei unterschiedliche Abklingfunktionen für die Gültigkeit des Messwerts in Abhängigkeit von der Zeit sind in Abbildung 4.4 dargestellt. Eine sigmoide Abklingfunktion g_{sigmoid} ist in Abbildung 4.4(a) dargestellt, die einen weichen Übergang zwischen der vollen Gültigkeit und der Degradierung hat. Eine *trapezoide* Funktion, gegeben durch

$$g_{\text{trapeziod}}(t) = \begin{cases} 0 & : t \geq t_z \\ 1 & : t \leq t_x \\ \frac{t_z - t}{t_z - t_x} & : \text{ sonst} \end{cases} \tag{4.9}$$

ist in Abbildung 4.4(b) dargestellt. Diese Funktion wird häufig eingesetzt, da sie einfacher zu handhaben ist als die sigmoide Funktion.



(a) Zeitliche Relevanz als sigmoide Funktion

(b) Zeitliche Relevanz als trapezoide Funktion

Abbildung 4.4: Zeitliche Relevanz des Kontexts

In der Regel kann die Menge Z aller Merkmale ζ nicht vollständig erfasst werden. Zur Bestimmung des Kontexts wird eine charakteristische Menge K mit signifikanten Merkmalen mit

$$K \subseteq Z \tag{4.10}$$

gesucht, aus der sich der Kontext derart komponieren lässt, dass sich eine hinreichend genaue Approximation ergibt.

Die einzelnen Merkmale eines Kontexts werden bei der Erfassung derart verknüpft, dass eine möglichst gute Approximation und damit eine hohe „Erkennung“ des Kontexts erreicht wird. Dieses vom Menschen zur Erfassung eingesetzte Verhalten wird in Annahme 1 als Funktionsvorschrift für ein System beschrieben. Die Funktion wird definiert als:

Definition 4.6 (Charakteristische Funktion)
*Eine **charakteristische Funktion** χ ist eine Vorschrift, die eine Teilmenge aller*

Kontextmerkmale eines Kontexts verknüpft, so dass der Kontext hinreichend genau bestimmt werden kann.

Die charakteristische Funktion ist eine surjektive Funktion und bildet die Menge aller Kontextmerkmale Z auf die Menge der Kontexte Ξ ab, geschrieben als

$$\chi \colon Z \to \Xi \tag{4.11}$$

mit

$$Z = \{\zeta_1, \zeta_2, \ldots, \zeta_n\} \text{ und } \Xi = \{\xi_1, \xi_2, \ldots, \xi_m\} \tag{4.12}$$

Die Verknüpfung der einzelnen Kontextmerkmale wird dabei mit einem generischen Verknüpfungsoperator $\otimes$ bezeichnet. Die Merkmale können bei der Verknüpfungvorschrift noch mit Gewichtungsfaktoren w_i gewichtet werden.

$$\chi = \bigotimes_i w_i \cdot \zeta_i(t_i, \varepsilon_i) \tag{4.13}$$

Der Verknüpfungsoperator stellt hierbei keinen konkreten Operator dar, sondern steht als Repräsentant für die Klasse aller möglichen Operatoren. Üblicherweise wird es auch keinen einzelnen Operator geben, wie in Gleichung (4.13) dargestellt. Vielmehr werden für bestimmte Kontextmerkmalsklassen spezifische Operatoren zur Verknüpfung eingesetzt, wie dies in Gleichung (4.14) gezeigt ist.

$$\xi = (\zeta_1 \otimes_1 \zeta_2 \otimes_1 \zeta_3) \otimes_2 (\zeta_4 \otimes_3 \zeta_5 \otimes_3 \zeta_6 \otimes_3 \zeta_7) \otimes_4 \ldots \otimes_2 (\zeta_{n-1} \otimes_m \zeta_n) \tag{4.14}$$

Insbesondere die Kontextmerkmale, die sich in ihren Eigenschaften und Repräsentationsformen sowie ihrer Abstraktionsebene stark unterscheiden können, machen es unmöglich einen gemeinsamen Operator zu finden. So lassen sich symbolische Lokationsinformationen nicht in einer einfachen Operation mit einem nummerischen Operator kombinieren, während Daten einer sensorischen Temperaturmessung sich gut mathematisch verarbeiten lassen. Die Verwendung von Regel und einem Regelsystem hingegen ist gut geeignet, Daten unterschiedlicher Informationsquellen zu verknüpfen.

Mögliche Operatoren sind Logik-Funktionen wie Vereinigung oder Durchschnitt oder aber statistische Verfahren wie Mittelwertbildung oder Mehrheitsentscheidungen. Ebenso können Methoden aus dem Soft Computing wie Fuzzy Logik, Neuronale Netze oder Entscheidungsbäume mit Wahrscheinlichkeiten als Operatoren eingesetzt werden. Die Darstellung in Gleichung (4.14) zeigt auch, dass eine hierarchische Verknüpfung der einzelnen Informationen möglich ist. Die jeweiligen Typen an Informationen werden durch die Verknüpfung aggregiert und mit der nächsten Stufe weiter zusammengefasst.

Die Gesamtbeschreibung dieses Aggregationsprozesses bzw. Kompositionsprozesses kann durch eine Regelvorschrift beschrieben werden, die dem $\otimes$-Operator in Gleichung (4.13) entspricht. Eine graphische Darstellung des Prozesses der Komposition von Kontexten aus Sensorendaten und Merkmalen in endlich vielen Schritten ist in Abbildung 4.5 gezeigt.

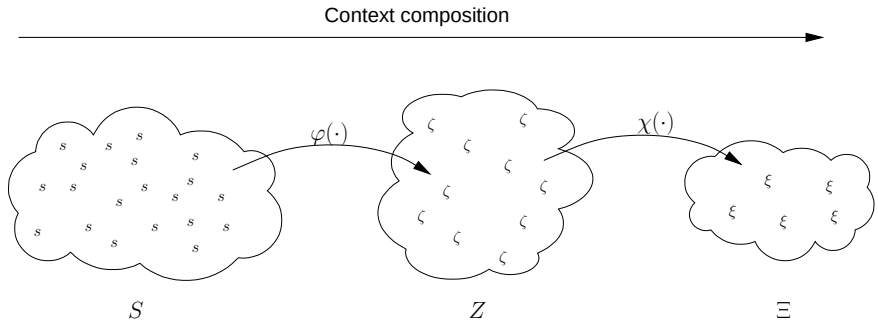


Abbildung 4.5: Komposition von Kontexten aus Merkmalen und Sensordaten

Für die weitere Handhabung des Kontexts zwischen Menschen und innerhalb eines Systems muss sich eindeutig auf einen Kontext bezogen werden können.

Annahme 4 *Jeder Kontext kann durch einen* ***Kontextbezeichner*** λ *benannt werden.*

Der Bezeichner λ dient der Identifikation des Kontexts und zur Benennung im sprachlichen Austausch zwischen Menschen sowie auch zwischen Anwendungen.

$$\lambda \multimap \xi \tag{4.15}$$

Der Bezeichner kann dabei eine Zeichenkette, ein Unified Resource Identifier (URI) oder eine Datenstruktur sein.

Ein *Kontextobjekt* c wird somit nach Definition 4.4 durch

$$c = (\lambda, \xi) \tag{4.16}$$

beschrieben. Zur genauen Beschreibung des Kontextbezeichners wurde ein Format entwickelt und in Unterabschnitt 7.5.2 beschrieben. Dieses Format erlaubt auch das standardisierte Einlesen und Verwenden in verschiedenen Anwendungen und die Verteilung von Kontexten.

4.3 Kontext-Spiral-Modell

Neben der begrifflichen Definition von Kontext wurde auch eine *funktionale* Beschreibung für die Verwendung von Kontexten entworfen. Dieses soll die Gewinnung, Verteilung und Nutzung von Kontext und Kontextmerkmalen für die konkrete Realisierung von Systemen und Anwendungen beschreiben.

Die Analyse existierender Ansätze im Gebiet der Sensorfusion sowie im Umfeld von Kontext-bewussten Anwendungen [Bas00, Elm02, Sch02b, Wu03] führte zum eigenen

Kontext-Spiral-Modell, welches die Phasen der *Kontexthandhabung* funktional beschreiben. Das nachfolgend vorgestellte Modell ist vom *Omnibusmodell* [BO99] von Bedworth und O'Brien beeinflusst. Das Omnibusmodell besitzt eine zyklische Anordnung wie auch das Modell von Boyd [Boy87], hat aber im Gegensatz zum JDL Modell [Kes92, SBW99, BP02] eine festgelegte Reihenfolge der einzelnen Prozesse.

Im Gegensatz zu anderen Modellen besitzt das *Kontext-Spiral-Modell* eine zusätzliche zeitliche Achse. Das Modell ist in Abbildung 4.6 dargestellt und besteht aus der zeitlichen Anordnung der vier Phasen:

Erfassung (*acquisition*): Abtasten und Observieren der Umgebung zur Erfassung der Phänomene eines Objekts.

Synthese (*synthesis*): Verarbeiten der Daten, um diese zu höherwertigen Informationen zu kombinieren.

Verteilung (*dissemination*): Transport der Kontextinformationen vom Ort, an dem sie erfasst wurden zum Ort, wo sie weiterverarbeitet oder verwendet werden.

Nutzung (*use*): Die Verwendung des Kontexts in einer Anwendung.

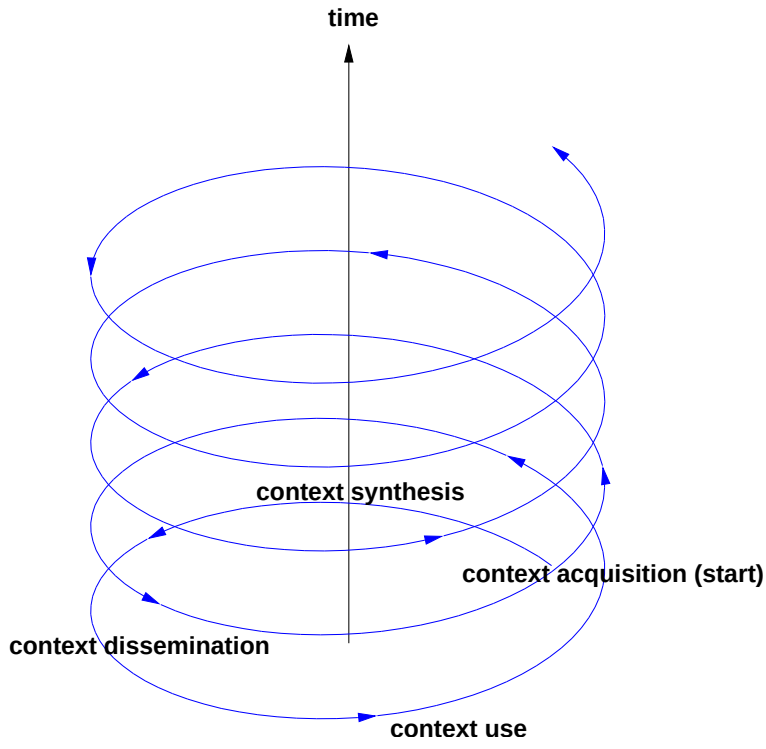


Abbildung 4.6: Prozess der Kontextgewinnung

Die Prozessphasen müssen nicht stringent in der eingezeichneten Ordnung durchlaufen werden. Befindet sich die Erfassungseinheit auf der Entität, die die Kontextinformationen nutzen will, so ist keine weitere Verteilung der Daten notwendig. Querverbindungen

zwischen den einzelnen Blöcken sind nicht eingezeichnet worden, um die Übersichtlichkeit zu erhalten. Die Spiralform des Modells verdeutlicht, dass der Prozess der Kontextgewinnung nicht nach einem einmaligen Durchlauf beendet ist. Dies ist zum einen notwendig, da sich der Kontext einer Entität potenziell ändert. Zum anderen kann die Einbeziehung des Kontexts in einen Entscheidungsprozess eine Änderung des äußeren Kontexts nachsichziehen, was eine Neubewertung des Kontexts notwendig macht.

Die einzelnen Phasen werden nachfolgend im Detail vorgestellt. Jede Phase unterteilt sich in weitere Unter-Prozesse, was eine weitere Verfeinerung des Modells erlaubt. Für jede Phase werden die geleisteten eigenen Beiträge aufgeführt und beschrieben.

4.3.1 Erfassung von Kontextinformationen

Die Erfassung eines Kontexts kann in der Regel nicht durch eine direkte „Messung“ erfolgen, daher wird der Kontext *indirekt* durch seine ihn beschreibenden Charakteristika bestimmt. Die charakteristischen Eigenschaften können üblicherweise durch Sensoren erfasst werden. Wie in Abschnitt 4.1 erwähnt, ist es nicht möglich, alle Charakteristika eines Kontexts aufzulisten, da es sich um mächtige Objekte handelt. Daher muss eine Teilmenge der Eigenschaften gewählt werden, die den Kontext hinreichend genau beschreibt. Die Auswahl der Eigenschaften ist nicht trivial und ergibt sich nicht stringent aus dem zu erfassenden Kontext. Vielmehr ist hierbei Expertenwissen bei der Modellierung notwendig. Darüber hinaus sind üblicherweise mehrere Iterationen notwendig, bis eine robuste Menge und Konfiguration an Sensoren gefunden werden kann, die eine hinreichend gute Approximation des Kontexts liefert.

Zur Erfassung der notwendigen Informationen zur Kontextgewinnung wurde ein Top-Down-Ansatz gewählt, welcher die stufenweise Unterscheidung des informatischen Verständnisses [Bae98] berücksichtigt. Hierbei werden die Stufen, Zeichen, Daten, Informationen und Wissen unterschieden. Daten besitzen im Gegensatz zu Zeichen eine für sie gültige Syntax. Informationen sind bedeutungstragende Daten. Wissen ist bei dieser, nach ihrem Sinngehalt abgestuften Unterscheidung, als Information, die in eine Handlung umsetzbar ist, definiert [KMB96]. In dieser Arbeit wird Kontext als Wissen angesehen, welches sich aus Kontextinformationen zusammensetzt und eine weitere Handlung beeinflusst. Die einzelnen Wissensstufen, die für diese Arbeit definiert wurden, sind in Abbildung 4.7 dargestellt.

Der Block *Erfassung* kann in die Unterprozesse *Abtastung* (*sensing*) und *Datenfusion* unterteilt werden, wie dies in Abbildung 4.8 gezeigt ist. Auf der untersten Stufe nehmen Sensoren Eigenschaften des zu beobachtenden Objekts wahr und verarbeiten die so gewonnenen *Sensorwerte* v, welche die Messung in elektrischen Einheiten wie Volt oder Ampere repräsentieren. Diese Werte werden im Sensor zu *Sensordaten* s, die eine syntaktische Struktur und eine Einheit des gemessenen Phänomens z. B. lx oder cd haben, umgewandelt. Diese Daten, auch Rohdaten genannt, werden vom Sensor bereitgestellt und können ausgelesen oder mittels eines Protokolls verteilt werden. Die Güte der Rohdaten ist in vielen Fällen nicht ausreichend, da sie Ungenauigkeiten und Schwankungen

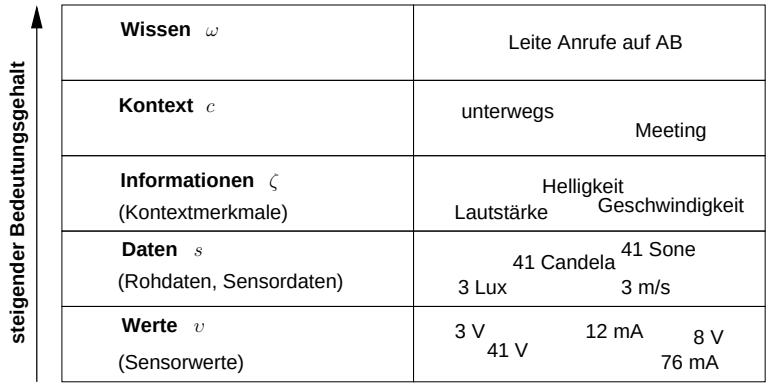


Abbildung 4.7: Stufen des Bedeutungsgehalts (adaptiert aus [Faa04])

unterworfen sind. Daher wird oftmals eine Rohdatenfusion durchgeführt, wie sie in Unterabschnitt 6.2.2 beschrieben ist. Durch den Einsatz von Aggregationsoperatoren wie Filter kann zusätzlich die Menge der anfallenden Daten reduziert werden, was die benötigte Bandbreite sowie die für die Weiterverarbeitung notwendige Rechenleistung reduziert.

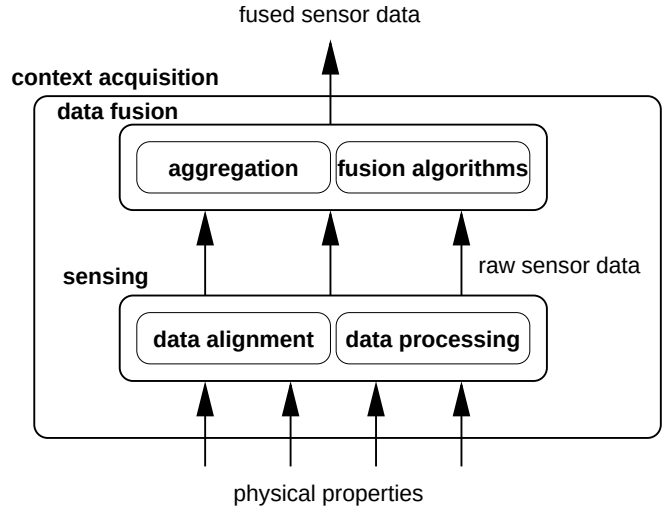


Abbildung 4.8: Erfassung von Sensordaten und anschließende Fusion der Daten

Zur Erfassung von Kontextinformationen ist eine Vielzahl an Sensoren notwendig. Eine geeignete Architektur, welche die einzelnen Sensoren und eine Menge an nützlichen Operatoren enthält, ist notwendig.

4.3.2 Synthese-Prozess

Die Daten, die von logischen und physischen Sensoren in einem syntaktisch korrekten und bekannten Format vorliegen, werden im Synthese-Prozess zu Informationen aggregiert. Das Ergebnis sind Informationen auf einem höheren Abstraktionsniveau und mit einem größeren Bedeutungsgehalt. Der Prozess kann in die Sub-Prozesse *Merkmalsextraktion* (*feature extraction*) und *Entscheidungsfindung* (*decision making*) gegliedert werden, wie er in der Abbildung 4.9 schematisiert ist.

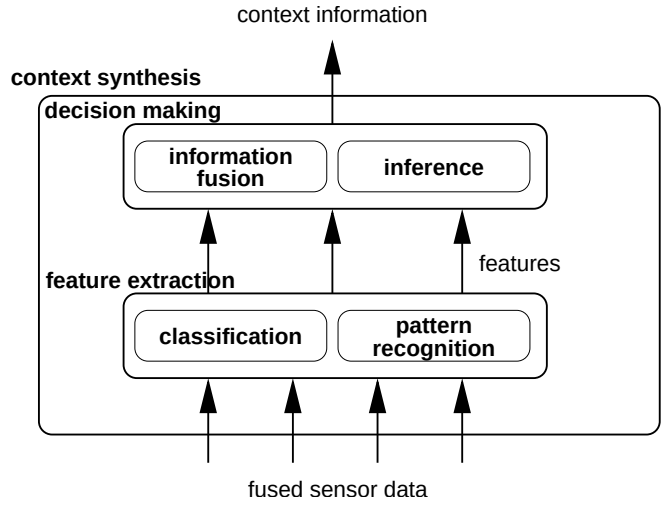


Abbildung 4.9: Synthetisierung der Kontextmerkmale durch Merkmalsextraktion und Inferenzbildung

In der Merkmalsextraktionsphase werden diejenigen Sensordaten ausgewählt, die charakteristische Merkmale des Kontexts repräsentieren. Aus den Daten werden dann die interessanten Merkmale (*features*) ζ extrahiert. Die Annahme ist hierbei, dass eine abzählbare und vom System handhabbare Menge an Merkmalen dafür ausreicht. Aus den vorliegenden Sensordaten werden Kontextinformationen, hier mit ζ bezeichnet, wie *hell* oder *dunkel* durch *Inferenzbildung* gewonnen. Durch den Schritt der Merkmalsextraktion wird die Datenmenge abermals verkleinert und für die Weiterverarbeitung im Entscheidungsfindungsprozess aufbereitet.

Die Entscheidungsfindung entspricht dabei einem Informationsfusionsprozess. In diesem Schritt wird aus den Kontextinformationen ζ der Kontext ξ bestimmt und mit einem Kontextbegriff c belegt. Dabei finden jeweils eine weitere Reduktion der Datenmengen und eine weitere Abstraktion statt. Eine Reihe von Fusions-Operationen steht für diesen Inferenz-Prozess zur Verfügung. Bekannte Verfahren sind Bayes'sche Netze, Kalman Filter oder Dempster-Shafer-Verfahren. Für die Fusionierung auf höherer Ebene sind insbesondere Ansätze aus dem Gebiet des *Soft Computings* wie Neuronale Netze oder Fuzzy Logik, wie sie in Abschnitt 6.5 ausgeführt sind, erfolgversprechend. Die Phasen der Merkmalsextraktion und der Entscheidungsfindung können dabei mehrmals durchlaufen werden. Dabei finden jeweils eine weitere Reduktion der Datenmenge und eine weitere Abstraktion statt.

4.3.3 Verteilung von Kontextinformationen

Die Entitäten, die an der Erfassung oder am Synthese-Prozess beteiligt sind, müssen nicht notwendigerweise auch die Informationen nutzen. Ein wichtiges Designprinzip bei der Erstellung von Kontext-bewussten Anwendungen ist die Trennung der Kontexterfassungsphase und der Nutzung der Kontexte. Daher müssen die Daten zwischen den *Produzenten* (*producern*) und den *Konsumenten* (*consumern*) der Kontextinformationen bzw. der Kontexte verteilt werden. Die generischen Begriffe Produzent und Konsument sind gewählt worden, da sie auf den unterschiedlichen Abstraktionsstufen der Kontextinformation gültig und dem *Entwurfsmuster* (*design pattern*) *Consumer-Producer* [Gra98] entlehnt sind.

Sind die beiden Prozesse auf einer Entität vereint, so kann der Austausch über lokale Interprozesskommunikation (*Interprocess Communication – IPC*) Konzepte wie Pipes-and-Filter [BMR+96] angewendet werden. Ist das Entwurfsziel eine verteilte Anwendung, so gilt diese Annahme nicht mehr und andere Interprozesskommunikationsverfahren müssen eingesetzt werden. Da es sich bei der Erzeugung und der Nutzung der Daten um einen asynchronen Prozess handelt, wurde ein *Publisher-Subscriber-Muster* [BMR+96] zur Verteilung gewählt. Es wird der Term *Daten* als generische Bezeichnung für Sensordaten, Informationen, Kontextinformationen und Kontexten gewählt.

Die Verteilung der Daten findet auf verschiedenen Abstraktionsebenen, zum einen in der Erfassungs- und Synthesephase, zum anderen in der Nutzungsphase statt. Beide Phasen werden jeweils getrennt voneinander betrachtet, da sie über unterschiedliche Anforderungen und Eigenschaften verfügen. Abbildung 4.10 zeigt den schematischen Aufbau des Verteilprozesses von Kontextmerkmalen, der aus einem Netzwerk an Sensoren und Aggregationskomponenten, einem zentralen Speicher für die Kontexte und einer Reihe Applikationen besteht, die den Kontext nutzen. Als konkrete Ausprägungen wurden ein *Kontextaggregationsnetzwerk* (*Context Aggregation Network – CAN*), welches in Abschnitt 7.2 ausführlich diskutiert wird, ein *ContextServer* als zentrale Integrationskomponente, der in Abschnitt 9.4 behandelt wird, sowie mehrere Anwendungen in Form von Kontext-bewussten Kommunikationsdiensten entwickelt und umgesetzt.

Als Struktur des Aggregationsnetzwerks wurde die Struktur eines gerichteten azyklischen Graphens gewählt. Zur Verteilung der Daten im Aggregationsnetzwerk wurde eine ereignisorientierte (*event driven*) Kommunikation eingesetzt. Eine Reduktion und Aggregation der Ereignisse wird durch Operatoren erreicht, die zwischen den Sensoren als primäre Produzenten und den ContextServern, als finale Konsumenten des Aggregationsnetzwerks liegen. Hierdurch können die Anforderungen an die Rechenleistung und insbesondere die Speichergröße gering gehalten werden, da die Operatoren überwiegend in einem zustandslosen Modus agieren. Für das Routing hält jeder Knoten nur eine Liste aller Nachbarn, an die er die Daten stromaufwärts (*upstream*) sendet. Das Versenden der Daten erfolgt als ein *Push*-Dienst.

Für die Verteilung der Kontexte wurde ein komplexeres Kommunikationsmodell verwendet, das die Daten nur bei Bedarf an die Kontext-Konsumenten versendet. Dadurch

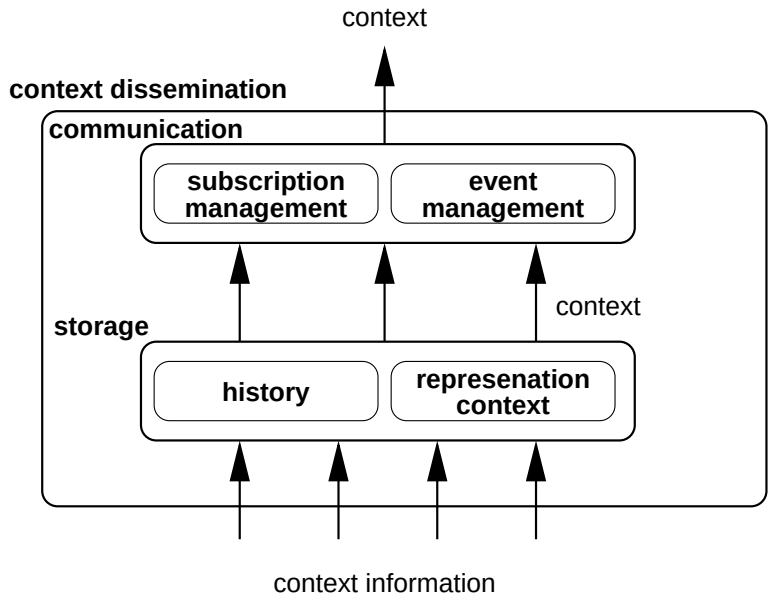


Abbildung 4.10: Verteilung von Kontexten

kann zum einen Bandbreite gespart werden und zum anderen wird die Belastung der Clients reduziert, da sie nicht aus der Gesamtmenge an Daten die Daten herausfiltern müssen, die für sie interessant sind. Ein Publish/Subscribe-Mechanismus wurde gewählt, um die erforderlichen Bedingungen zu erfüllen. Hierbei subskribieren sich die Anwendungen auf einen bestimmten Nutzer-Kontext. Ändert sich der Kontext des Nutzers (z. B. *Besprechung* → *Mittagspause*), so werden alle interessierten Anwendungen benachrichtigt, dass sich eine Änderung ereignet hat. Die Server-Applikation muss zu diesem Zweck eine Liste aller aktuellen Subskriptionen und der dazugehörigen Anwendungen speichern. Dies verschiebt die Belastung von den Clients zu den Servern, was für leichtgewichtige Anwendungen ein wichtiges Entwurfskriterium ist.

4.3.4 Nutzung von Kontext

Die Kontexte werden in einem oder mehreren zentralen Instanzen, auf welche die Anwendungen zugreifen können, vorgehalten. Im entwickelten System wurde diese Aufgabe vom ContextServer übernommen. Dieser stellt die Kontexte den Anwendungen über eine standardisierte Schnittstellentechnologie in Form von Web Services, wie sie in Unterabschnitt 7.4.3 beschrieben werden, zur Verfügung. Die Anwendungen erhalten die Kontexte über *Notifikationsnachrichten* in einem wohldefinierten Format, das in Unterabschnitt 7.5.2 näher beschrieben ist. Die Anwendungen können den empfangenen Kontext nutzen, um weitere Handlungen auszuführen. Zu diesen gehören die Anwendung des Kontexts zur:

Auslösung: Weitere neuer Kommunikationsdienste, wie diese beispielsweise zur Realisierung eines nahtlosen Kommunikationspfades notwendig sind, werden ausgelöst.

Hierbei kann eine Umwandlung der ursprünglichen Kommunikationsform mittels Gateways in ein anderes Protokoll oder in ein anderes Medienformat (z. B. Speech-to-Text) vorgenommen werden. Es handelt sich dabei um neue Kommunikationsbeziehungen, die aber eine logische Zuordnung zur selben Sitzung besitzen.

Adaption: Die Kommunikationsdienste, die auf der Entität ausgeführt werden werden durch eine Kontext-bewusste Parametrisierung adaptiert. Die aktivierten Dienste passen sich in ihrer Ausführung dem aktuellen Kontext an, um so den Nutzern einen effektiveren Dienst bieten zu können. Ein Beispiel hierfür sind CPL-Skripte, wie diese in Abschnitt 8.2 beschrieben werden.

Reduktion: Die eingehenden Kommunikationsanfragen, die der Nutzer manuell bearbeiten muss, werden durch eine Kontext-bewusste Anrufsteuerung reduziert. Die Reduktion wird durch eine Filterung erreicht, die in Abhängigkeit des gegenwärtigen Kontexts und auf der Basis von definierten Regeln Anrufumleitungen ausführt, wie diese in Unterabschnitt 3.3.1 vorgestellt wurden.

4.4 Zusammenfassung

Die Verwendung von Kontexten wurde in Kapitel 3 als ein Schlüsselkonzept für die Erbringung von effizienten Kommunikationsdiensten identifiziert. In diesem Kapitel wurden Definitionen des Begriffs Kontext aus unterschiedlichen, relevanten Disziplinen aufgeführt. Aus den Eigenschaften des Kontexts wurde eine eigene Kontextdefinition aufgestellt, aus der eine geeignete Kontextmodellierung hergeleitet worden ist. Hierbei wurde insbesondere die Möglichkeit der Komposition von Kontexten aus Kontextmerkmalen sowie ihrer zeitliche und räumliche Relevanz herausgearbeitet. Das entworfene Kontext-Spiral-Modell beschreibt die Prozessphasen, die notwendig sind, um einen Kontext aus „Messung“ der Kontextmerkmale und deren Komposition zu gewinnen. Die Phasen Erfassung, Synthetisierung, Verteilung und Anwendung werden konzeptionell beschrieben und bilden die Grundlage für das Design und die Architektur eines Frameworks für die Erstellung und die Bereitstellung von Kontext-bewussten Kommunikationsdiensten.

Kapitel 5

Framework für Kontext-bewusste Kommunikationsdienste

Wer hohe Türme bauen will, muss
lange beim Fundament verweilen.

Anton Bruckner

Für die Erstellung, die Ausführung und das Management von Kontext-bewussten Diensten ist in dieser Arbeit ein Framework für den Service Engineering-Prozess Kontext-bewusster Kommunikationsdienste vorgestellt worden. Die herausgearbeiteten Konzepte, Modelle und die Implementierung erfolgten dabei unter einem System-orientierten Ansatz. Das Framework soll Entwicklern die notwendige Unterstützung bei der Realisierung von Kontext-bewussten Diensten bereitstellen. Die entworfene Systemarchitektur und die Kommunikationsinfrastruktur unterstützen dabei umfassend alle Phasen des im Kontext-Spiral-Modell beschriebenen Prozesses. Um eine Einordnung des zu entwerfenden Systems zu ermöglichen, werden Eigenschaften und Klassifikationen aus dem Gebiet der Kontext-bewussten Anwendungen (Context-aware Computing) betrachtet.

Kurzübersicht

Die Struktur des vorliegende Kapitel ist wie folgt: In einer Anwendungsfallanalyse werden in Abschnitt 5.1 für das zu entwerfende System für Kontext-bewusste Dienste die funktionalen Prozesse identifiziert. Eine Diskussion alternativer Architekturformen und verwandter Arbeiten wird in Abschnitt 5.2 geführt. Die für die weitere Betrachtung gewählte verteilte Architektur wird ebenfalls beschrieben. In Abschnitt 5.3 werden Kommunikationsinfrastrukturansätze identifiziert und bewertet. Der eigene Ansatz wird dargelegt und in Abschnitt 5.4 in das Gesamtsystem eingeordnet. Abbildung 5.1 visualisiert die Struktur und die Beziehung der Abschnitte des Kapitels.

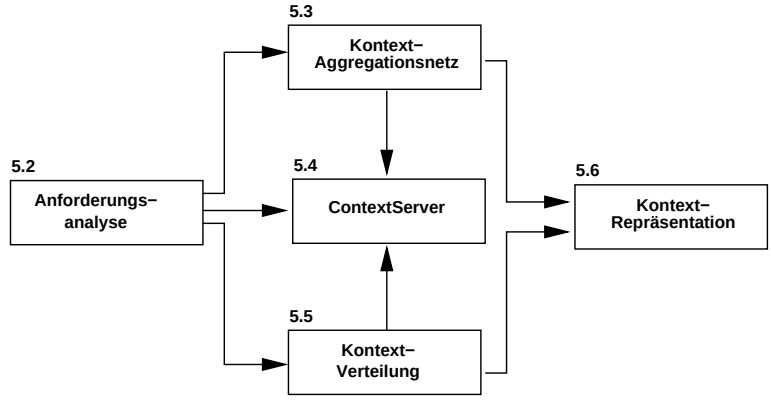


Abbildung 5.1: Graphische Darstellung der Struktur des Kapitel 5

5.1 Anforderungsanalyse des Systems

Das in Kapitel 4 aufgestellte *Kontext-Spiral-Modell* sieht die Trennung des Prozesses der Kontextgewinnung in vier prinzipiellen Phasen vor. Diese bilden die Grundvoraussetzungen für die Bereitstellung von Kontexten und Kontextinformationen an andere Dienste. Darüber hinaus beeinflussen sie die in Abschnitt 5.2 geführte Diskussion der geeigneten Architektur des Systems. Weitere vom System geforderte Funktionalitäten sind in einer Anwendungsfalldiagrammanalyse aufgeführt worden. Das zu entwerfende Framework für die Erstellung von Kontext-bewussten Diensten soll die identifizierten Anwendungsfälle unterstützen können.

5.1.1 Anwendungsfalldiagrammanalyse

Das Anwendungsfalldiagramm (*use case diagram*) wurde als standardisiertes Beschreibungsmittel zu einer ersten Anforderungsanalyse angewendet. Die wesentlichen Funktionsblöcke eines Kontext-bewussten Kommunikationssystems sind in Abbildung 5.2 aufgeführt. Die einzelnen Anwendungsfälle werden nachfolgend kurz beschrieben.

Diensterstellung (*Develop Services*): Die Entwicklung und der Nutzer sollen bei der Erstellung von Kommunikationsdiensten unterstützt werden. Dazu soll der Kontext als vorhandenes Informationsobjekt wahrgenommen werden. Die Einbeziehung des Kontexts soll die Kommunikationsbeziehung und die Kommunikation vereinfachen bzw. auf die Bedürfnisse des Nutzers anpassen.

Dienstparametrisierung (*Parametrize Services*): Vorhandene Dienste sollen vom Nutzer parametrisiert werden, so dass sie sich an die gewünschten Vorgaben anpassen können. Die Parametrisierung kann den Kontext einschließen.

Kommunikationsfilterung (*Filter Communication*): Das System soll dem Nutzer Möglichkeiten geben, seine Kommunikationsbeziehungen zu steuern. Die Filterung wie

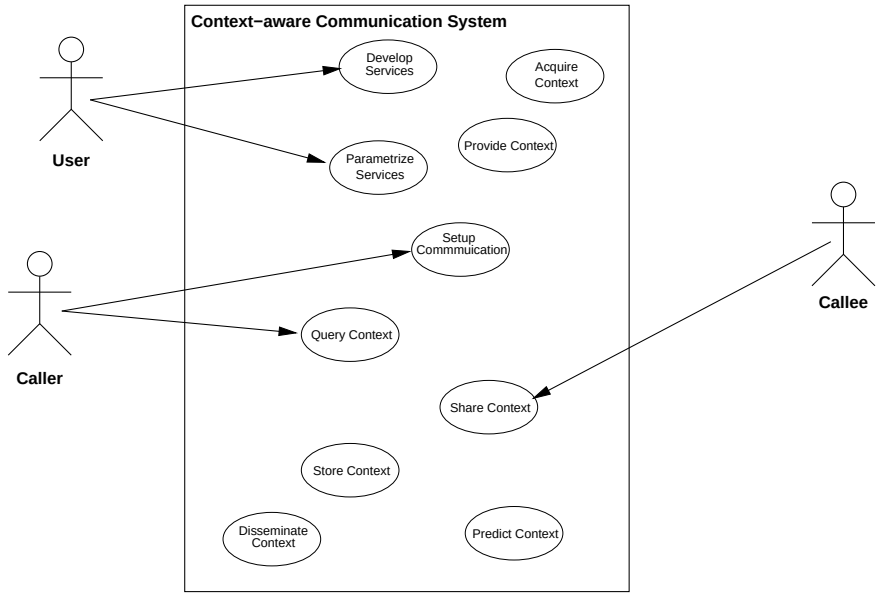


Abbildung 5.2: Anwendungsfalldiagramm für ein Kontext-bewusstes Kommunikationssystem

sie in Unterabschnitt 3.3.1 vorgeschlagen worden ist, ist eine mögliche Anwendung.

Aushandlung (*Negotiate*): Eine Aushandlung der Kommunikationszeitpunkte, wie mit dem Kommunikationsbroker in Unterabschnitt 3.3.3 beschrieben, soll vom System unterstützt werden. Dies erleichtert die Kommunikation.

Kommunikationsaufbau (*Setup Communication*): Der Nutzer soll in die Lage versetzt werden, im System Kommunikationsbeziehungen zu den Teilnehmern aufzubauen. Dies kann mittels unterschiedlicher Kommunikationstypen und -protokolle erfolgen.

Kontexterfassung (*Acquire Context*): Eine automatische Erfassung von Kontexten ist eine Grundvoraussetzung für Kontext-bewusste Dienste. Das System muss eine entsprechende Funktionalität zur Verfügung stellen.

Kontextspeicherung (*Store Context*): Das Vorhalten von aktuellen aber auch zeitlich älteren Kontexten ist eine wichtige Funktion für selbstlernende Systeme, die Kontextvorhersage und die Kontextverteilung.

Kontextbereitstellung (*Provide Context*): Das System muss anderen Anwendungen und Systemen Kontexte und Kontextinformationen bereitstellen.

Kontextverteilung (*Disseminate Context*): Der erfasste und optional gespeicherte Kontext muss an die Kontextnutzer verteilt werden können. Dies kann mittels unterschiedlicher Kontextverteilmechanismen erfolgen.

Tabelle 5.1: Zuordnung der Anwendungsfälle (use cases) zu den Typen von Kontext-bewussten Anwendungen

Use Case	Typ 1	Typ 2	Typ 3	Typ 4	Typ 5	Typ 6
Develop Services	✓	✓				
Parametrize Services				✓	✓	
Filter Communication	✓			✓		
Negotiate	✓	✓		✓		
Setup Communication						
Acquire Context		✓				
Store Context			✓			
Provide Context		✓				
Disseminate Context		✓				✓
Query Context		✓				✓
Share Context		✓				✓
Predict Context			✓	✓		
Self-learning			✓	✓	✓	

Kontextabfrage (*Query Context*): Ein Nutzer oder eine Anwendung muss in der Lage sein, den aktuellen Kontext einer anderen Entität abzufragen.

Kontextaustausch (*Share Context*): Ein Nutzer soll vom System Methoden zur Verfügung gestellt bekommen, seinen eigenen Kontext an interessierte Gesprächspartner mitzuteilen.

Kontextvorhersage (*Predict Context*): Aus den gespeicherten Kontexten und Modellen soll das System einen möglichen zukünftigen Kontext vorhersagen können, damit Anwendungen sich bereits auf diesen einstellen können.

Selbstlernen (*Self-learning*): Die manuelle Erstellung Kontext-bewusster Dienste oder die Parametrisierung der Kontexterfassung sind aufwändige und fehleranfällige Verfahren. Ein Selbstlernendes System kann den Nutzer entlasten und sich automatisch an Veränderungen anpassen.

Die Anforderungen einer Vielzahl an Anwendungsfällen decken sich mit der oben aufgeführten Klassifikation von Kontext-bewussten Anwendungen. Tabelle 5.1 stellt die Übereinstimmung mit den Typen, wie diese in Unterabschnitt 4.1.2 aufgeführt sind, und den Anwendungsfällen dar. Die Typen sind auf das neue Anwendungsgebiet übertragen worden. Es lässt sich daraus folgern, dass es sich bei dem vorgeschlagenen Gesamtsystem um ein Kontext-bewusstes System handelt.

5.2 Systemarchitekturen für Kontext-bewusste Dienste

Zur Erbringung der vorgestellten Kontext-bewussten Kommunikationsdienste wurden drei wesentliche Bereiche identifiziert. Die drei Bereiche sind die *Kontextgewinnung*, die

Kontextspeicherung- und *verteilung* sowie die *Kontextnutzung*. Für alle drei Bereiche sind in dieser Arbeit die notwendigen Anforderungen für einen Lösungsansatz herausgearbeitet worden. Existierende Arbeiten wurden jeweils identifiziert, bewertet und berücksichtigt. Es wurde dabei ein systemorientierter Ansatz gewählt, der alle Phasen abdeckt und unterstützt.

Um Kontexte in Applikationen anwenden zu können, müssen diese erfasst und in eine verwendbare Repräsentationsform umgewandelt werden. Diese Schritte erfolgen während der Kontextgewinnung. Aus den in Unterabschnitt 4.1.1 aufgeführten Definitionen ergibt sich, dass sich ein Kontext nicht direkt „messen“ lässt. Es müssen demnach Mechanismen eingesetzt werden, die aus erfassbaren Informationen den gegenwärtigen Kontext eines Nutzers oder eines Objekts erfassen können. Allgemein hat sich die Verwendung von Sensoren zur Gewinnung von Informationen über die Umgebung etabliert [Dey00, Sch02b].

Die erfassten Kontexte werden von den Anwendungen genutzt, um ihr Verhalten anzupassen oder neue Aktionen anzustoßen. Dazu muss der Kontext in einer maschinenlesbaren Repräsentation vorliegen und den Anwendungen zugänglich gemacht werden. Erfasste Kontexte sollen gespeichert werden können, um Auswertungen über den Verlauf des Kontexts eines Nutzers durchführen zu können. Diese Informationen können genutzt werden, um Muster in Kontexten zu finden, was wiederum Aussagen über mögliche Kontexte in näherer Zukunft ermöglicht. Darüber hinaus soll der Kontext zwischen unterschiedlichen Anwendungen ausgetauscht werden, wie dies beispielsweise für den in Unterabschnitt 3.3.2 beschriebenen Ansatz notwendig ist.

5.2.1 Architekturvarianten

Die Erfassung, Gewinnung, Verteilung und Nutzung von Sensordaten sowie Kontexten kann in unterschiedlichen Architekturen erfolgen. Einige grundlegende Ansätze wurden identifiziert, betrachtet und verglichen. Nachfolgend werden prinzipielle Systemarchitekturen mit ihren Eigenschaften vorgestellt. Diese werden hinsichtlich ihrer Verwendbarkeit für Kontext-bewusste Dienste bewertet. Anschließend werden verwandte Arbeiten auf dem Gebiet der Architekturen für Kontext-bewusste Systeme dargestellt.

Aus der Analyse der vorgestellten Architekturen und weiteren Rahmenbedingungen wurde die für die Arbeit ausgewählte eigene Architektur abgeleitet. Da es sich um ein heterogenes Umfeld bezüglich der verwendeten Sensoren sowie der eingesetzten Endgeräte und der Kommunikation handelt, gibt es keine globale Architektur, die für jede Konstellation optimal ist. Beim Entwurf der Architektur wurde auf bekannte Muster aus dem Software Engineering zurückgegriffen. Abschließend wird die konkrete Architektur vorgestellt.

5.2.1.1 Monolithen

Ein Ansatz, der alle Funktionalität und Ausführlogik in einer einzelnen Entität vereint, ist in Abbildung 5.3 dargestellt. Bei dieser *monolithischen* Architektur sind die

funktionalen Bausteine Aggregation und Entscheidungsfindung kombiniert. Auch die notwendigen Sensoren sind in die Entität integriert und liefern die Daten direkt an den Aggregationsprozess. Beispielsweise ist ein PDA ein typisches Endgerät, das sich gut mit einer monolithischen Architektur beschreiben lässt, da es oftmals eine kleine Anzahl an Sensoren mit sich führt, die in den Aggregations- und Entscheidungsprozess einbezogen werden. Der PDA ist damit in der Lage, eine intrinsische Kontextgewinnung durchzuführen. Auch bietet sich hier eine Kommunikation auf Peer-to-Peer (P2P)-Basis an, wie es in Unterabschnitt 5.3.1.1 beschrieben ist, um mit anderen Geräten in der Nähe ad-hoc Informationen auszutauschen. Dadurch können Sensordaten bezogen werden, die das Gerät nicht erfassen kann oder die Daten können abgeglichen werden, um Ungenauigkeiten zu reduzieren oder zu kompensieren.

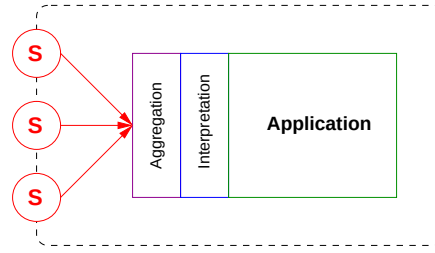


Abbildung 5.3: Monolitische Architektur

5.2.1.2 Mediatoren

Eine Verteilung der Funktionalität führt zu einer *Mediator*-Architektur, die in Abbildung 5.4 aufgezeigt ist. Als Mediatoren können Operatoren dienen, die in den jeweiligen Phasen der Aggregation die Daten entgegennehmen, bearbeiten und weiterleiten. Die Anwendung erhält dann Kontextinformationen ζ, die in der Entscheidungsfindung interpretiert werden können. Durch ein Platzieren der Mediatoren in die Nähe der Datenquellen kann eine Reduktion des Netzwerkverkehrs erreicht werden. Zusätzlich wird die Anwendung von der Kommunikation mit den Sensoren entlastet. Die Mediatoren können darüber hinaus auch die spezifischen Protokolle und Formate der einzelnen Sensoren kapseln und so die weitere Verarbeitung in der Anwendung vereinfachen.

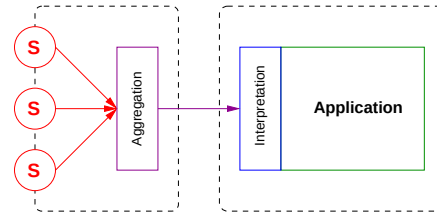


Abbildung 5.4: Architektur mit einem Mediator

5.2.1.3 Proxies

Eine noch stärke Verteilung der einzelnen Funktionsblöcke ist in Abbildung 5.5 dargestellt. Dezentrale *Proxies* übernehmen die Entscheidungsfindung und liefern die Kontexte c an die jeweiligen Anwendungen. Diese Anordnung entspricht dem Proxy-Entwurfsmuster aus [BMR+96], welches eine höhere Wiederverwendbarkeit und eine Vereinfachung der Anwendungen bietet. Der in Abschnitt 9.4 vorgeschlagene ContextServer implementiert eine solche Proxy-Architektur. Der Server stellt die jeweils aktuellen Kontexte über eine definierte Schnittstelle anderen Anwendungen zur Verfügung. Über Remote Procedure Call (RPC)-Mechanismen kann auf die Methoden zur Speicherung, Manipulation und Beziehen der Daten zugegriffen werden. Bei der konkreten Realisierung wurden Web Services eingesetzt. Mit dieser Architektur kann das Entwurfskriterium der Trennung von Kontexterfassung und Kontextnutzung erreicht werden. Dies erlaubt es, den Anwendungsentwickler von der eigenen Realisierung eines Prozesses zur Erfassung der Synthetisierung zu entlasten. Vielmehr kann der Entwickler auf die einzelnen Kontextobjekte zugreifen, wie auf andere Informationsobjekte auch.

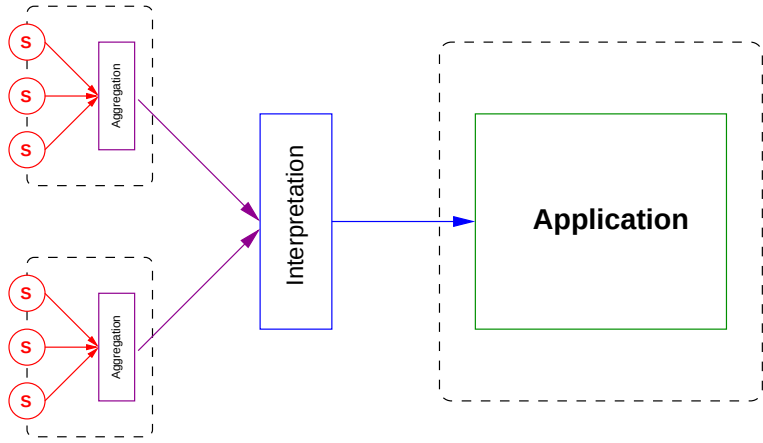


Abbildung 5.5: Architekturvariante mit Proxies

5.2.2 Verwandte Arbeiten: Architekturunterstützung

Eine Vielzahl an Architekturen oder Frameworks für die Umsetzung von Kontextbewussten Anwendungen wurde vorgeschlagen. Bekannte, aber hier nicht weiter betrachtete Ansätze sind Muse [CM00], CoolTown [KBM+00, CD00], CALAIS [Nel98a], Location-Awareness [LMD96, Leo98], Context Information Service (CIS) [Pas98, Pas01], SOLAR [CK02a]. GUIDE [Mit02]. Nachfolgend werden oft zitierte und wegbereitende Ansätze vorgestellt.

5.2.2.1 Mobile Context-Aware Computing

Das in [Sch95] vorgestellte System zur Unterstützung von Kontext-bewussten mobilen Anwendungen basiert auf den Arbeiten am PARCTAB Mobile Computing System [SAG+93, AGS+93]. Unter Kontext wird im Wesentlichen die Gesamtheit der Informationen über Personen (z. B. Status, Präferenzen, etc.) oder über Beziehungen zwischen Objekten und Nutzern verstanden. Dies zeigt sich darin, dass in der Systemarchitektur jeder Nutzer von einem *User Agent* und jedes Artefakt von einem *Device Agent* repräsentiert wird. Der User Agent kann vom Nutzer weitreichend konfiguriert werden, um beispielsweise seine Privatsphäre zu schützen.

Der Fokus der adressierten Anwendungen sind *Proximity-Dienste*, die Informationen zu Geräten in der Nachbarschaft liefern. Dazu stellen die Geräte bei Veränderung ihres Zustands dem Active Map Service (AMS) Informationen zur Verfügung. Dieser Dienst kennt die globale Kenntnis der Geräte und ihrer räumlichen Zuordnung. Von diesem können andere Anwendungen Informationen beziehen. Die zentrale Struktur ist für lokale Anwendungen gut geeignet, skaliert aber nicht für weitverteilte Systeme und große Mengen an Kontextinformationen.

5.2.2.2 Context Toolkit

Das *Context Toolkit* ist ein umfassendes Framework, das aus dem Context Server [SA98] hervorgegangen ist und die Erstellung von Kontext-bewussten Anwendungen unterstützen soll [SDA99, DSA99, DA00, Dey00, DSA01]. Ein wichtiges Merkmal ist die Separation von Kontexterfassung und Kontextnutzung. Die Architektur des Context Toolkits besitzt Anleihen der Konzeption von graphischen Benutzerschnittstellen (GUI) und besteht aus den drei Hauptkomponenten: Context Widgets zur Erfassung von Daten, Context Interpreters zur Umwandlung von Rohdaten in komplexere Informationen und der Context Aggregation zur Verbindung von Kontexten und Entitäten.

Die Applikationen melden sich bei den Widgets und werden per Callback-Mechanismus über Ereignisse, die Kontexte repräsentieren, benachrichtigt. Ein Datenschutzkonzept stellt die Sicherheit der Privatsphäre der Nutzer sicher. Das Context Toolkit implementiert ein dienstorientiertes Paradigma mit zentralen Komponenten, was zu Skalierungsproblemen bei großen Datenmengen führen kann.

5.2.2.3 Technology for Enabling Awareness (TEA)

Das Technology for Enabling Awareness (TEA)-Projekt [SAT+99, Sch02b] stellt eine Architektur bereit, die die Umwandlung von Sensordaten in Kontexte, auf die Anwendungen zugreifen können, unterstützt. Die Architektur besteht aus den vier Schichten *Sensors*, *Cues*, *Context Profiles* sowie *Application and Scripting*. Auf der untersten Schicht werden low-level Informationen aus Sensoren gewonnen. Diese Daten werden dann in der Cue-Schicht gefiltert und gegebenenfalls umgewandelt, um eine einfachere Interpretation der Daten zu ermöglichen und um die Datenmenge zu reduzieren.

Das Abbilden der Kontext-Cues auf die Kontextprofile erfolgt anhand von festen Regeln, wie „Licht = 85% UND Lautstärke = 60% → Meeting“. Auf der obersten Schicht werden die Kontexte dann verwendet, um Anwendungen zu parametrisieren. Kontexte werden dabei als Daten-Tupel repräsentiert. Jedoch werden keine Programmierschnittstellen zu Verfügung gestellt, die Anwendungsentwickler bei der Erstellung der Kontext-bewussten Programme unterstützen.

5.2.3 Konkrete Architektur zur Unterstützung von Kontext-bewussten Kommunikationsdiensten

Die Verteilung der Funktionalitäten, wie sie sich bei der Anwendung des Proxy-Entwurfsmuster ergibt, erlaubt eine hohe Flexibilität und eine gute Lastverteilung. Darüber hinaus kommt sie der identifizierten Teilung in die Phasen der Kontexterfassung, -synthetisierung, -verteilung und -nutzung entgegen. Die einzelnen Phasen werden oftmals an verschiedenen Lokalitäten ausgeführt. So kann der Kontext von einer Anwendung genutzt werden, die räumlich von den Sensoren weit entfernt ist. Auch die einzelnen Sensoren sind in der Regel räumlich verteilt, was für viele zu messende Phänomene, wie z. B. die Lokation auch gefordert ist.

Die Zuordnung der Prozessschritte des Kontext-Spiral-Modells aus Abschnitt 4.3 auf die Architektur ist in Abbildung 5.6 zu sehen. Die Erfassungsphase wird auf ein *Sensoraggregationsnetz* abgebildet, welches aus einer Menge an verteilten und über ein Netzwerk verbundenen Sensoren besteht. Die Aggregation der Sensordaten zur Reduktion der Datenmengen und die Merkmalsextraktion, wie sie in den Prozessblöcken in Abbildung 4.8 und Abbildung 4.9 gezeigt ist, soll ebenfalls vom Sensoraggregationsnetz bereitgestellt werden.

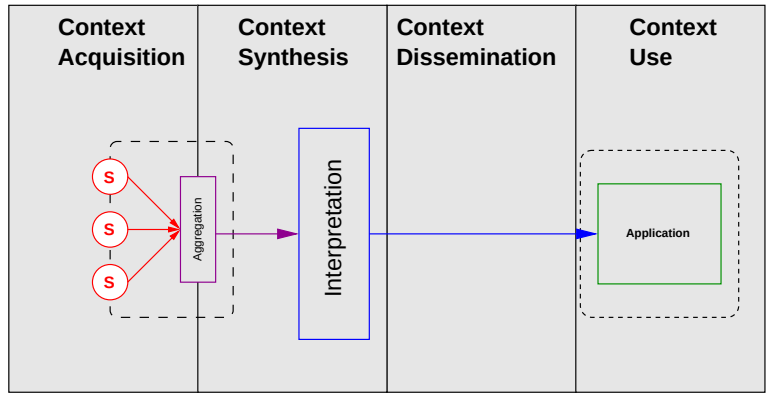


Abbildung 5.6: Abbildung der Architektur auf auf Kontext-Spiral-Modell

Die Synthetisierungsphase soll sowohl durch die Operatoren des Sensoraggregationsnetz realisiert werden können, als auch in einer oder mehreren zentralen Instanzen, in denen die Daten interpretiert und durch Entscheidungsfindungsverfahren auf Kontexte abgebildet werden. Der ContextServer wird als eine solche zentrale Instanz vorgeschlagen, in

der auch die im Funktionsblock der Kontextverteilung aus Abbildung 4.10 aufgeführten Prozesse der Speicherung und Verteilung von Kontexten realisiert sind.

Die Anwendungen sind vom ContextServer getrennt und erhalten die Kontexte über das Netzwerk. Die Zahl der ContextServer ist wesentlich kleiner als die Anzahl der Applikationen und Dienste. Die ContextServer stellen die Schnittstelle zwischen den Anwendungen und den Sensoren dar. In einigen Anwendungsfällen können die Komponenten, wie in Abbildung 5.3 gezeigt, auf einer Entität zusammenfallen. Dies ist zum Beispiel bei mobilen Geräten wie PDAs oder Mobiltelefonen der Fall. Für die in dieser Arbeit adressierten Dienste wird eine *verteilte* Architektur vorgeschlagen. Da die Komponenten verteilt sind, muss eine *Kommunikationsinfrastruktur* bzw. *Kommunikationsmiddleware* für den Austausch der Informationen sorgen. Verschiedene Varianten werden im nächsten Abschnitt behandelt.

5.3 Kommunikationsinfrastruktur

Die gewählte Architektur aus Abschnitt 5.2 erfordert die Bereitstellung einer Kommunikationsinfrastruktur. Es können dabei drei unterschiedliche Bereiche, in denen eine Kommunikation erforderlich ist, identifiziert werden. Zum einen zwischen den Sensoren, den Aggregationsoperatoren und dem ContextServer. Des Weiteren zwischen dem ContextServer und den einzelnen Anwendungen und Diensten. Zum anderen zwischen den Kommunikationsdiensten, um eine Verteilung von Kontexten zur Vermeidung von Anrufen, wie dies in Unterabschnitt 3.3.2 beschrieben worden ist, zu ermöglichen.

Da die drei Bereiche unterschiedliche Anforderungen an die Kommunikationsinfrastruktur haben, wurde jeweils ein spezieller Ansatz verfolgt, der die geforderten Funktionalitäten erfüllt. Dieser muss jedoch nicht für die anderen Bereiche geeignet sein. Nachfolgend werden Kommunikationsparadigmen vorgestellt. Anschließend werden relevante und verwandte Arbeiten im Bereich Kommunikationsinfrastrukturen mit dem Schwerpunkt auf Kontext-bewusste Anwendungen diskutiert. Abschließend wird die entworfene eigene Kommunikationsinfrastruktur vorgestellt und beschrieben.

5.3.1 Kommunikationsinfrastrukturen

Verschiedene *Kommunikationsarchitekturen* und *Datenverteilmechanismen* existieren für verteilte Systeme. Diese werden nachfolgend kurz aufgeführt. Zwei generelle Architekturen können unterschieden werden. Zum einen sind dies das Client/Server-System und zum anderen das Peer-to-Peer-System. In Abbildung 5.7 sind die beiden Modelle dargestellt.

5.3.1.1 Kommunikationsarchitekturen

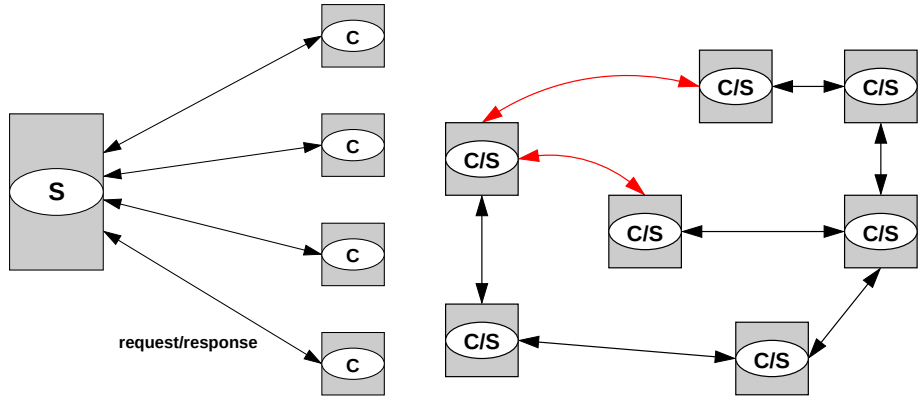
Das Client/Server-Konzept wie in Abbildung 5.7(a) gezeigt, ist eine sehr häufig umgesetzte Kommunikationsarchitektur, bei der ein (leistungsfähiger) Server (S) Daten oder

Dienste bereithält, die vom Client (C) über eine Anfrage (*request*) abgefragt werden können. Die Ergebnisse werden als Antworten (*reponse*) an den Client zurückgeschickt. Eine Abwandlung ist die Verwendung eines Proxies zwischen den Clients und dem Server.

Das Peer-to-Peer-Paradigma, wie in Abbildung 5.7(b) dargestellt, ist eine Abkehr von dem seit vielen Jahren eingesetzten Client/Server-Paradigma, bei dem gleichrangige Entitäten (*peers*) Daten direkt miteinander austauschen. Die zentrale Struktur der Client/Server-Architektur skaliert in seiner einfachen Form bei einigen aktuellen Anforderungen nicht mehr ausreichend. Insbesondere besteht die Gefahr eines Single-Point-of-Failure hinsichtlich der Ressourcen und bei Angriffen aus dem Netzwerk auf den zentralen Punkt.

Ein Peer-to-Peer-System wird in [SW04] definiert als:

Definition 5.1 (Peer-to-Peer System)
*Ein **Peer-to-Peer-System** wird als ein sich selbst organisierendes System gleichberechtigter, autonomer Einheiten (Peers) verstanden, das vorzugsweise ohne Nutzung zentraler Dienste auf der Basis eines Rechnernetzes mit dem Ziel der gegenseitigen Nutzung von Ressourcen operiert.*



(a) Client/Server Kommunikationsarchitektur

(b) P2P Kommunikationsarchitektur

Abbildung 5.7: Kommunikationsarchitekturen

5.3.1.2 Datenverteilmechanismen

Die Daten in den Kommunikationsarchitekturen können auf unterschiedliche Weise verteilt werden. Eine Einordnung der Verfahren wurde in [AAB+98] durchgeführt und ist in Abbildung 5.8 abgebildet. Die wesentlichen Unterscheidungsmerkmale sind die zeitlichen Sendeoptionen *periodisch* (*periodic*) und *aperiodisch* (*aperiodic*), die Art der

Beziehung (*Unicast* oder *1-N*) und ob die Daten *aktiv* (*pull*) oder *passiv* (*push*) bezogen werden.

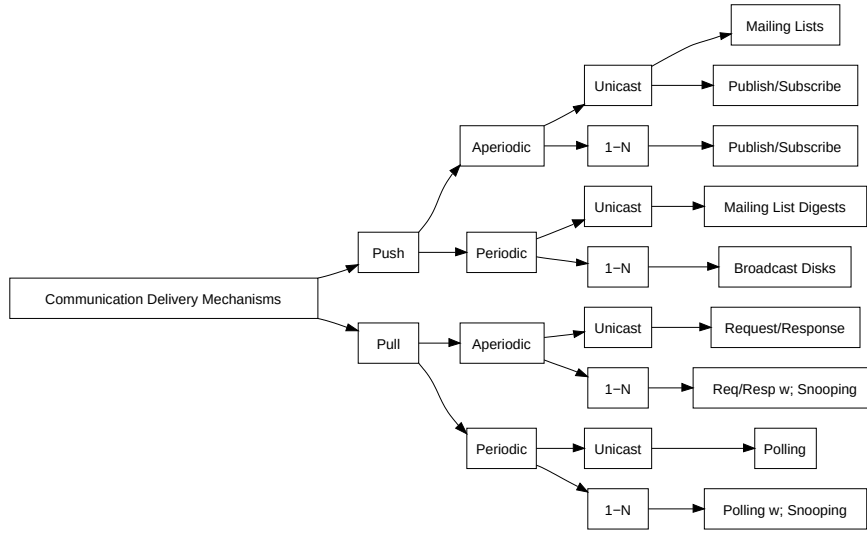


Abbildung 5.8: Einteilung der Datenverteilmechanismen

Zu einer Reihe dieser Konzepte werden bekannte Ansätze, die im Bereich Ubiquitous Computing eine Rolle spielen, vorgestellt. Insbesondere sind dies Client/Server-System, Peer-to-Peer-System, Shared Memory Space-System sowie Event-basierte Systeme. Zuvor werden die prinzipiellen Verfahren kurz erläutert.

Bei einem *Request/Response-Verfahren* werden die Daten von einer Entität, z. B. einem Client, wie in Abbildung 5.7(a), angefragt und von der angefragten Entität, z. B. einem Server gesendet. Das klassische Beispiel sind Browser, die Webseiten bei einem Webserver anfragen.

Ein *verteilter gemeinsam genutzter Speicher* ist ein alternativer Ansatz zu Nachrichtenaustauschverfahren. Dieser stellt eine Abstraktion für die gemeinsame Nutzung von Daten zwischen mehreren Prozessen dar. Die Prozesse können dabei lesend und aktualisierend auf den gemeinsamen Speicherbereich zugreifen. Der Speicherbereich erscheint den Prozessen als lokaler Speicher innerhalb ihres Adressraums. Abbildung 5.9 illustriert dieses Datenverteilmodell.

Das *ereignisgesteuerte Paradigma* ist für die Kommunikationsunterstützung von lose gekoppelten Komponenten in verteilten Systemen sehr geeignet. Die Architektur von Event-basierten Systemen skaliert auch für heterogene und große Umgebungen. Eine schematische Abbildung des Verfahrens ist in Abbildung 5.10 gegeben. Notifikationen (*notifications*) werden nur bei Eintreten eines Ereignisses (*events*) an diejenigen Empfänger gesendet, die für dieses Ereignis ein Abonnement (*subscription*) haben. Ein Ereignis kann jede observierbare Aktion sein. Die Notifikation wird vom Beobachter (*observer*) eines Ereignisses erzeugt und ist die Vergegenständlichung (*reification*) in Form einer Datenstruktur des zugehörigen Ereignisses. Die Daten kann der Client vom

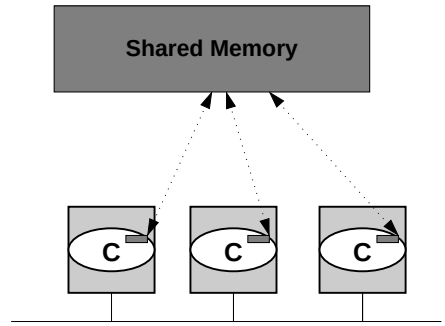


Abbildung 5.9: Datenverteilung über einen gemeinsam genutzten Speicher

Server abholen (*pull*) oder aber der Server verteilt die Informationen aktiv an die Clients (*push*).

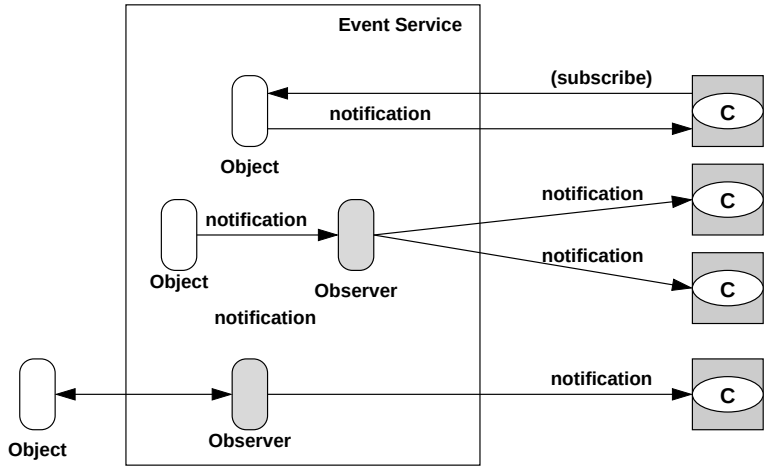


Abbildung 5.10: Ereignis-basierte Datenverteilung

5.3.2 Verwandte Arbeiten: Kommunikationsinfrastrukturen

Nachfolgend werden relevante verwandte Arbeiten im Umfeld des Ubiquitous Computing präsentiert. Die Ansätze sind nach der in Abbildung 5.8 vorgestellten Klassifizierung eingeteilt.

5.3.2.1 Shared Memory Space-Systeme: Tupelspaces und Blackboard Systeme

Ein Informationsverteilmechanismus, in welchem Prozesse über einen gemeinsamen Speicherbereich (*shared data space*), den *Tupelspace* (*tuple space*), kommunizieren ist *Linda* [CG89]. Bei diesem System werden Informationen in Tupeln, die in einem

assoziierten Speicherraum liegen und über einen Schlüssel adressiert werden können, ausgetauscht. Dies ist der wesentliche Unterschied zu Systemen, die (kurze) Nachrichten versenden. Linda kann als eine *koordinierende Sprache* (*coordination language*) aufgefasst werden, die verteilte Anwendungen verbinden kann. Das System wurde in verschiedenen Sprachen implementiert.

Der Nachteil des Tupelspace ist, dass es sich um einen einzigen gemeinsam genutzten Kommunikationskanal handelt (*single shared communication channel*), der schlecht skaliert. Ein komplexer Ansatz, dieses Problem zu lösen, wurde in [MP93] vorgestellt. Der Tupelspace wird dabei in eine Vielzahl rekursiver Tupelspaces aufgeteilt, die jeweils für die Koordination eines Kommunikationspaares zuständig sind. Eine zusätzliche Erweiterung des originären Ansatzes ist L^2imbo, das eine verteilte Plattform für Mobile Umgebungen zur Verfügung stellt [FDWB98].

Ein weiteres System, welches einen gemeinsamen Datenspeicher nutzt, sind der in [SG01] vorgestellte *FuzzySpace* sowie die *Shared Information Spaces* [BFKL04]. Bei dem vorgeschlagenen FuzzySpace-System handelt es sich um ein *Blackboardsystem*, das die Lebensdauer und Relevanz eines Objekts mittels Fuzzy Logik Mitgliedsfunktionen beschreibt. Nach Ablauf der Lebenszeit wird das Objekt aus dem System gelöscht. Dadurch kann nicht mehr auf ältere Objekte zugegriffen werden, was die Anzahl der gespeicherten Daten reduziert, eine Analyse aus der Historie, wie sie zur Kontextprädiktion genutzt werden kann, allerdings verhindert.

Der *Information Bus* [OPSS93] ist ein Ansatz, der ohne Callback-Subskriptionen auskommt. Bei diesen Bus werden alle Ereignisse und Änderungen per Broadcast verteilt. Ein Filtermechanismus erlaubt es den Anwendungen, nur die Ereignisse zu empfangen, die interessant für sie sind. Für die Anwendungen ergibt sich eine sehr einfache Handhabung, da sie keine aufwändigen Subskriptionsmechanismen implementieren müssen. Eine *1-N* Kommunikation über drahtlose Netze mittels eines periodischen Push-Mechanismusses ist *Broadcast Disks* [AAFSZ95].

Ein Nachteil dieser Klassen von Informationsverteilungssystemen ist die Limitierung auf lokal verteilte Systeme. Zum Auffinden der Tupel über den Schlüssel werden Broadcast-Verfahren angewendet. Bei Anwendung in einem weitverteilten System führt dies zu großen Nachrichtenmengen und ist oftmals nicht praktikabel.

5.3.2.2 Client/Server-Systeme

Das *Rover Toolkit* [JDGK95] stellt Anwendungen ein einheitliches verteiltes Objektesystem (*uniform distributed object system*) auf Grundlage einer Client/Server-Architektur zur Verfügung.

Mit Relocatable Dynamic Objects (RDO) und Queued Remote Procedure Call (QRPC) werden zwei Konzepte in Rover für die mobile Kommunikation bereitgestellt. Ein RDO ist ein dynamisches Objekt mit einer definierten Schnittstelle, das zwischen Server und Client ausgetauscht werden kann. Ein QRPC-Mechanismus verhindert, dass Clients

versuchen, non-blocking RPC-Aufrufe durchzuführen. Stattdessen werden die Anfragen gesammelt (*queued*) und bei Netzkonnektivität versandt.

Das neXus-Projekt verfolgt die digitale Repräsentation der realen Welt mittels Modellierung (*augmented area models*) [HKL+99, NM01]. Hierfür werden insbesondere aufwändige Lokationsmodelle [BBR02, DR03] entwickelt, die als Grundlage für die virtuellen Welten dienen. Der Anwendungsfokus in neXus beschränkt sich daher auch weitgehend auf ortsbezogene Dienste (*Location Based Services – LBS*), da Lokation als Hauptkontext betrachtet wird. Eine Erweiterung der Anwendungstypen ist mit der Bereitstellung des *ContextCubes* [BBHS03], der Temperatur, Luftfeuchtigkeit, Helligkeit und andere Phänomene erfassen kann, zu erwarten.

Änderungen werden in der Plattform mittels asynchroner Notifikationen propagiert und können weitere Aktionen in dem Modell auslösen. Anwendungen beziehen Orts-Informationen in neXus von *föderierten Datenbanken* (*federated databases*). Dazu setzten die Applikationen Anfragen (*queries*) ab, die von den Datenbanken empfangen und in Form von XML-Dokumenten beantwortet werden.

5.3.2.3 Event-basierte Systeme: Föderierte Nachrichtensysteme und Event/Publish-Systeme

Event-basierte Anwendungen werden bereits in großer Anzahl in kommerziellen Systemen eingesetzt und sind auch in einer Vielzahl von Forschungsansätzen [Müh02, Pie04, RKCD01] beschrieben. Nachfolgend werden relevante Systeme und Ansätze beschrieben, die im Umfeld des Ubiquitous Computing oder Pervasive Computing entstanden sind.

Die Architektur der am XeroxParc entwickelten Anwendungen [SAW94, Sch95] enthält ein Publisher/Subscriber-Kommunikationsmodell, welches die Informationen an die beteiligten Kontext-bewussten Anwendungen verteilt. Ein push-basierter Eventmechanismus, der Clients über Informationen bestimmter Themengebiete benachrichtigt, wurde in [FZ98] vorgeschlagen. Die *Cambridge Event Architecture* (CEA) [BBHM95, BBMS98, BMB+00] beschreibt ein verteiltes System für die aktive Distribution von Informationen. Ein Ereignis wird definiert als ein asynchrones Auftreten (*asynchronous occurrence*), das Details über die Aktivität, die im System eingetreten ist, und einen Verknüpfungspunkt (*index point*) zu den laufenden Anwendungen enthält. Die Architektur basiert auf einer synchronen Common Object Request Broker Architecture (CORBA)-Middleware und wurde um Event-Mediatoren erweitert, die zwischen den Produzenten und Konsumenten als Puffer sitzen, um eine enge Kopplung der beiden Elemente aufzuheben. Ein Repositorium zur Erfassung und Speicherung der auftretenden Ereignisse wurde ebenfalls definiert.

In [Zei04] wird die Middleware *Rebeca* für mobile Nutzer, die sich in Infrastrukturen bewegen, in der sich kommunizierende und interagierende Artefakte befinden, entworfen. Das statische *Request/Reply*-Paradigma ist für die dynamischen Umgebungen zu statisch und zu starr (*tight-coupled*). Die vorgeschlagene verteilte Event-basierte Publisher/Subscriber-Architektur, wie sie auch im Ansatz *Siena* [CRW00, CRW01]

zu finden ist, erlaubt eine zeitliche und räumliche Entkopplung (*decoupling*) von Produzenten und Konsumenten von Daten. Die inhärente Unsicherheit asynchroner und anonymer Kommunikation zwischen den Clients und dem System wird in diesem Ansatz durch Mechanismen, die eine Konsistenz der Daten durch Zugriff auf ältere Daten erreichen, verringert.

5.3.2.4 Peer-to-Peer

In [Che04] wird ein *Context Fusion Network* (CFN) vorgeschlagen, das Anwendungen bei der Erfassung, Aggregation und Verteilung von Kontextinformationen unterstützt. Das *Solar* genannte System besteht aus einem skalierbaren und selbst-organisierenden Overlay-Netzwerk – *Planets* – und Netzwerken aus Informationsflussgraphen (*directed acyclic information flow graph*). Der Graph verbindet die Sensoren mit deterministischen Operatoren. Diese Anordnung ist auch in vielen anderen Event-Kompositionsansätzen [LCB99, Cil02, PSB03] (*event composition*) zu finden. Unterschiedliche Subskriptionstypen sind in Solar definiert wie beispielsweise XML-Dokumente. Es bleibt jedoch unklar, wie diese unterschiedlichen Typen miteinander verbunden werden können. Die Operatoren liefern über Datafusionmechanismen Informationen an die Planets.

Die Planets verwenden ein Peer-to-Peer Routingprotokoll *Pastry* auf Basis von Distributed Hash Tables (DHT) [RD01]. Die Overlay-Architektur des Context Fusion Network (CFN) mit den Planets ist in Abbildung 5.11 dargestellt. Die Planets sind mit P bezeichnet und verbinden die Sensoren S und die Anwendungen A. Die Operatoren, die die Data-Fusion kooperativ ausführen sind als ausgefüllte Kreise • eingezeichnet. Die dezentrale Struktur der funktionsäquivalenten Server erlaubt eine hohe Skalierbarkeit und Flexibilität.

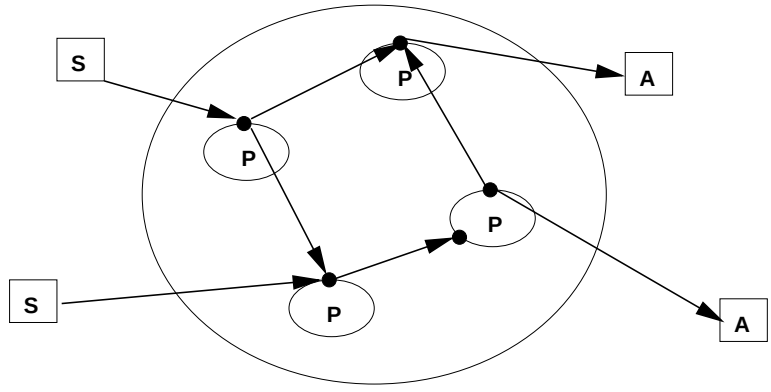


Abbildung 5.11: Architektur des Overlay-Netzes in Solar mit Planets

Ein weiterer Ansatz, der ein Event-basiertes Publish/Subscribe-System mit einem Peer-to-Peer Overlay-Ansatz verbindet, ist *Hermes* [PB02, Pie04]. Die Routing-Algorithmen verwenden DHTs für das Routing im Overlaynetz. Ein ähnlicher Ansatz findet sich auch in [TBF+03]. Das vorgeschlagene System kann durch das Peer-to-Peer-Netz eine

höhere Fehlertoleranz gewährleisten. Das System verwendet ein Typ- und Attributsbasiertes Subskriptions-Modell mit Typüberprüfung der Events und Vererbung der Eventtypen.

5.3.3 Gewählte Kommunikationsinfrastruktur

Im Rahmen dieser Arbeit wurde ein Konzept für eine Kommunikationsinfrastruktur, die die Gewinnung des Nutzerkontexts sowie die Verteilung des Kontexts für ein IP-Telefonie-System auf SIP-Basis bereitstellt, entworfen und umgesetzt. Hierbei wurde insbesondere auf die Berücksichtigung der Anforderungen eines Echtzeitkommunikationssystems geachtet. Zusätzlich werden Mechanismen eingesetzt bzw. erweitert, die im SIP-Umfeld bereits zum Einsatz kommen.

Das Gesamtsystem kann in Subsysteme zur automatischen Erfassung, Verteilung und Verwendung für Kommunikationsdienste unterteilt werden. Für die jeweiligen Teile werden unterschiedliche Netze vorgeschlagen, da die Teilbereiche unterschiedliche Eigenschaften und Anforderungen besitzen. Zur Erklärung der Wahl der Mechanismen wird ein Top-Down-Ansatz gewählt.

In IP-Telefonie-Systemen wird Kontext zwischen teilnehmenden SIP UAs ausgetauscht, wie beispielsweise in Unterabschnitt 3.3.2 für Dienste beschrieben wurde. Hierfür wurden zwei Signalisierungskonzepte entworfen und realisiert. Dies ist zum einen ein *transparenter* Modus, der sich direkt in den Standardrufaufbau mittels INVITE-Nachricht integriert und sich standardkonform gegenüber SIP-Entitäten verhält, die nicht die notwendigen Erweiterungen besitzen. Ein weiterer Modus erlaubt die *explizite* Anfrage von Kontextinformationen sowie die *Subskription* auf einen bestimmten Kontext unter der Verwendung von SUBSCRIBE/NOTIFY-Nachrichten.

Die Kommunikationsdienste im UA bzw. die CPL-Engine benötigen den Kontext innerhalb einer Zeitschranke t_{Δ}, um den offenen Kommunikationsaufbau so fortzuführen, dass der Anrufer dies nicht als störend empfindet. Eine erst beim Eintreffen der Signalisierung durchgeführte Abfrage, wie dies beispielsweise in der NEXUS-Plattform der Fall ist, kann möglicherweise die Informationen nicht rechtzeitig liefern. Daher wurde ein Publish/Subscribe-Mechanismus gewählt, bei dem die erforderlichen Informationen asynchron vom ContextServer an den Client gesendet werden. Der Kontext wird im UA und in der CPL-Engine lokal vorgehalten. Der Austausch zwischen den UAs erfolgt im Request/Reply-Mechanismus.

Zwischen den ContextServer-Entitäten wird ebenfalls ein Subskriptions-Mechanismus eingesetzt. So kann ein mobiler Nutzer sich bei unterschiedlichen ContextServer-Entitäten einloggen. Dieser ContextServer kann anschließend den Heim-ContextServer kontaktieren. Die Adressierung des dem Nutzer zugeordneten ContextServer kann unter Verwendung von DHT-Mechanismen, wie dies in [Che04] vorgeschlagen worden ist, erfolgen.

Zur Kontextgewinnung wird für das Aggregationsnetzwerk ein einfacher Push-Mechanismus vorgeschlagen. Die Beziehung zwischen den Sensoren und den Operatorknoten sowie den ContextServers wird durch eine quasi-statische Konfiguration mittels der

Sprache CALL spezifiziert. Dabei können die Knoten *Overlays* bilden, je nachdem wie sie zusammengestellt sind. Der Push-Mechanismus in Kombination mit Filtern und weiteren Aggregationsoperatoren erlaubt den Aufbau von sehr effizienten, schnellen und nachrichtensparenden Kommunikationsbeziehungen mit einer deutlich niedrigeren Komplexität im Vergleich zu einer vollständigen ereignisbasierten Kommunikation.

Der Nachteil dieser Variante ist allerdings die mangelnde Flexibilität in einer dynamischen Umgebung. Dies fällt jedoch durch den avisierten Einsatz in einer lokal begrenzten und administrierten Innenraum-Infrastruktur nicht so stark ins Gewicht. Eine Verbesserung und zukünftige Erweiterung der Datenverteilung ist die Verwendung von *selbstbeschreibungsfähigen* Sensoren und Operatoren, so dass ein sich *selbstorganisierendes* Netz entstehen kann. Die Verteilung der Daten sowie das Management der Sensoren beruhen auf dem ereignisbasierten Paradigma.

5.4 Evaluation

Die gewonnenen Erkenntnisse über Kontexte sowie die addressierten Anforderungen zur Effizienzsteigerung führten zu dem Bedarf nach einer systemorientierten Unterstützung bei der Erstellung von Kontext-bewussten Diensten. Die Unterstützung ist in zwei Komponenten aufgeteilt. Eine Komponente bietet eine Systemunterstützung bei der Erfassung und Verteilung von Kontexten. Die andere Komponente stellt Methoden zur Vereinfachung und Absicherung des Diensterstellungsprozesses bereit. Der Diensterstellungsprozess wird in Kapitel 8 behandelt.

In diesem Kapitel wurden Architekturen und Kommunikationsverteilmechanismen untersucht, die geeignet sind, die Prozessphasen des Kontext-Spiral-Modells zu realisieren. Die in Unterabschnitt 5.1.1 geführte Anwendungsfallanalyse identifizierte vom System zu erfüllende Anforderungen. Die Bewertung der aufgeführten Anforderungen sowie die Komplexität und die Heterogenität der Gesamtumgebung führte zu einem verteilten Systementwurf, der in Abbildung 5.12 aufgezeigt ist. Die jeweiligen Zahlen korrespondieren dabei mit den Kapiteln dieser Arbeit, in welchem die einzelnen Teilbereiche im Detail betrachtet werden.

Für die Gewinnung von Kontexten ⑥ wurde ein geeignetes Modell entwickelt, das den Prozess beschreibt. Für die Erfassung der Kontextmerkmale wird dabei ein Graph aus (virtuellen) Sensoren und Aggregationsoperatoren vorgeschlagen. Die Topologie des Graphens kann mittels einer Konfigurationssprache wie z. B. CALL administriert werden. Die Verteilung der Daten erfolgt in der Push-orientierten Weise von den Quellen zu den Senken.

Zur Verteilung der Kontexte ⑦ wird eine zentrale Integrationskomponente, der ContextServer, vorgeschlagen. Diese Komponente trennt die Kontexterfassung von der Nutzung der Kontexte. Die ContextServer verteilen die Kontexte an die Anwendungen nach dem ereignisorientierten Paradigma mittels Publish-Nachrichten zu den vorher subskribierten Clients. Dadurch können die Prozesse entkoppelt werden und die Verzögerungszeit,

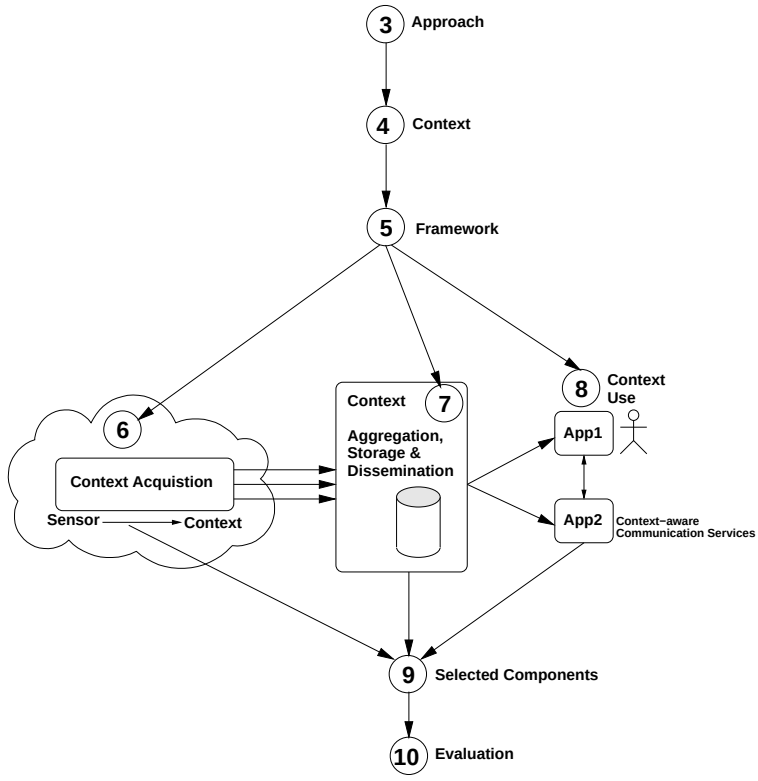


Abbildung 5.12: Outline der Arbeit

die bei einem Request/Response-Verfahren entstehen, kann vermieden werden. Dies ist für den Echtzeitcharakter der Anwendungen im Telefonie-Umfeld essentiell.

Die Erstellung von Kontext-bewussten Kommunikationsdiensten ⑧ mittels unterschiedlicher Methoden zeigt, wie die in den Abschnitten 3.3.1 und 3.3.2 vorgeschlagenen Effizienzsteigerungen bei der Mensch-Mensch-Kommunikation realisiert werden können. Jeder Bereich wurde systematisch untersucht und eine umfassende Lösung für alle Teilbereiche erarbeitet. Zum Nachweis der Realisierbarkeit der vorgeschlagenen Ansätze und Systeme werden repräsentative Komponenten ⑨ entworfen und umgesetzt. Dabei werden die methodischen Überlegungen der vorherigen Kapitel überprüft und einer kritischen Revision unterzogen.

Kapitel 6

Automatische Kontextgewinnung

Alle Menschen streben von Natur
nach Wissen.

Metaphysik I, 980a21
ARISTOTELES

Dieses Kapitel beschreibt den Prozess der Kontextgewinnung, die eine Grundvoraussetzung für die Bereitstellung der Funktionalitäten des in Kapitel 5 vorgestellten Frameworks für Kontext-bewusste Dienste darstellt. Diese benötigen den aktuellen Kontext, um ihre Funktionalität anpassen zu können. Der Kontext kann vom Nutzer manuell eingegeben werden. Eine weitaus größere Nutzbarkeit kann jedoch erreicht werden, wenn der Kontext automatisch vom System erfasst und zur Verfügung gestellt werden kann. Hierfür werden Sensoren für die Erfassung von Umgebungsphänomenen vorgeschlagen. Eine Abstraktionsschicht wird in Form der Virtuellen Sensoren eingeführt, was eine weitere Verarbeitung der Informationen ermöglicht. Mit einem Fuzzy Logik Regelsystem wird ein tragfähiges Konzept zur Synthetisierung von Kontexten aus Sensordaten vorgestellt.

Kurzübersicht

Das Kapitel 6 ist folgendermaßen strukturiert: Die Motivation für die automatische Kontexterfassung ist in Abschnitt 6.1 dargestellt. Die für eine automatische Erfassung notwendige sensorische Erfassung ist in Abschnitt 6.2 mit ihren Elementen, Eigenschaften und Konzepten vorgestellt. Insbesondere wird der Themenkomplex der Sensorfusion, der die konzeptionelle Basis für die Aggregation von Kontextinformationen darstellt, ausführlich erläutert.

Eine Abstraktionsschicht namens *Virtual Sensors*, die zwischen den Sensoren und den Anwendungen liegt, wird in Abschnitt 6.3 eingeführt. In Abschnitt 6.4 werden Syntheseverfahren für Kontextinformationen beschrieben. Detailliert wird ein Fuzzy Logik

Regelsystem als Syntheseverfahren in Abschnitt 6.5 vorgestellt. Abbildung 6.1 gibt die Struktur des Kapitels als Grafik wieder.

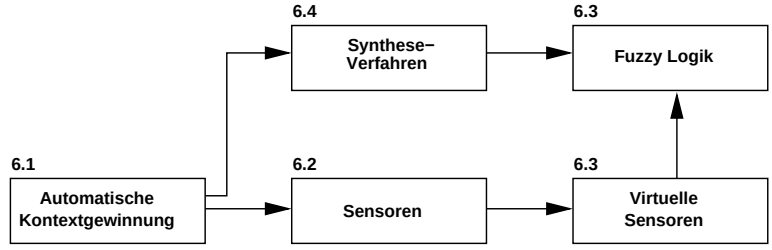


Abbildung 6.1: Strukturübersicht von Kapitel 6

6.1 Motivation

In Kapitel 4 sind die Eigenschaften von Kontexten beschrieben worden. Dabei wurde festgestellt, dass Kontexte aus Kontextmerkmalen bestehen, die zusammengesetzt den Kontext repräsentieren. Kontexte werden im Allgemeinen durch eine Untermenge an charakteristischen Merkmalen ausreichend approximiert.

Die Bereitstellung von Kontexten ist eine Grundvoraussetzung für die Bereitstellung von Kontext-bewussten Diensten. Die einzubeziehenden Kontexte müssen dabei dem Nutzer in einer verwendbaren Form vorliegen. Die Erfassung kann hierbei auf zwei Arten erfolgen:

Explizite Eingaben: Der Nutzer gibt seinen Kontext oder den eines anderen Objektes manuell ein. Dabei kann er beispielsweise aus einer Liste vordefinierter Kontexte wählen oder gibt den Kontext frei ein.

Implizite Eingaben: Der Kontext wird dabei von der Infrastruktur oder vom Gerät erfasst und den Anwendungen zur Verfügung gestellt. Die Erfassung erfolgt hierbei automatisch vom System aus. Dies geschieht dabei oftmals ohne die Wahrnehmung des Nutzers.

Eine oft geforderte Eigenschaft von Kontext-bewussten Anwendungen ist, dass diese aus der aktiven Wahrnehmung des Nutzers verschwinden und ihre Dienste implizit anbieten. Daher kann die *explizite* Eingabe des Kontexts keine favorisierte Lösung zur Bereitstellung von Kontexten sein. Darüber hinaus müssen Kontexte von Entitäten bestimmt werden, in deren räumlicher Nähe der Nutzer sich nicht zwingend befindet.

Die *explizite* Eingabe erfordert ein System zur Erfassung des Kontexts. Da sich dieser in der Regel nicht direkt erfassen lässt, wird er indirekt über die Kontextmerkmale bestimmt. Dies erfordert zum einen Verfahren und Techniken, die Merkmale erfassen können und zum anderen Methoden, diese geeignet zu kombinieren.

Für die automatische maschinelle Erfassung von Kontextmerkmalen haben sich *sensorische Perzeptionstechniken* als leistungsstarke Verfahren bewährt. Dabei stellen die Sensoren gleichsam den Übergang von der physischen in eine elektrische bzw. digitale Welt dar. Wie in Abbildung 4.7 verdeutlicht, werden aus den Rohdaten durch weitere Verfahrensschritte Daten und Informationen – und damit die notwendigen Kontextmerkmale gewonnen. Die sensorischen Erfassungstechniken werden nachfolgend vorgestellt und erklärt. In Abschnitt 6.4 wird die Weiterverarbeitung der Daten zu Merkmalen und Kontexten behandelt.

6.2 Sensoren

Die uns umgebende Umwelt setzt sich aus einer Vielzahl von Phänomenen zusammen. Organische Lebensformen wie Menschen, Tiere und Pflanzen besitzen verschiedene *Sinnesorgane*, diese Phänomene aufzunehmen. Mittels Sinneseindrücken kann die Umwelt wahrgenommen werden. Der Prozess der Sinneserfassung beinhaltet den kompletten Vorgang der Aufnahme der erfassbaren Stimuli, deren Umwandlung in Transportsignale sowie deren Verarbeitung. Zur Abbildung der Vielzahl von Signalen auf sinntragende Einheiten wendet das Gehirn Mustererkennungskonzepte (*pattern matching*) an [Nei76, Gol97].

Der Prozess der neuronalen Signale oder das Konzept von mRNA (*messenger RNA*), etc. sind bis heute nicht im vollen Umfang verstanden. Dennoch finden biologische Konzepte häufig technische Nachahmung. Eine (unvollständige) technische Abbildung der Sinnesorgane stellen *Sensoren* dar. Diese liefern Informationen über einen bestimmten Aspekt einer beobachteten Umgebung.

Definition 6.1 (Sensor)
*Ein **Sensor** ist eine Entität, die eine physische Eigenschaft aufnehmen kann. Diese wird in Form eines Messergebnisses bereitgestellt. Ein Sensor bildet dabei den Wert eines Umgebungsattributs auf eine quantitative Messgröße ab.*

Jeder Sensor S_i kann dabei durch eine zeitabhängige Funktion repräsentiert werden. Diese ist in Gleichung (6.1) dargestellt. Die Funktion kann als Ergebnis einen Skalar, einen Vektor oder einen symbolischen Wert liefern. In [BI98] wird diese Funktion folgendermaßen angeben:

$$S_i : t \mapsto X_i \tag{6.1}$$

Hierbei ist t die (diskrete) Zeit und $X_i \in D_i$ das Ergebnis. Die Domäne D stellt eine endliche oder unendliche Menge an möglichen Werten dar. Eine Vielzahl an Sensoren wird heute durch digitale Systeme realisiert, in denen die Zeit als diskrete Variable angesehen werden kann.

Ein sehr generisches Sensormodell ist in Abbildung 6.2 dargestellt. Ein Sensor besitzt mindestens eine Schnittstelle, die Phänomene aus der Umgebung aufnehmen kann.

Die Art dieser Schnittstelle variiert zwischen den verschiedenen Sensortypen. Nach der Verarbeitung der Eingangsstimuli in Signale werden diese über eine weitere Schnittstelle ausgegeben.

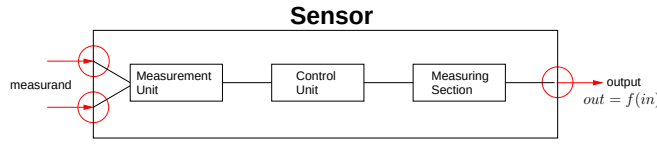


Abbildung 6.2: Generisches Sensor-Modell

6.2.1 Sensortypen

Zwei grundsätzliche Arten von Sensoren werden in [SBG99] unterschieden: *physische* und *logische* Sensoren. Physische Sensoren sind Hardwarekomponenten, die die Phänomene aus der Umgebung messen können. Die erfassten Informationen werden in Form von elektrischen Spannungen repräsentiert. Diese wiederum können ein analoges oder digitales Signal darstellen.

Die andere Typklasse von Sensoren wird als die der *logischen Sensoren* bezeichnet. Dies sind Komponenten, die Informationen erfassen, die nicht direkt physisch messbar sind. Es sind oftmals logische Informationen wie UserID, laufende Prozesse oder Systemzeit. Die Ausgangsmessgröße liegt in der Regel in digitaler Form vor.

6.2.2 Sensorfusion

Die Messung mit Sensoren ist mit einer Reihe von *Limitierungen* behaftet, von denen einige nachfolgend aufgeführt sind.

Sensorausfall: Fällt ein Sensor aufgrund eines Defekts aus (*sensor deprivation*), so verliert das System die Beobachtung des Objektes.

Eingeschränkte räumliche Abdeckung: Ein Einzelsensor deckt in der Regel nur einen eingeschränkten räumlichen Bereich ab. Ein Temperaturfühler kann die Temperatur nur für einen bestimmten Bereich zuverlässig erfassen.

Eingeschränkte zeitliche Abdeckung: Die maximale Messfrequenz der Abtastung der Umwelt des Sensors ist endlich. Zusätzlich kann der Sensor während potenziell notwendiger Setup-Zeiten keine Daten erfassen und weiterleiten.

Ungenauigkeit: Messungen eines einzelnen Sensors sind üblicherweise in ihrer Präzision eingeschränkt, wobei die Präzision von den Charakteristiken des zu messenden Objektes, dem eingesetzten Sensor und der Anwendung abhängt.

Unsicherheit: Die Unsicherheit der Messung hängt im Wesentlichen von dem zu messenden Objekt und weniger von dem eingesetzten Sensor ab. Unsicherheit kann beispielsweise entstehen, wenn ein Sensor nicht in der Lage ist, alle notwendigen Umgebungsattribute zu messen oder wenn die Beobachtung mehrdeutige Ergebnisse liefert [Mur96]. In [FH95] wird festgestellt, dass die eingeschränkte Wahrnehmung des Objektes es einem einzelnen Sensor unmöglich macht, die Unsicherheit der Messung zu verringern.

Die oben genannten Einschränkungen von Messverfahren mit Einzelsensoren sind für viele Anwendungsbereiche nicht akzeptabel. Die Kompensation von Ausfällen ist ein Hauptforschungsfeld bei der Untersuchung von *fehlertoleranten* Komponenten. Ein Standardansatz bei fehlertoleranten Komponenten ist die Verwendung von mindestens drei identischen Einheiten und einem Mehrheitsentscheider (*voter*) [vN56].

Durch den Einsatz von mindestens zwei identischen Einheiten lassen sich Systeme mit (*fail-silent*) Verhalten konzipieren [KKG+90]. Ein solches System liefert entweder das korrekte Ergebnis oder im Fehlerfall kein Ergebnis. Dieses Konzept wurde in Form von Sensorfusion auf Sensorsysteme übertragen. So erlaubt die *überlappende* Messung mit mehreren Sensoren Robustheit gegen den Ausfall eines Sensors.

Weitergehende Verfahren zur Lösung des Problems von Einzelsensoren sind unter dem Begriff *Sensorfusion* zusammengefasst. Die Verwendung von mehreren Sensoren (*multiple sensors*) erlaubt durch Aggregation eine höherwertige (*high-level*) Ausgabe. Durch diese Multi-Sensorfusion genannte Technik können Informationen von einer Güte erreicht werden, wie dies mit einem Einzelsensor nicht möglich gewesen wäre [SE01].

Die Definitionen und Begrifflichkeiten in Fusionssystemen variieren zwischen den verschiedenen Anwendungsszenarien. So ist *Data-Fusion* (*data fusion*) die generischste Bezeichnung der Verarbeitung von Daten verschiedener Quellen [Wal98]. Darüber hinaus wird der Begriff in einigen Modellen für die Fusionierung auf der Rohdatenebene verwendet [Das97].

Definitionen und eine vereinheitlichende Terminologie zu Data-Fusion sind in [LH98, HL01] zu finden. Eine alternative Bezeichnung der Data-Fusion wird in [WL90, Hal92] mit *Multi-Sensor Fusion* angegeben. Der Begriff bezieht sich dabei stärker auf die Kombination von Daten verschiedener Sensoren. Einen allumfassenden Begriff für alle Bereiche der Fusion ist *Informationsfusion* (*information fusion*) [Das01]. Darüber hinaus kann dieser Begriff auch für verwandte Anwendungsgebiete wie Data-Mining oder Datenbankintegration verwendet werden.

Die International Society of Information Fusion definiert *Informationsfusion* folgendermaßen:

Definition 6.2 (Informationsfusion)
*Die **Informationsfusion** umfasst Theorien, Techniken und Werkzeuge, die entwickelt worden sind, die Synergien in den Informationen, die von mehreren Quellen (Sensoren, Datenbanken, menschliche Eingaben) stammen, nutzen. Derart dass Entscheidungsergebnisse oder Aktionen in bestimmter Weise besser (qualitativ oder quantitativ in Bezug*

auf Genauigkeit, Robustheit, etc.) sind, als diese wäre, wenn diese Synergie nicht genutzt worden wäre und die Informationen einzeln bewertet worden wären.

Der Begriff *Sensorfusion* wird dabei als Unterbegriff des Begriffs *Informationsfusion* aufgefasst und wird definiert als [Elm02]:

Definition 6.3 (Sensorfusion)
Sensorfusion *ist die Kombination von Sensordaten oder Daten, die von Sensordaten abgeleitet sind, derart, dass das Ergebnis der Fusion eine bessere Information darstellt, als es durch isolierte Auswertung der Informationen der einzelnen Sensoren möglich wäre.*

6.2.2.1 Eigenschaften von Sensorfusionverfahren

Die Fusionierung von Sensordaten, die von einer Menge homogener, aber auch heterogener Sensoren stammen, liefert eine Reihe von vorteilhaften Eigenschaften [BRG96, Gro98]:

Robustheit und Verlässlichkeit: Systeme, die aus einer Vielzahl von Sensoren bestehen, die dasselbe Objekt beobachten, haben eine inhärente Redundanz, die es dem System erlauben, auch dann noch Messresultate zu liefern, wenn es zu (partiellen) Ausfällen kommt.

Erweiterte räumliche und zeitliche Abdeckung: Durch eine Vielzahl an Sensoren können die Eigenschaften eines Objektes an verschiedenen Stellen beobachtet werden. So können mehrere Temperaturfühler die Temperatur eines Objektes an verschiedenen Bereichen messen und so eine genauere Erfassung ermöglichen. Während der Setup-Zeit eines Sensors kann ein anderer Sensor die Messung aufrechterhalten.

Erhöhtes Vertrauen: Durch Vergleich der Messergebnisse eines Sensors mit der Messung desselben Objektes durch einen anderen Sensor kann das Vertrauen (*confidence*) in die Messung erhöht werden. Durch die Verwendung von nicht-identischen Sensoren (beispielsweise von verschiedenen Herstellern) können systemimmanente Fehler minimiert werden.

Verringerung der Mehrdeutigkeit und der Unsicherheit: Die Verknüpfung von Daten, die von mehreren Sensoren erfasst worden sind, ermöglicht die Reduktion der Mehrdeutigkeit bei der Interpretation der Daten. Dies verringert auch die Unsicherheit in die gewonnenen Ergebnisse.

Robustheit gegen Interferenz: Durch die Kombination von unterschiedlichen Messverfahren (z. B. optische und Ultraschall-) kann eine mehrdimensionale Abtastung der Umgebung erzielt werden. Solche kombinierten Systeme sind in der Regel weniger anfällig gegen Interferenz bei der Messung.

Verbesserte Auflösung: Die Auflösung der Abtastung eines Objektes kann durch die fusionierte Betrachtung der Messdaten von unabhängigen Sensoren erhöht werden. Die so erzielten Ergebnisse sind üblicherweise aussagekräftiger als die Ergebnisse einer Einzelsensormessung.

In [Rao98] wurde durch Vergleich der Ergebnisse eines Sensorfusionssystems mit den Ergebnissen der Einzelsensoren gezeigt, dass der Fusionsprozess der Daten mindestens ebenso gute Ergebnisse liefert wie der jeweils beste Einzelsensor. Ein anderer Vorteil eines vorgeschalteten Sensorfusionsverfahrens liegt in der Reduktion der Komplexität der Anwendung. Ohne Sensorfusion muss die Anwendung mit der großen Anzahl an ungenauen, mehrdeutigen und unvollständigen Daten umgehen, was einen hohen Aufwand darstellt.

6.2.3 Konfigurationen

Die bloße Verteilung von mehreren Sensoren garantiert noch keine besseren Ergebnisse. Eine für die jeweilige Anwendung geeignete Anordnung der Sensoren und der Operationen ist erforderlich. Diese *Konfigurationen* lassen sich verschieden kategorisieren.

Bei dem 3-Ebenen-Kategorisierungsmodell wird zwischen den Ebenen *niedrig*, *mittel* und *hoch* (*low, intermediate, high*) unterschieden. Auf jeder Ebene kann ein Fusionsprozess stattfinden.

Fusion auf niedriger Ebene: Diese wird auch als *Rohdatenfusion* bezeichnet und bezieht sich auf die Kombination von mehreren Informationsquellen. Das Ergebnis dieses Prozesses hat in der Regel einen höheren Aussagewert.

Fusion auf mittlerer Ebene: Diese Ebene wird auch *Feature-Level-Fusion* genannt. Sie bezeichnet die Verknüpfung von verschiedenen Merkmalen eines Objektes zu einer Merkmalsmatrix. Das Ergebnis dieser Stufe wird oftmals zur Segmentierung oder Detektion genutzt.

Fusion auf hoher Ebene: Diese wird auch *Entscheidungsfusion* genannt und beschreibt die Einbeziehung von Entscheidungsmethoden wie Mehrheitswahl, Fuzzy-Logik oder andere statistische Methoden.

Alternativ kann man Sensorfusionssysteme anhand des Abstraktionslevels ihrer Ein- und Ausgabewerte kategorisieren. Das in [Das97] beschriebene Modell stellt eine Verfeinerung des 3-Ebenen-Modells dar. Dem Modell liegt die Beobachtung zu Grunde, dass bei manchen Fusionsprozessen die Eingaben zu einer anderen Abstraktionsebene gehören als die Ausgabe. So erhält ein Mustererkennungsprozess Eingabewerte aus der Merkmalsklasse und produziert Ausgaben auf der Entscheidungsebene. Die sich aus diesem Paradigma ergebenden fünf Kategorien an Ein- und Ausgabeebenen sind in Abbildung 6.3 dargestellt.

Eine Kategorisierung, basierend auf der Anordnung der Sensoren, wurde in [DW88] vorgeschlagen. Das Modell unterscheidet drei verschiedene Typen an Sensoranordnungen:

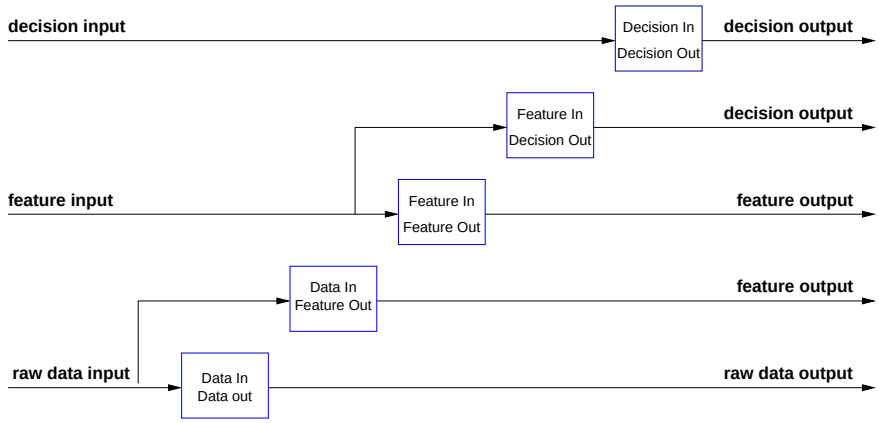


Abbildung 6.3: Ein-/Ausgabe-Modell für Sensorfusionsarchitekturen

Komplementär: Sind die einzelnen Sensoren nicht direkt voneinander abhängig, so nennt man sie *komplementär*. Durch die Kombination der Einzelsensoren kann ein vollständigeres Bild der beobachteten Phänomene erreicht werden.

Die Fusion der komplementären Daten ist problemlos möglich, da diese voneinander unabhängig sind [BI98]. In Abbildung 6.4 wird diese Konfiguration durch die beiden Sensoren S_2 und S_3 verdeutlicht, die jeder einen anderen Teil, des Umfelds von Interesse messen.

Konkurrierend: Liefern verschiedene Sensoren unabhängige Daten von demselben Objekt, so sind diese *konkurrierend* zueinander. Durch diese Anordnung können redundante Sensorsysteme konzipiert werden [LK89].

Die beiden Sensoren S_1 und S_2 observieren die gleiche Eigenschaft desselben Objektes. Durch die erreichte Fehlertoleranz wird die Robustheit des Systems erhöht. Bei auftretenden Fehlern wird die Leistung des Systems stückweise degradiert, es fällt aber nicht auf einmal aus [BZJK01].

Kooperierend: Das Ergebnis der *kooperierenden* Fusion von Einzelsensordaten liefert Informationen, die mit Einzelsensoren nicht erreichbar wären. Jedoch benötigen kooperative Systeme auch das komplexeste Setup [BI98], da der Fusionsprozess anfällig gegenüber Fehlern und Ungenauigkeiten jedes einzelnen einbezogenen Sensors ist.

In Abbildung 6.4 stellen die beiden Sensoren S_4 und S_5 eine kooperative Konfiguration dar. Jeder der Sensoren beobachtet dasselbe Objekt, hier mit C gekennzeichnet. Jedoch kann erst durch die kooperative Fusion eine emergente Ansicht des Objektes erreicht werden, eine Eigenschaft, die bei der in Abschnitt 9.2 beschriebenen WLAN-basierten Lokationsbestimmung genutzt wurde.

Die drei Konfigurationstypen sind nicht wechselseitig exklusiv, sondern werden in Abhängigkeit der Anwendung miteinander kombiniert. Solche Systeme besitzen eine *hy-*

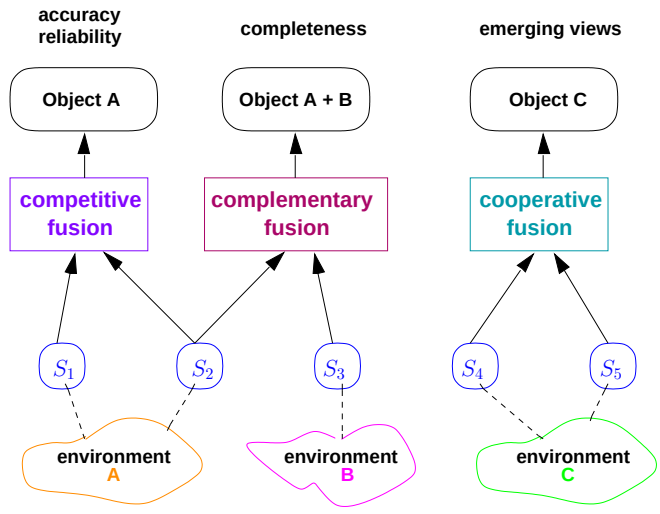


Abbildung 6.4: Komplementäre, konkurrierende und kooperative Sensorfusionskonfigurationen

bride Architektur. So lassen sich die Vorteile der einzelnen Konfigurationen dort einsetzen, wo sie den höchsten Nutzen erzielen. Generell ist das einzusetzende Sensor Fusion Modell stark von der anvisierten Anwendung abhängig, so dass es keine einheitliche, anderen Konfigurationen in allen Belangen überlegene Sensor Fusion Architektur geben wird. Vielmehr werden erfolgreiche Systeme aus einer Kombination von existierenden und erprobten Architekturen bestehen [KZK97].

6.3 Virtuelle Sensoren

Es existiert eine kaum überschaubare Anzahl an Sensortypen und Herstellern von Sensoren. Sensoren, insbesondere physische Sensoren, sind oftmals nur über eine proprietären Schnittstelle und ein herstellereigenes Protokoll ansprechbar. Für den Entwickler von Applikationen und Systemen stellt die Interaktion mit einer Vielzahl an unterschiedlichen Sensoren eine nicht zu vernachlässigende Mehrarbeit dar, da die jeweiligen Spezifika der eingesetzten Sensoren berücksichtigt werden müssen. Bei einem Austausch von Sensoren müssten diese herstellerspezifischen Interaktionen mit den Sensoren entsprechend angepasst werden. Auch bei einer Sensorfusion muss mit den einbezogenen Sensoren individuell kommuniziert werden.

Eine *Abstraktionsebene* bietet hierbei einen gangbaren und erprobten Weg, den Aufwand bei der Erstellung von Applikationen zu reduzieren. So stellt die Abstraktionsebene eine standardisierte Schnittstelle mit wohldefiniertem Funktionsumfang und Datenformat der darüberliegenden Anwendung zur Verfügung. Die herstellerspezifischen Details der Adressierung und Kommunikation mit den Sensoren werden vor der Anwendung verborgen. Eine solche Abstraktionsebene ist auch in den Arbeiten zum Context-Toolkit [DA00] und bei [Sch02b] für Sensoren beschrieben worden.

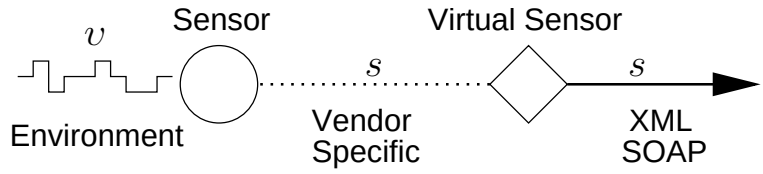


Abbildung 6.5: Virtueller Sensor

Physische Sensoren sind in der Regel einfache elektronische Bauteile, die über keinen oder nur wenig Speicher und über einen leistungsschwachen Prozessor verfügen. Daher kann die Umwandlung der anfallenden Sensordaten in ein einheitliches und aussagekräftiges Format nicht auf dem Sensor selbst erfolgen. Auch kann das vom Hersteller vorgegebene Format und Protokoll der Datenübertragung in der Regel nicht geändert werden. Oftmals liegen die Sensorwerte v auch nur in Form von Spannungsstufen vor und müssen erst interpretiert werden. Andere Sensoren liefern bereits Sensordaten s mit Wert und Einheit. Die Vielgestaltigkeit der Sensoren macht es für den Anwendungsentwickler sehr schwer, die unterschiedlichen Sensoren einzubeziehen, da er die Unterschiede kennen und adressieren muss.

Um diesem entgegenzuwirken, wird die Einführung von *virtuellen Sensoren* als Abstraktionsebene vorgeschlagen. Ein Virtueller Sensor (VS) (*Virtual Sensor*) ist als Software realisiert und agiert als Proxy für einen oder mehrere reale Sensoren. Etwaige geforderte Funktionalitäten können von den realen Sensoren auf Geräte verlagert werden, die über genügend Kapazitäten verfügen, um die geforderten Eigenschaften erfüllen zu können. Eine Vielzahl an Plattform steht dem Entwickler dabei zur Verfügung. Dies können x86-Architektur oder ARM-Prozessor sein, aber auch programmierbare Mikrocontroller (*programmable micro-controller*) wie beispielsweise ein PIC [www13] oder ein FPGA von Xilinx [www33], die ebenfalls ausreichend Leistung bieten können.

Abbildung 6.5 zeigt schematisch einen virtuellen Sensor. Ein virtueller Sensor besitzt auf der einen Seite Schnittstellen zu den herstellereigenen Protokollen und Formaten, die er über eine weitere Schnittstelle in einem standardisierten Format und mit einem festgelegten Protokoll zur Verfügung stellt. Die standardisierte Schnittstelle des virtuellen Sensors ist in dieser Arbeit mittels der Web Service Description Language (WSDL) beschrieben. WSDL bietet hierbei eine standardisierte Form der Beschreibung von Schnittstellen, so dass andere Applikationen diejenigen Sensoren finden und verwenden können, die sie benötigen. Die virtuellen Sensoren verfügen dadurch auch über eine *Selbstbeschreibungsmöglichkeit*. Ein weiterer Vorteil der Kapselung der realen Sensoren durch virtuelle Sensoren wird durch die umfangreiche Attributierung der Sensoren erreicht.

In einem Aggregationsnetzwerk, wie es in Abbildung 7.3 auf Seite 117 zu sehen ist, sind die virtuellen Sensoren die Schnittstelle zwischen der physischen Welt mit realen Sensoren als Datenquelle und der virtuellen Welt mit den Operator-Knoten dar. Die virtuellen Sensoren stellen damit auch die Grenze dar zwischen der Domäne des Aggregationsnetzwerks und der physischen Umgebung, die in die Aggregation einbezogen

werden soll. Innerhalb des Aggregationsnetzwerks wird ein einheitliches Protokoll und Format verwendet. Auch soll der Typ der Sensoren standardisiert sein, damit Anwendungen die Sensoren finden und verwenden können.

Für den Austausch von Daten zwischen dem virtuellen Sensor und den Komponenten des Aggregationsnetzes wurde eine erweiterte Fassung des XML-basierten Presence Information Data Format (PIDF) namens PIDF-CE entwickelt. Die Erweiterungen, die an der ursprünglichen Version vorgenommen wurden, sind in Unterabschnitt 7.5.2 näher erklärt. Ein Beispiel für Sensordaten, die in PIDF-CE formatiert sind, ist in Anhang C gezeigt. Die einzelnen Sensoren und Sensordaten können mit einer Reihe von *Attributen* versehen werden, um sie besser charakterisieren zu können. Weiterhin werden anhand dieser Attribute Sensoren selbstständig vom System ausgewählt. Die wichtigsten Attribute sind in Abbildung 6.6 abgebildet.

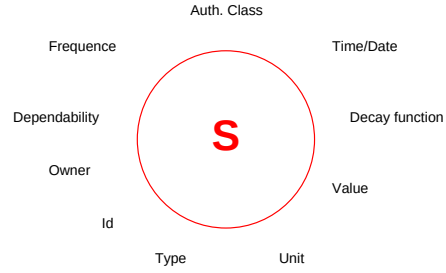


Abbildung 6.6: Sensor Attribute

Die einzelnen Attribute sind nachfolgend erläutert:

Id: Jeder Sensor ist über einen global eindeutigen *Bezeichner* identifiziert. Hierdurch kann jeder Sensor von anderen Sensoren unterschieden werden. Durch eine hierarchische Identifizierung können Gruppen von Sensoren zur Messung oder Wartung erfasst werden.

Type: Eine Vielzahl an unterschiedlichen *Sensortypen* ist verfügbar. Beispielsweise Sensoren, die Temperatur, Geschwindigkeit oder Nutzeraktivität messen. Um bestimmte Typen von Sensoren identifizieren zu können, müssen diese mit Selbstbeschreibungsinformationen versehen werden. Hierdurch kann das System diese Sensoren bei Bedarf selbstständig auswählen, um sie in den Aggregationsprozess mit einzubinden. Hierfür wird eine Beschreibung mittels der Verwendung der Web Service Description Language vorgeschlagen.

Value: Ein Sensor erfasst ein spezifisches Phänomen eines Objektes oder der Umwelt. Dieses wird in einen *Messwert* umgewandelt und kann von anderen Entitäten abgefragt und genutzt werden.

Unit: Jeder Wert hat zusätzlich eine *Einheit*. Existiert keine physikalische Einheit (z. B. nach dem SI-System) erhält der Wert eine symbolische Beschreibung, wie zum Beispiel eine Raumnummer.

Time/Date: Jede Messung wird mit einem *Zeitstempel* markiert. Als Zeit- und Datumsformat wird der iCal-Spezifikation gefolgt. Das Attribut wird verwendet werden, um beispielsweise das Alter der Messung zu bestimmen oder die Abklingfunktion zu berechnen. Zusätzlich kann die Information verwendet werden, um einen zeitlichen Verlauf zu erstellen, der für Kontextvorhersagetechniken verwendet werden kann.

Owner: Jeder Sensor kann einem *Besitzer* oder einer Besitzergruppe zugeordnet werden. Dadurch kann der Sensor einer bestimmten Entität zugeordnet werden.

Dependability: Die Sensoreigenschaften variieren zwischen den unterschiedlichen Sensortypen. Das Attribut *Verlässlichkeit* erlaubt es, jedem Sensor einen Wert zuzuweisen, um seine Verlässlichkeit in die Bewertung der Messwerte einfließen zu lassen. Durch Sensorfusionstechniken kann die Verlässlichkeit eines Einzelsensors erhöht werden und damit die Qualität der Messung verbessert werden.

Frequency: Mit welcher Wiederholfrequenz ein Sensor einen Messwert bereitstellens kann, wird mittels dieses Attributs beschrieben. Manche Phänomene werden weniger oft beobachtet als andere.

Decay Function: Die *Abklingfunktion* beschreibt, wie sich die Gültigkeit des Messwerts mit der Zeit ändert. Mit fortdauernder Zeit werden bestimmte Werte wie z. B. Lokation des Nutzers weniger relevant. Für verschiedene Phänomene können individuelle Abklingfunktionen definiert werden. Gegenwärtig sind dies: *no decay*, *linear decay* oder *logarithmic decay*. Jede der Funktionen wird über einen Satz an Parametern beschrieben.

Authentication Class: Sensordaten sind sehr sensible Informationen, die nur an berechtigte Nutzer verteilt werden sollen. Die Festlegung einer *Authentisierungsklasse* regelt den Grad und die Art der kryptographischen Sicherheit und damit den Zugang zu den Daten. Ein Konzept mit drei Authentisierungsstufen (*family*, *co-worker* und *others*) wurde verwendet.

6.4 Syntheseverfahren für Kontextinformationen

Im vorherigen Abschnitt wurden die Phänomene einer Nutzer- oder Objektumgebung mittels Sensoren erfasst und über die virtuellen Sensoren in einem einheitlichen Format den nächsten Prozessschritten oder Anwendungen zur Verfügung gestellt. Diese Schritte sind in Abbildung 4.9 dargestellt und bestehen im Wesentlichen aus einer Merkmalsextraktion und der Synthetisierung durch Inferenzbildung.

Die Merkmalsextraktion überführt die Menge der Sensorendaten s in die Menge der Kontextmerkmale ζ, wie dies in Gleichung (4.3) mit der Funktion $\varphi(\cdot)$ beschrieben und in Abbildung 4.5 dargestellt ist. Eine Vielzahl an Verfahren für die Merkmalsextraktion steht zur Verfügung. Es können z. B. statistische Methoden, Support Vector

Machines oder FOURIER-Transformationen sein. Diese werden im Folgenden nicht weiter betrachtet, sondern es werden die Merkmale als Eingangsgrößen für die nachfolgende Kontextaggregation durch Inferenzbildung genommen.

6.4.1 Aggregationsverfahren für Kontextinformationen

Die Aggregation, die in dieser Arbeit verfolgt wird, entspricht der *Entscheidungsfusion*, wie diese in Unterabschnitt 6.2.3 beschrieben ist. Eine Reihe von Verfahren wurde für die Verwendung in Kontext-bewussten Umgebungen vorgeschlagen und eingesetzt. Einige wichtige Verfahren werden nachfolgend beschrieben und anschließend wird in Abschnitt 6.5 der eigene Ansatz mit einem Fuzzy Logik Regelsystem detailiert ausgeführt.

Bei einem klassischen Infernzverfahren wird versucht, die Gültigkeit einer Hypothese durch empirische Wahrscheinlichkeiten auszuwerten. Dabei werden Tests für die Hypothese mit Tests gegen eine alternative Gegenhypothese verglichen. Inferenz wird dabei definiert als

Definition 6.4 (Inferenz)
Inferenz *ist die Überführung einer Aussage oder Beurteilung, die als wahr angenommen wird, in eine andere Aussage, deren Wahrheit auf der der vorherigen Aussage beruht.*

6.4.1.1 Bayessche Netze

Eine Vielzahl von Sensorfusionsverfahren benutzt BAYESsche Netze als Inferenzmethode [DW90, CS96]. Das Verfahren ist nach dem englischen Geistlichen Thomas BAYES benannt [Bay63]. Das in Gleichung (6.2) gezeigte Theorem quantifiziert die Wahrscheinlichkeit einer Hypothese H_i unter der Bedingung, dass ein bestimmtes Ereignis E eingetreten ist.

$$\mathbb{P}(H_i|E) = \frac{\mathbb{P}(E|H_i)\mathbb{P}(H_i)}{\sum_i \mathbb{P}(E|H_i)\mathbb{P}(H_i)} \tag{6.2}$$

Hierbei bezeichnet $\mathbb{P}(H_i)$ die *a priori* Wahrscheinlichkeit, dass die Hypothese für ein Kontextmerkmal eingetreten ist und $\mathbb{P}(E|H_i)$ die Wahrscheinlichkeit, dass das Ereignis E festgestellt werden kann, wenn H_i eingetreten ist.

Das BAYESsche Inferenzverfahren kann zur Klassifizierung verwendet werden, wenn eine Vielzahl an Hypothesen herangezogen werden. Ein Vorteil der Methode gegenüber klassischen Inferenzverfahren ist, dass die Hypothese inkrementell mit jeder neuen Beobachtung bewertet werden kann, und dass a priori Wissen über die H_i einbezogen werden kann.

Die Notwendigkeit zur a priori Kenntnis der Wahrscheinlichkeiten $\mathbb{P}(E|H_i)$ und $\mathbb{P}(E)$ stellt einen Nachteil des Verfahrens dar, da diese nicht immer vorliegen [Kle99, BI98, Bla98]. Bei der Einbeziehung von vielen Hypothesen und abhängigen Ereignissen wächst die Komplexität des Verfahrens stark an. Ein weiterer Nachteil ist, dass nur exklusiv

unabhängige (*mutual exclusivity*) Hypothesen bewertet und dass keine generellen Unsicherheiten berücksichtigt werden können.

6.4.1.2 Dempster-Shafer Theorie

Die DEMPSTER-SHAFER-Theorie ist ein Entscheidungskalkül, welches mit unsicheren und unvollständigen Informationen umgehen kann [Pro92, Hal92, Kli99, Roc97]. Sie stellt Methoden bereit, die es erlauben mit sich widersprechenden Daten zu arbeiten, was mit dem BAYESschen-Theorem nicht möglich ist [Bog87].

Eine mathematische Generalisierung der BAYESschen Theorie, um dieses Problem zu lösen, wurde von DEMPSTER mittels *subjektiver Wahrscheinlichkeiten* entwickelt [Dem67]. Dazu wurden Operationen definiert, die statt mit Wahrscheinlichkeiten mit *belief* und *mass* Funktionen rechnen. Diese Theorie wurde von SHAFER zu einer mathematischen Theorie der Evidenz (*evidence*) erweitert [Sha76].

Das prinzipielle Verfahren kann wie folgt verstanden werden. Die Grundmenge aller sich gegenseitig ausschließenden Hypothesen H wird (Θ) genannt (*frame of discernment*). Die Potenzmenge 2^Θ ist eine hierarchisch angeordnete Aussageklasse, die auch (Θ) selbst enthält. Die Funktion $m\colon 2^\Theta \rightarrow [0,1]$ heißt Massefunktion (*mass function*), wenn gilt:

$$m(\emptyset) = 0 \tag{6.3}$$

$$\sum_{H \in 2^\Theta} m(H) = 1 \tag{6.4}$$

Die Funktion gibt das Maß für die Unterstützung der Elemente des frame of discernments für H an. Die Differenz zu 1 wird Θ zugewiesen und repräsentiert das *Unwissen* (*uncommited belief*). Jede Wahrscheinlichkeit gegen eine Aussage wird als Wahrscheinlichkeit des Komplements der Aussage verstanden. Zur normalisierten Zusammenfassung von mehreren Datenquellen können mittels der DEMPSTER-Kombinationsregel (*rule of combination*) die Massefunktionen zu einer neuen Massefunktion verrechnet werden. E ist hierbei die Evidenz für die Unterstützung des Vertrauens in die Hypothese.

$$m'(H) = \sum_{E_{1_i} \cap E_{2_j} = H} m_1(E_{1_i}) \cdot m_2(E_{2_j}) \tag{6.5}$$

Vertrauen in eine Hypothese wird beim DEMPSTER-SHAFER-Verfahren als Konfidenzintervall angegeben, welches durch die Überzeugung (*belief*) und die Plausibilität (*plausibility*) beschrieben ist.

$$[\mathrm{Bel}(H), \mathrm{Pl}(H)] \tag{6.6}$$

Die Funktion $\mathrm{Bel}\colon 2^\Theta \rightarrow [0,1]$ ist definiert als:

$$\mathrm{Bel}(H) = \sum_{E \subseteq H} m(E) \tag{6.7}$$

und beschreibt die Unterstützung für H. Die Funktion Pl: $\rightarrow [0,1]$ heisst *plausibility function*, wenn gilt:

$$\mathrm{Pl}(H) = \sum_{E \cap H \neq \emptyset} m(E) \tag{6.8}$$

Sie gibt die maximal mögliche Unterstützung aller E, deren Schnittmenge mit H nicht leer sind, für H an.

Aus der Intervallbreite $\mathrm{Pl}(H) - \mathrm{Bel}(H)$ kann die Glaubwürdigkeit einer Aussage bestimmt werden. Je breiter das Intervall, desto weniger vertrauenswürdig ist die Aussage. Insbesondere die Eigenschaft der Assoziativität und Kommutativität [SF02] der Evidenzkombinationsregeln macht die DEMPSTER-SHAFER-Theorie neben der BAYESschen-Methode zu einem weitverbreiteten Verfahren für Sensorfusionen. Allerdings sind die Ergebnisse beim Einsatz zur Kontextaggregation wie in [WSSY02] gezeigt, nicht besser als die klassischen BAYESschen Verfahren, was die hohe Komplexität und den größeren Rechenaufwand nicht rechtfertigt.

6.4.1.3 Kohonen Self-Organizing Maps

In [Lae99, LC00] werden Self-Organizing Maps (SOM) bzw. *Kohonen Self-Organizing Map* (KSOM) [Koh89] zur Aggregation von Kontextmerkmalen zu Kontexten vorgeschlagen. Kohonen Maps basieren auf [vdM73] und sind unüberwachte (*unsupervised*) Neuronale Netze, die Strukturen in sehr großen und hochdimensionalen Datenräumen finden und mittels Training erlernen können. Die SOMs enthalten zusätzlich *Nachbarschaftsbeziehungen* zwischen den Neuronen, wie dies in Abbildung 6.7 im Output-Layer zu sehen ist. Hierdurch ist es möglich, diese Netze als topologieerhaltende Abbildungen zu verwenden.

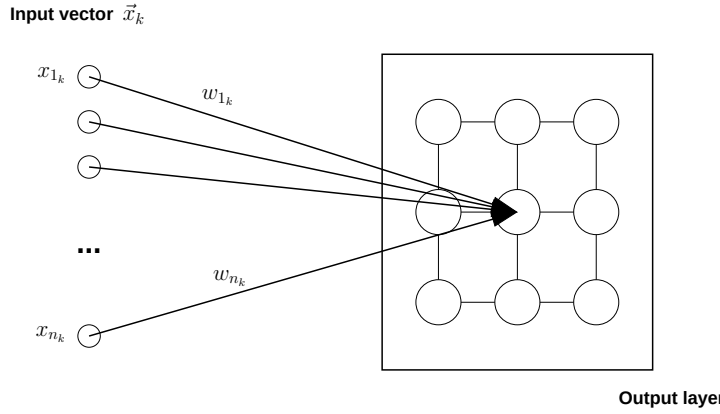


Abbildung 6.7: Kohonen Self-organizing Maps

Vorteile der Kohonen Self-Organizing Maps (KSOM) sind, dass sie auch aus fehlerbehafteten (*noisy*) Eingangsdaten gute Ergebnisse erzielen können. Der Aktivierungsvorgang der Neuronen in der Ergebnismatrix kann in KSOMs graphisch sichtbar gemacht

werden, wodurch dem Nutzer ein Einblick in den Abbildungsvorgang auf die Kontexte gegeben wird.

Der Nachteil einer Kohonen Self-Organizing Map (KSOM) ist ihr unüberwachter Lernmodus. Nach einer schnellen Adaptierung bleiben die Verknüpfungen weitgehend festgelegt. Dies ist eine starke Einschränkung, wenn das System weiterhin adaptiv bleiben muss. Dieses Problem ist auch als *stability-plasticity dilemma* bekannt und Erweiterungen der KSOM versuchen, dies durch Einführung von Überwachung zu lösen. Ein weiterer Nachteil ist die geringe Geschwindigkeit, mit der die Daten zusammengefasst (*clustering*) und abgebildet werden können.

6.5 Fuzzy Logik zur Kontextaggregation

Ein auf der Basis der Fuzzy Logik aufgebautes Regelsystem wurde als eine neuartige Möglichkeit zur Kombination von Kontextinformationen entworfen, implementiert und evaluiert. Die Komponente ist in den *Synthetisierungsblock* der Kontextspirale einzuordnen. Nachfolgend wird eine kurze Einführung in das Konzept der Fuzzy Logik, ihrer mathematischen Prinzipien und den Aufbau eines Fuzzy Logik Regelsystems gegeben. Abschließend werden die durchgeführten Arbeiten auf Basis einer frei verfügbaren Bibliothek mit Fuzzy Logik Operationen zur Erstellung einer eigenen Anwendung dargestellt.

6.5.1 Einführung Fuzzy Logik

Der griechische Philosoph Aristoteles postulierte den berühmten *Ausschluss des Dritten* „Tertium non datur“, was die Entwicklung der klassischen binären (zweiwertigen) Logik und ihrer Systeme einleitete. Sein Lehrer Platon vertrat den Standpunkt, dass es eine dritte Region zwischen wahr und falsch geben müsste. Erst viele Jahrhunderte später wurde diese Ansicht von den Philosophen Georg Hegel und Bertrand Russel [Rus23] in ihren Arbeiten wieder aufgegriffen. Mathematisch wurden mehrwertige Logiken motiviert durch die Quantentheorie von Łukasiewicz. Später wurde durch die Beobachtung, dass bei einer Vielzahl von mathematischen Modellen in der Biologie und Technik Unschärfen und Vagheiten (*vagueness*) nummerisch dargestellt werden müssen, eine mehrwertige Logik von Black entwickelt.

So führte Łukasiewicz eine dritte Logikstufe namens „possible“ mit dem Wert $\frac{1}{2}$ ein [Łuk20]. Später entwickelte er auch vier- und fünfwertige Logiken. Seine wichtigen Überlegungen bezog er aus der Wahrscheinlichkeitstheorie, aus der er eine unendlichwertige Logik, die alle Zahlen aus dem Intervall $[0, 1]$ als Wahrheitswerte zulässt, formulierte [Łuk13]. Black entwarf ein Verfahren, mit welchem die Unschärfe von Symbolen nummerisch dargestellt wird. Hierzu wird die Vagheit mittels ihres Komplements ausgedrückt. Elemente, die weder zum Symbol selbst, noch zum Komplement gehören, nennt er *Fransen* (*frings*) [Bla37]. Eine vollständige und systematische unscharfe Logik, die die Ideen von Łukasiewicz und Black verbindet, wurde 1965 im grundlegenden Aufsatz

"Fuzzy Sets" von ZADEH veröffentlicht [Zad65]. Diese ermöglicht eine Generalisierung der bivalenten Logik.

Das Konzept der Fuzzy Set Theory bzw. *Fuzzy Logic* stellt dabei eine Verallgemeinerung der klassischen Mengenlehre bzw. der BOOLschen Logik dar. ZADEH hat bei der Entwicklung der Fuzzy Logik die Verwendung von unscharfen (*fuzzy*) Begriffen, wie diese beispielsweise in der Sprache verwendet werden – im Gegensatz zu scharf abgegrenzten Zahlenwerten bei Berechnungen – als Vorbild angesehen [Zad94]. In einer Reihe von Disziplinen hat sich die Fuzzy Logik als formale Beschreibung der menschlichen Denkweise als äußerst nützlich erwiesen. Diese Verfahren wurden auf Regelkreise angewendet. Diese *Fuzzy Control* geht auf MAMDANI [MA75] zurück.

Die Verwendung eines Fuzzy Logik Systems als *Inferenzmethode* zur Aggregation von Kontextinformationen ist neuartig. Sie bietet eine Reihe von Vorteilen gegenüber anderen Verfahren wie BAYESschen Netzen [Bay63], DEMPSTER-SHAFER Reasoning [Sha76, Dem67] oder KALMAN-Filtern [Kal60, KB61]. Fuzzy Logik ist gut geeignet, um zwischen *Präzision* (*precision*) und *Signifikanz* (*significance*) zu unterscheiden. Sie erlaubt es, Aussagen aus vagen, mehrdeutigen und ungenauen Informationen zu treffen. Ein weiterer Vorteil liegt in der intuitiven Benutzung von Fuzzy Logik Regeln, die in einer natürlichen sprachlichen Art und Weise aufgestellt und gedeutet werden können. Sie erlauben eine nicht-lineare Abbildung mit theoretisch uneingeschränkter Komplexität einer Eingangsmenge auf eine Ausgangsmenge.

Ein Fuzzy Logik System, wie es in Abbildung 6.8 dargestellt ist, wurde in dieser Arbeit entwickelt. Der prinzipielle Entscheidungsfindungsprozess eines Fuzzy Logik Regelsystems umfasst drei Phasen:

1. Die eingehenden reellwertigen Daten $(x_1 \dots x_n)$ werden *fuzzifiziert*, d. h. es werden die Zugehörigkeitswerte $\mu_{LW_{i,j}}(x_i)$ zu den linguistischen Werten gebildet.

2. In der *Inferenzphase* findet die Regelbasisauswertung durch logische Verknüpfungsoperationen wie Aggregation, Implikation und Akkumulation statt.

3. Abschließend wird das Fuzzy-Ergebnis μ_{res} mittels *Defuzzifizierung* unter Anwendung von logischen Junktoren wieder auf einen reellwertigen Zahlenwert gebracht.

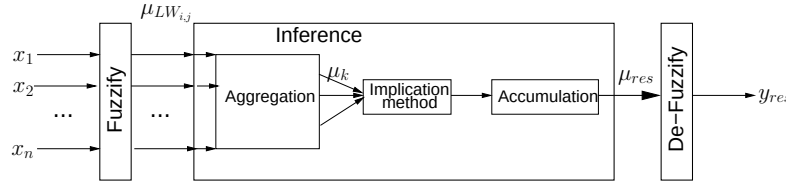


Abbildung 6.8: Fuzzy Logik System

Die Verwendung eines Fuzzy Logik Regelwerks erlaubt es, die zu aggregierenden erfassten Merkmale der Natur und ihre Verknüpfung zu beschreiben. Die zu verknüpfenden Ausdrücke sind in Form von *WENN ... DANN*-Regeln beschrieben. Als linguistische

Variable wurden höherwertige Kontextinformationen wie Raumtemperatur, Mausbewegung aber auch Lokationsinformationen verwendet. Nachfolgend wird eine Einführung in die notwendigen Konzepte der Fuzzy Logik gegeben. Die einzelnen Phasen werden anhand des Beispiels der Raumtemperatur verdeutlicht.

6.5.2 Fuzzy Logik Theorie

Jedes Element x einer Grundmenge X gehört in der klassischen Mengentheorie entweder zur Teilmenge A oder aber zur komplementäre Menge $\bar{A}$. Die Zuordnung wird durch eine Indikator- oder charakteristische Funktion χ beschrieben.

Definition 6.5 (Charakteristische Funktion)
Es sei X eine Grundmenge und A eine Teilmenge von X. Dann heißt die Funktion

$$\chi_A \colon X \to \{0,1\} \tag{6.9}$$

$$\chi_A(x) = \begin{cases} 1 \colon x \in A \\ 0 \colon x \notin A \end{cases} \tag{6.10}$$

***Indikatorfunktion- oder charakteristische Funktion** der Menge A. Die Menge A wird mit Hilfe der charakteristischen Funktion vollständig beschrieben.*

Eine Zuordnung von Werten auf eine scharfe (*crisp*) Menge kann in manchen Anwendungsfällen problematisch sein. So hat jeder Mensch eine (eigene) Vorstellung, welche Temperatur er als „angenehm“ empfindet. Eine exakte Angabe eines Temperaturbereichs beispielsweise von 17 °C bis 28 °C führt zu der Frage, wieso die Temperatur 28.5 °C nicht mehr angenehm sein soll. Ein *gradueller* Übergang von einer unangenehmen Temperatur zu einer angenehmen würde einer Modellierung der menschlichen Wahrnehmung wesentlich näher kommen.

Fuzzy Mengen (*Fuzzy sets*) ermöglichen eine graduelle Zugehörigkeit von Werten zu einer Menge und lassen sich durch eine Verallgemeinerung der charakteristischen Funktion definieren. Hierfür wird das Bild der zweielementigen Menge $\{0,1\}$ auf das Einheitsintervall $I := [0,1]$ ausgeweitet. Im Unterschied zu klassischen Mengen wird den einzelnen Elementen jeweils ein Grad zugewiesen, zu dem sie einer Teilmenge angehören. Dieser Grad wird durch eine Funktion bezeichnet, die *Zugehörigkeitsfunktion* (*membership function*) heißt und mit μ gekennzeichnet ist. Sie ist eine charakteristische Funktion und beschreibt die Fuzzy-Menge vollständig.

Definition 6.6 (Fuzzy Menge)
*Eine unscharfe Menge oder **Fuzzy Menge** $\tilde{A}$ einer Grundmenge X ist eine Menge von geordneten Paaren*

$$\left\{(x, \mu_{\tilde{A}}(x)) \quad | \, x \in X\right\}$$

*Hierbei ist μ eine charakteristische Funktion $\mu_{\tilde{A}}(x) \colon x \to [0,1]$, welche den Grad angibt, mit dem ein Element $x \in X$ in der Menge $\tilde{A}$ enthalten ist. μ wird als **Zugehörigkeitsfunktion** (membership function) bezeichnet.*

Tabelle 6.1: Fuzzy-Rechenoperationen und ihre Kombinationen

Aggregation	Implikation	Akkumulation	Defuzzifizierung
Minimum	Minimum	Maximum	COA
algebraisches Produkt	Produkt	algebraische Summe	MOM
Einstein Produkt	Łukasiewicz	Einstein Summe	MAX
begrenzte Differenz	Kleene-Dienes	begrenzte Summe	

Der Zugehörigkeitsgrad ist dabei um so größer, je mehr der Wert der modellierten Vorstellung von Zugehörigkeit entsprechen soll. Ist eine Temperatur von 0 °C auf keinen Fall als angenehm einzustufen, so kann dies durch $\mu(0\ °\mathrm{C}) = 0$ angeben. Eine Temperatur von 14 °C kann als „weniger“ angenehm als 25 °C beschrieben werden, ausgedrückt durch $\mu(14\ °\mathrm{C}) < \mu(25\ °\mathrm{C})$. Die wahrgenommene angenehme Raumtemperatur kann in unterschiedlichen Erdteilen voneinander abweichen. Die Erstellung einer Zugehörigkeitsfunktion μ ist subjektiv und kontextabhängig, daher existiert keine eindeutige Vorschrift, wie diese aufgestellt werden kann. Die Freiheiten bei der Wahl der Zugehörigkeitsfunktionen erlauben auf der einen Seite eine große Flexibilität in verschiedenen spezifischen Anwendungen, auf der anderen Seite ist die Festlegung aber auch eine komplexe Aufgabe, die oftmals Expertenwissen voraussetzt. Wichtige Funktionsformen sind Rampen, Trapeze, sigmoide S-förmige Funktionen oder Singletons. Singletons sind definiert als:

$$\mu(x) = \begin{cases} 1 \text{ für } x = m \\ 0 \text{ sonst} \end{cases} \tag{6.11}$$

Zur Verknüpfung zweier Fuzzy Mengen sind Mengenoperationen wie Vereinigung (*union*), Durchschnitt (*intersection*) und Komplement (*complement*) für die Zugehörigkeitsfunktion auf dem Intervall $[0, 1]$ definiert worden.

$$A(x) \cup B(x) := \mu_{A \cup B}(x) := \max\{\mu_A(x), \mu_B(x)\} \tag{6.12}$$

$$A(x) \cap B(x) := \mu_{A \cap B}(x) := \min\{\mu_A(x), \mu_B(x)\} \tag{6.13}$$

$$\bar{A}(x) := \mu_{\bar{A}(x)} := 1 - \mu_A(x) \tag{6.14}$$

Die Operatoren min und max bilden jeweils das punktweise Minimum bzw. Maximum der Zugehörigkeitsfunktionen. Eine graphische Repräsentation der Operatoren and, or, und not ist in Abbildung 6.9 dargestellt. Eine Reihe weiterer Und-Funktionen und Oder-Operationen, wie das algebraische Produkt und die algebraische Summe, das Einstein-Produkt und die Einstein-Summe sind ebenfalls definiert. Tabelle 6.1 listet die häufigsten Fuzzy-Rechenoperationen und ihre Kombination auf.

Die Funktionen $\cup$ und $\cap$ sind Funktionen der Vorschrift $[0, 1] \times [0, 1] \rightarrow [0, 1]$ und t-norm bzw. t-conorm. Die Operatoren besitzen eine Reihe von Eigenschaften, die aus der klassischen Mengenlehre bekannt sind. So gilt Assoziativität, Kommutativität, Monotonie und die Verknüpfungen mit 1 und 0. Darüber hinaus gelten die Stetigkeit von μ und

die DE MORGANschen Gesetze. Allerdings gelten nicht das Gesetz des Widerspruchs und das Gesetz des ausgeschlossenen Dritten.

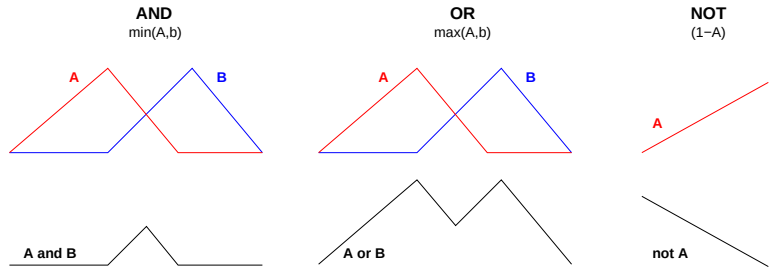


Abbildung 6.9: Operatoren in einer Mehrwertigenlogik

6.5.2.1 Fuzzy Regelbasis

Das Aufstellen einer einzigen Fuzzy Menge ist in der Regel nicht ausreichend, um den gesamten Bereich eines Phänomens zu beschreiben. Dazu werden in der Fuzzy Logik *linguistische Variablen* $\mathcal{L}$ und *linguistische Terme* eingeführt. Eine linguistische Variable umfasst die unterschiedlichen Aspekte der Regel als eine Gruppe von Fuzzy Mengen mit einander überlappenden linguistischen Termen. Eine Variable ist eine sprachliche Bezeichnung für eine technische Eingangsgröße wie z. B. *Raumtemperatur*. Die Werte sind Merkmalsattribute der Variable, wie z. B. *angenehm* und können durch Zugehörigkeitsfunktionen ausgedrückt werden. Dadurch kann eine Empfindung wie "die Raumtemperatur ist angenehm" ausgedrückt werden:

$$\mathcal{L} = \left\{ \tilde{A}_1 \ldots, \tilde{A}_p \right\} = \left\{ (x, \mu_{\tilde{A}_1}(x)), \ldots (x, \mu_{\tilde{A}_p}(x)) \right\} \tag{6.15}$$

mit der üblicherweise gefolgerten Eigenschaft

$$\sum_{j=1}^{p} \mu_{i,j}(x_i) = 1 \tag{6.16}$$

Eine Abbildung der Fuzzy Mengen für die linguistische Variable „Raumtempertatur" ist in Abbildung 6.10 zu sehen.

Die zentrale Komponente des Fuzzy Logik Systems stellt die Inferenzeinheit dar. Sie enthält die Regelbasis, die aus einer endlichen Anzahl an *linguistische Kontrollregeln* R besteht. Eine Regel R_j hat die Form

$$R_j : \text{ IF } x_1 \in \tilde{A}_{1,j} \text{ AND ... AND } x_n \in \tilde{A}_{n,j} \text{ THEN } z_j \in \tilde{B}_j \tag{6.17}$$

und ist eine Fuzzy-Implikation der Form

$$\tilde{A}_{1,j} \times, \ldots \times \tilde{A}_{n,j} \rightarrow \tilde{B}_j \tag{6.18}$$

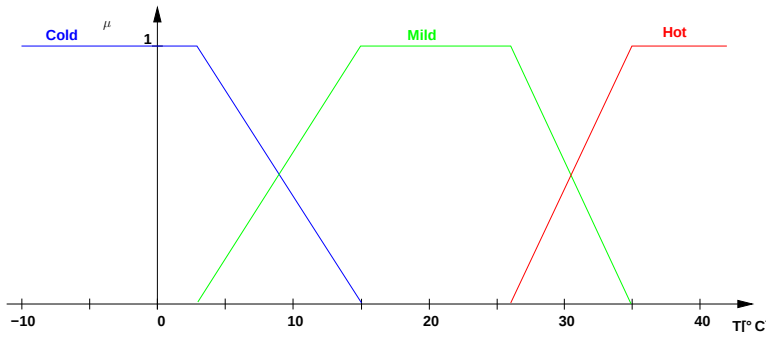


Abbildung 6.10: Graphische Darstellung der linguistischen Variable „Raumtemperatur“

Die Auswertung der Bedingung $x_i \in \tilde{A}_{1,j}$ beschreibt den *Wahrheitswert* oder *Akzeptanzwert* mit dem x_i in der Fuzzy Menge liegt, was $\mu_{\tilde{A}_{i,j}}(x_i)$ entspricht.

Bekannte und häufig eingesetzte Regelsysteme sind von MAMDANI [MA75] und SUGENO [Sug85, ST85] entworfen worden. Jede Regel besteht dabei aus einer *Prämisse* und einer *Konklusion*. Bei SUGENO besteht die Konklusion nicht mehr aus unscharfen Werten sondern aus reellwertigen Zahlen, was zu einem Singleton-Regler führt.

Die Bedingungen der Prämisse in Form von linguistischen Termen sind durch t-Normen (üblicherweise wird für die AND Operationen der min-Operator verwendet) miteinander verknüpft. Eine Prämisse könnte

> “IF *Raumtemperatur* ∈ ’angenehm’ AND *Mausbewegung* ∈ ’normal’
> AND *Lokation* ∈ ’Büroräume’ ”

lauten. Die Ausführung der Verknüpfungen wird als *Aggregation* bezeichnet.

$$\mu_{\text{agg}}(x_1, \ldots, x_n) = \min\{\mu_{i,j}(x_1), \ldots \mu_{n,p}\} \tag{6.19}$$

Im anschließenden *Implikationsschritt* wird $\mu_{\text{agg}}(\vec{x})$ mit der Zugehörigkeitsfunktion $\mu_{\tilde{B}_j}(y)$ ebenfalls mittels des min-Operators zum Implikationsergebnis μ_k verknüpft.

Eine Regelbasis besteht üblicherweise aus einer Vielzahl solcher oben dargestellter Regeln. Deren Implikationsergebnisse werden in der *Akkumulationsphase* mit t-Conorm disjunktiv verknüpft. Als OR Operation wird in der Regel der max-Operator verwendet. Regelbasen für mehrere Eingangsgrößen werden üblicherweise in aus der BOOLschen Logik bekannten KARNAUGH-Diagrammen dargestellt. Sollte die Anzahl der notwendigen Regeln zu groß werden, so lassen sich Fuzzy Regelsysteme in einzelne Teilsysteme zerteilen, die jeweils die Schritte der Fuzzifizierung, Inferenz und Defuzzifizierung enthalten. Das Ergebnis, die Zugehörigkeitsfunktion μ_{res}, aus der Akkumulation stellt sich wie folgt dar:

$$\mu_{\text{res}}(\vec{x}, y) = \tilde{C} = \max\{\mu_1(\vec{x}, y) \ldots \mu_m(\vec{x}, y)\} \tag{6.20}$$

Aus diesem unscharfen Ergebnisgebirge der Entscheidungslogik muss im letzten Schritt, der *Defuzzifizierung*, ein scharfer (*crisper*) Ereigniswert y_{res} bestimmt werden. Es gibt

keine kanonische Abbildungsvorschrift wie aus der Fuzzy Menge ein Wert bestimmt werden kann. Daher existiert eine Reihe von unterschiedlichen Verfahren. Bekannte Verfahren sind hierbei die Maximumsmethode (MAX), Mittelwert der Maxima (*Mean of Maxima – MOM*) oder der Flächenschwerpunkt (*Center of Area – COA*). Allgemein gilt

$$y_{\text{res}} = f(x_1, \ldots, x_n) \tag{6.21}$$

In Abbildung 6.11 ist die graphische Repräsentation des Flächenschwerpunkts zu y_{res} dargestellt.

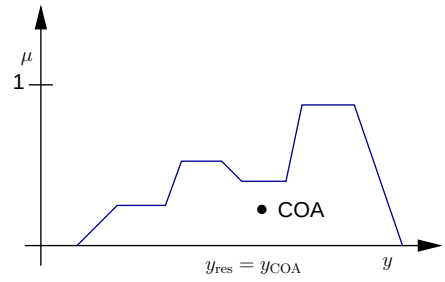


Abbildung 6.11: Flächenschwerpunkt als Ergebniswert

6.6 Bewertung

Die automatische Erfassung von Kontextmerkmalen und die Bestimmung von Kontexten aus diesen ist eine äußerst wichtige Teilfunktion, die ein System zur Unterstützung von Kontext-bewussten Diensten und Anwendungen bieten muss. Die Erfassung ohne explizite Interaktion des Nutzers ist insbesondere vor dem Hintergrund der pervasiven und impliziten Eigenschaften, die Kontext-bewusste Applikationen haben sollen, entscheidend. Für die Perzeption der Umgebungsphänome werden Sensoren verwendet, die über das Konzept der virtuellen Sensoren hinsichtlich ihres Typs und des eingesetzten Datenformats und Protokolls gekapselt werden. Sie stellen die Schnittstelle zwischen der physischen und der virtuellen Welt dar. Für den Anwendungsentwickler stellt dies eine Treibern vergleichbare Abstraktionsschicht dar, wodurch die Entwicklung von Diensten und Anwendungen vereinfacht werden kann.

Eine weitere Abstraktion der Daten wird im Prozess der Kontextsynthetisierung erreicht. Dabei werden die Kontextmerkmale auf Kontexte abgebildet. Bekannte Verfahren wie Bayessche Inferenz, Dempster-Shafer-Theorie der Evidenz und Selbstorganisierende Kohonen Maps wurden vorgestellt und im Zusammenspiel mit der Aggregation von Kontexten und dem Kontext-bewussten Diensten beleuchtet und bewertet. Mit einem Fuzzy Logik Regelsystem wurde ein eigener Ansatz zur Kontextgewinnung vorgestellt. Dieser zeichnet sich durch seine Nähe zu einer natürlich-sprachlichen Beschreibung der Zusammensetzung von Kontexten aus. Ein fundiertes mathematisches Modell und eine breite Softwareunterstützung sind weitere Pluspunkte dieses Ansatzes. In Abschnitt 9.3 wird eine prototypische Umsetzung eines Fuzzy Logik Regelsystems

vorgestellt und zusätzlich eine Evaluation der Leistungsfähigkeit des Ansatzes durchgeführt.

Kapitel 7

Kontextinfrastruktur

> Der Fortschritt lebt vom Austausch des Wissens.
>
> Albert Einstein

In Kapitel 5 wurde die Architektur für eine Unterstützung von Kontext-bewussten Diensten und Anwendungen entworfen. Die vorgeschlagene dreigeteilte Architektur umfasst einen ContextServer als Integrationskomponente zwischen dem Netz der Kontextproduzenten und den Nutzern der Kontexte. Dieses Kapitel beschreibt die entworfene Kommunikationsinfrastruktur sowie die notwendigen Mechanismen für die Verteilung der Daten zwischen den involvierten Komponenten.

Die von Sensoren produzierten Sensordaten sowie die im Syntheseprozess erzeugten Kontextmerkmale werden im Kontextaggregationsnetzwerk verteilt. Zur Erzeugung und zum Management dieses Aggregationsnetzwerks wurde die eigene Sprache CAL entwickelt. Kontexte werden im ContextServer gespeichert und Anwendungen können über eine Service-orientierte Middleware auf der Basis von Web Services auf diese zugreifen. Weiterhin wird zur Datenverteilung ein ereignisbasierter Publish/Subscribe-Mechanismus vorgeschlagen.

Für die Verteilung der Kontexte zwischen SIP-Entitäten wird eine Erweiterung von SIP vorgeschlagen, die eine In-band-Signalisierung ermöglicht. Diese kann sowohl zwischen SIP-Endgeräten, als auch mit einer 3rd-Party Anrufsteuerungskomponente (z. B. CPL-Engine) genutzt werden. Mit PIDF-CE wurde ein Format für eine Repräsentation des Kontexts als XML-Objekt definiert. Die Notation enthält einen reichen Satz an Attributen für jedes Kontextobjekt sowie für Kontextmerkmale und Sensordaten.

Kurzübersicht

Die Struktur des Kapitel 7 ist folgendermaßen: Eine Betrachtung der möglichen Architekturen und Verteilmechanismen von Kontextinformationen ist in Abschnitt 7.1 gegeben. Die betrachteten Konzepte umfassen die in Abbildung 5.12 dargestellten drei Teile des Systems aus Kontextproduzentennetzwerk, der Integrationskomponente ContextServer und der Verteilung von Kontexten an Kontextnutzer.

Eine Beschreibung des Aggregationsnetzes sowie der Sprache CALL, die für die Initiierung und das Management des Netzes entworfen wurde, befindet sich in Abschnitt 7.4. Der ContextServer als eine zentrale Instanz zur Speicherung von Kontexten und zum Management von Subskriptionen von Kontexten seitens der Anwendungen ist in Abschnitt 9.4 dargestellt.

Kontexte müssen, um sie in einem Computersystem nutzen zu können, in eine lesbare Notation überführt werden. Zu diesem Zweck wird mit der XML-basierten Notation PIDF-CE eine Kontextrepräsentation geschaffen und in Abschnitt 7.5 eingehend erläutert werden. Die Struktur des Kapitels ist in Abbildung 7.1 dargestellt.

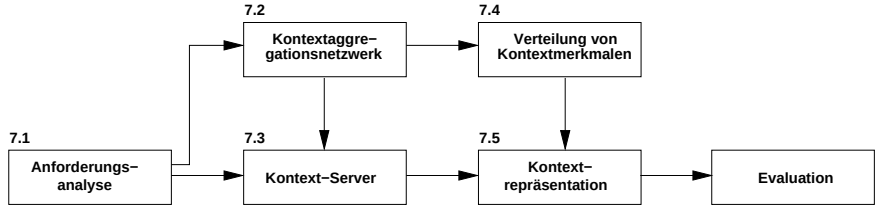


Abbildung 7.1: Graphische Strukturrepräsentation von Kapitel 7

7.1 Anforderungsanalyse

Wie in Kapitel 5 entwickelt, besteht das Gesamtsystem aus Kontextproduzenten, Kontextspeicher, -verteiler und -nutzer. Es wurde dabei eine *verteilte Architektur*, wie in Abbildung 5.6 dargestellt, gewählt. Die einzelnen Komponenten wie Sensoren, Kontextnutzer oder Kontextverteiler sind miteinander vernetzt und koordinieren sich durch den Austausch von Nachrichten. Die einzelnen Aktoren sind unabhängig voneinander und lose miteinander gekoppelt (*loosly coupled*). Sie führen ihre Prozesse autonom und nebenläufig aus, müssen aber trotzdem Ressourcen gemeinsam nutzen können. Die charakteristischen Eigenschaften des Gesamtsystems für die Unterstützung von Kontextbewussten Diensten müssen durch eine geeignete *Architektur* unterstützt werden.

Das System besteht aus einer dynamischen Anzahl an hinzukommenden und verlassenden Kontextproduzenten und Kontextkonsumenten. Diese zeichnen sich durch eine immanente *Heterogenität* bezüglich ihrer Kapazitäten und unterstützten Protokolle aus. Die bereitgestellte Infrastruktur muss sich daher durch *Skalierbarkeit* und *Offenheit*,

aber auch durch Gewährleistung von *Sicherheit* auszeichnen. Für den Entwickler soll die Infrastruktur durch *Transparenz* [ANS89, Int92a] und Abstraktion die Separation und die unterschiedlichen Eigenschaften der Komponenten verbergen. Eine weitere Anforderung, die sich aus der Anwendung für Kontext-bewusste Kommunikationsdienste ergibt, ist die Unterstützung der Echtzeitcharakteristik.

Zur Erfüllung der aufgeführten Anforderungen wurde eine *Middleware* als zusätzliche Schicht eingeführt, die die notwendigen Funktionalitäten bereitstellt. Insbesondere das Auffinden der Komponenten und deren Kommunikation untereinander sollen unterstützt werden. Die eingesetzte Middleware soll dabei dem Ende-zu-Ende-Paradigma [SRC84] folgen und die Intelligenz und Verarbeitungslogik möglichst auf Entitäten in der Anwendungsschicht verlagern, da hier oftmals mehr Wissen existiert, wie mit Informationen zu verfahren ist. Eine Middleware wird in [Ber96] definiert als:

Definition 7.1 (Middleware)
*Eine **Middleware** ist ein allgemeiner Dienst, der zwischen den Plattformen und Applikationen lokalisiert ist und eine große Zahl unterschiedlicher Anwendungen unterstützt. Plattformen sind Rechnerarchitekturen, wie Prozessoren, Speicher oder Betriebssysteme. Ein Middlewaredienst ist durch seine Schnittstelle (API) sowie durch die ihn unterstützten Protokolle definiert.*

Im Rahmen der Arbeit wurden zwei Bereiche identifiziert, in denen eine Kommunikationsinfrastruktur in Form einer Middleware notwendig ist. Zum einen wird eine Kommunikation zwischen den Sensoren und den Operatoren, die deren Daten weiterverarbeiten, notwendig. Zum anderen werden Informationen zwischen den Kontextquellen oder Kontextspeichern und den Applikationen ausgetauscht. Die Kommunikationsinfrastruktur ist in Form einer Schichtendarstellung in Abbildung 7.2 abgebildet. Dies reflektiert zum einen die verteilte Architektur aus Abbildung 5.6 und bietet zum anderen Unterstützung für die intrinsische und die extrinsische Formen der Kontextgewinnung.

Bei der *intrinsischen* Gewinnung von Kontexten sind alle Phasen des Kontext-Spiral-Models auf einer monolithischen Entität, wie in Abbildung 5.3 zu sehen, vereint. Ein mit Sensoren ausgestatteter PDA oder ein Mobiltelefon würden intrinsisch die Merkmale erfassen und den Kontext über eine Kommunikationsmiddleware verteilen.

Das Hauptanwendungsfeld dieser Arbeit bezieht sich auf *intelligente Büroräume*, so dass Sensoren in der *Infrastruktur* verteilt sind und die Gewinnung der Kontextemerkmale aus Sicht des Endgeräts des Nutzers in *extrinsischer* Form erfolgt. Die einzelnen Sensoren sind mit den verarbeitenden Entitäten vernetzt und benutzen ihrerseits eine Kommunikationsmiddleware zum Verteilen der gewonnenen Daten. Hierfür wurde eine geeignete Infrastruktur entworfen, die in Abschnitt 7.2 beschrieben ist.

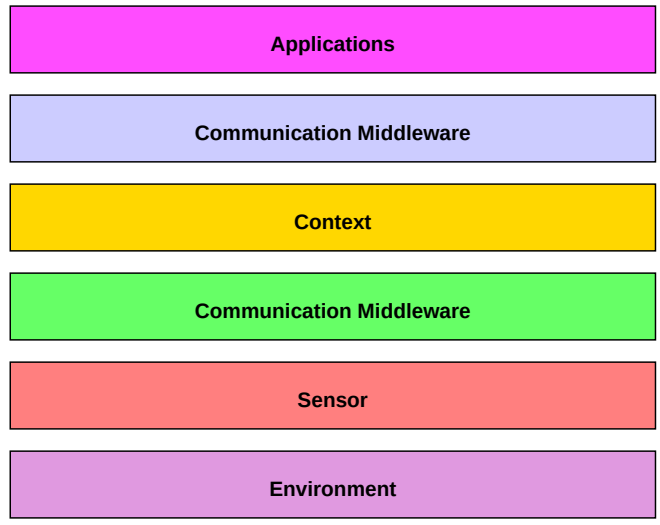


Abbildung 7.2: Schichtenbild von den Sensoren bis zur Applikation

7.2 Architektur zur Kontextaggregation

Die Aggregation von Daten zu Kontextinformationen erfordert eine unterstützende Architektur. Diese Architektur muss in der Lage sein, eine sehr große Anzahl von Daten- und Informationsquellen zu finden und abzufragen. Darüber hinaus müssen die anfallenden Daten effizient zum Ziel transportiert werden. In dieser Arbeit werden keine einzelne Entität als Aggregationspunkt gewählt, sondern ein *Kontextaggregationsnetzwerk* (*Context Aggregation Network – CAN*), in welchem die Funktionalitäten in verteilter Weise erbracht werden. Abbildung 7.3 zeigt ein solches Netzwerk.

Auf Grund der Tatsache, dass die Anzahl der Quellen wesentlich größer ist als die Zahl der Applikationen, ist ein Digraph mit einer baumartigen Struktur eine geeignete Topologie. Darüber hinaus wird in der Regel nach jedem Aggregationsschritt die Zahl der Verbindungen kleiner, da eine Aggregation aus mehreren Eingabeströmen einen Ausgabestrom erzeugt. In der Regel gilt daher:

$$grad_{\text{in}}(v) \geq grad_{\text{out}}(v) \tag{7.1}$$

Das entworfene *Aggregationsnetzwerk* besteht aus drei logischen Ebenen, wie es in Abbildung 7.3 gezeigt ist. Auf der obersten Ebene sind die Nutzer der Kontextinformationen angeordnet. Dies sind Applikationen, die Kontextinformationen anzeigen, speichern oder aber weitervermitteln. Es gibt keinen einzelnen Wurzelknoten, vielmehr eine Anzahl von Top-Level-Knoten, die als logischer Wurzelknoten gesehen werden kann. In der konkreten Anwendung handelt es sich dabei um den ContextServer, der in Abschnitt 9.4 beschrieben wird.

Die mittlere Ebene besteht aus den *Operatoren* (OP) im Netzwerk. Ein Operator führt

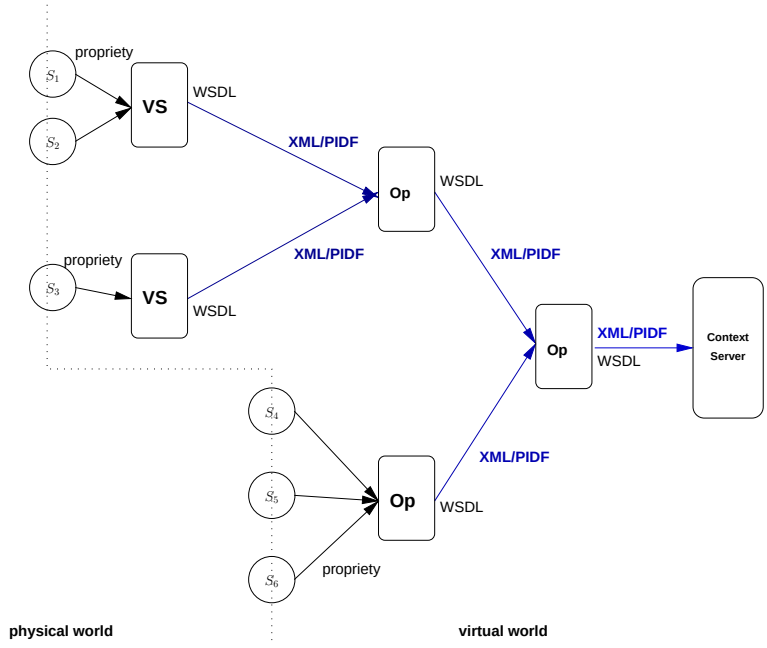


Abbildung 7.3: Architektur eines Kontextaggretationsnetzwerks

eine deterministische Funktion auf eingehende Daten aus und sendet das Ergebnis der Operation weiter. Eine Reihe von Operatoren wurde definiert und wird nachfolgend beschrieben. Die einzelnen Operatoren können kaskadiert werden und formen einen *Operatorgraphen* mit Sensoren als Quellen und Applikationen wie dem ContextServer als Senken. Die Operatoren des Netzes sind sowohl Quellen als auch Senken der Daten. Die beteiligten Komponenten senden die ausgehenden Daten jeweils in Richtung der Senken. Die Verknüpfung der Operatoren und Sensoren wird zur Zeit über eine Konfiguration in der Sprache CALL gebildet.

Operatoren empfangen Daten, verarbeiten diese und senden die Ergebnisse weiter. Die Blattknoten des Baums stellen die unterste Ebene dar. Die Blattknoten sind die originären *Datenquellen*, in Abbildung 7.3 sind es Sensoren (S_i) des Netzwerks. Die Quellen besitzen keine eingehende Kante innerhalb der Domäne des Aggregationsnetzes.

7.2.1 Operatoren

Jeder Knoten des Graphen besteht aus einer Kombination von drei Funktionsblöcken, wie dies schematisch in Abbildung 7.4 abgebildet ist. Ein Funktionsblock ist für das Empfangen von Daten zuständig. Ein weiterer Block verarbeitet die Daten und ein dritter sendet die Daten an den nächsten Knoten weiter. Eine schematische Darstellung ist in Abbildung 6.2 gezeigt. Drei Klassen von Knoten sind für das Aggregationsnetzwerk spezifiziert: *Quellen*, *Operatoren* und *Senken*. Quellen der Daten sind üblicherweise

physische und logische Sensoren, wie diese in Abschnitt 6.2 definiert worden sind. Innerhalb der Domäne des Aggregationsnetzes sind alle diese Sensoren durch *virtuelle Sensoren* gekapselt.

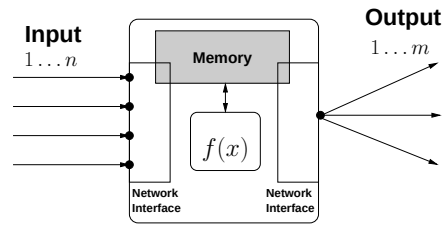
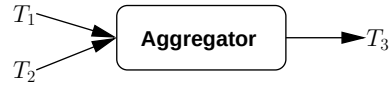


Abbildung 7.4: Schematischer Aufbau eines Operators

Drei generische Typen von Operatoren sind in dieser Arbeit für das Aggregationsnetz definiert worden. Die Operatoren *Filter*, *Transformatoren* und *Aggregatoren* sind in Abbildung 7.5 dargestellt. Verwandte Ansätze wie Solar [Che04] kennen 4 Typen von Operatoren. Die Eigenschaften und die Funktionalität jedes Typs im Aggregationsnetzwerk sind nachfolgend angegeben.

(a) Filter

(b) Transformator



(c) Aggregator

Abbildung 7.5: Verschiedene Operatoren für die Aggregation von Kontextinformationen

Filter: Eine Filterkomponente erlaubt es, den eingehenden Datenstrom auf einen Teil zu reduzieren. Die Komponente verringert die Zahl der zu transportierenden Ereignisse (*events*), aber auch die Zahl der Operationen, die die Empfänger der Nachrichten bei ihren Eingang ausführen müssen. Filter sind *zustandslose* (*stateless*) Operatoren, sie müssen keinen internen Zustand halten. Die eingehenden Daten werden direkt bearbeitet und weitergeleitet. Dadurch können diese Komponenten sehr einfach implementiert werden.

Der ausgehende Datenstrom hat dabei den selben Typ wie der eingehende Datenstrom, wie dies in Abbildung 7.5(a) zu sehen ist. Die Menge der ausgehenden Daten ist dabei eine Untermenge der Menge an eingehenden Daten.

$$|M_{out}| \subseteq |M_{in}|$$

So kann der Filter alle Daten, die Temperaturwerte unterhalb von 30 °C repräsentieren, blockieren und andere weiterleiten oder aber nur alle 60 s einen Wert versenden.

Transformator: Ein Transformator überführt Daten von einer Repräsentation in eine andere. Der ausgehende Datentyp (T_2) ist dabei in der Regel ein anderer als der Typ der eingehenden Daten (T_1). Auch der Transformator kann ohne größeren Zwischenspeicher und zustandslos realisiert werden. Die eingehenden Daten können direkt umgewandelt und weitergeschickt werden.

Die Menge an Daten bleibt bei dieser Operation gleich. Jeder Filter und jeder Transformator besitzt einen Eingangsstrom und einen oder mehrere Ausgangsströme.

$$|M_{out}|=|M_{in}|$$

Ein Transformator könnte Temperaturdaten mit der Einheit Celsius in Temperaturdaten der Einheit Fahrenheit umwandeln. Ein weiteres Beispiel ist die Umwandlung von Lokationskoordinaten in symbolische Lokationsinformationen wie beispielsweise *Mensa*, *Auto* oder *Besprechungsraum*.

Aggregator: Aggregatoren bilden die umfangreichste Klasse an Operatoren. Dieser kombiniert mehrere Eingangsströme und sendet das aggregierte Ergebnis der Operation weiter. Im einfachsten Fall werden Eingangsströme von mehreren Knoten zusammengefasst und unverändert als ein Datenstrom weitergeschickt. Aufwändigere Operationen sind Inferenzverfahren, wie diese in Unterabschnitt 6.4.1 beschrieben worden sind. Diese Operatoren kombinieren die eingehenden Daten zu höherwertigen Informationen. Dabei steigt der Abstraktionsgrad der Information an. Eine Aggregationsoperation kann auch die Generalisierung der Information zur Folge haben.

Bei dieser Art von Operation können sich sowohl die Typen als auch die Mengen der ein- und ausgehenden Daten ändern. Für einige Operationen wie gleitender Durchschnitt (*moving average*) müssen vergangene Werte vorgehalten werden, was einen Speicher erforderlich macht. Der Transformator muss daher als zustandsbehafteter (*stateful*) Operator realisiert werden.

In Tabelle 7.1 erfolgt eine Zuordnung von Operatorfunktionen, wie sie z. B. in Unterabschnitt 6.4.1 beschrieben sind, auf die drei definierten Operatortypen. Die Auflistung ist unvollständig und soll einen Überblick über gängige und in dieser Arbeit entwickelte Verfahren verschaffen.

7.2.2 CALL – Context Aggregation Level Language

Das eingesetzte Netzwerk aus einer Vielzahl an Sensoren und Operatoren wird in dieser Arbeit als *Aggregationsnetz* bezeichnet, um sich eindeutig von dem zur Zeit sehr intensiv erforschten *Sensornetzen* abzugrenzen. Das Einsatzgebiet, das für die Gewinnung von Kontextinformationen betrachtet wird, unterscheidet sich in einigen wichtigen Kriterien, wie der Anzahl der Sensoren, den betrachteten Informationen und den eingesetzten

Tabelle 7.1: Einordnung der Operatorfunktionen auf die Operationstypen

Funktion	Typ
min, max	Filter
avg	Filter
Range	Filter
F → °C	Transformer
PS → kW	Transformer
Fuzzy Logic	Aggregator
Dempser-Shafer	Aggregator
Bayesian	Aggregator
Neural Network	Aggregator

Geräte. So wird von einer administrierten und bewusst vernetzten Menge an Sensoren und Komponenten ausgegangen, wohingegen ein Sensornetz aus autonomen und per Funk sendenden Sensoren ohne zentrale Administration bestehen kann. Auch sind die Problematik des Energieverbrauchs der einzelnen Sensoren und der Einsatz von ad-hoc Routingverfahren nicht in dem Maße relevant, da sich das hier beschriebene Szenario auf Innenräume bezieht, die in der Regel über eine ausreichende Infrastruktur verfügen.

Um die Verteilung der Sensoren und Operatoren sowie deren Vernetzung zentral konfigurieren zu können, wurde die Sprache *Context Aggregation Level Language* CALL entwickelt. Die Intention der Sprache ist es, die Komponenten in der Wirkumgebung zum Einsatz zu bringen und sie administrieren zu können. Die von CALL erstellte Topologie formt ein Overlay-Netz oberhalb der vorhandenen Sensoren und Entitäten, auf denen Operatoren ausgeführt werden. Dies ist in Abbildung 7.6 gezeigt.

CALL unterstützt virtuelle Sensoren und die in Abschnitt 6.3 beschriebenen Operatoren *Filter*, *Transformer* und *Aggregator*. Jede dieser Komponenten, deren schematischer Aufbau in Abbildung 7.4 dargestellt ist, kann hinsichtlich ihrer Funktionalität parametrisiert werden. Zusätzlich wird über CALL die Verbindung zwischen den Komponenten spezifiziert, in dem die Adressen für die ausgehenden Verbindungen angegeben werden.

CALL ist eine XML-basierte Skriptsprache, welche das spezifizierte Aggregationsnetz durch Instanzierung von JAVA-Klassen realisiert. Jeder Operator ist dabei von einer generischen Mediator-Klasse abgeleitet, die die für alle Komponenten notwendigen Grundfunktionalitäten zur Verfügung stellen. Zu diesen Funktionalitäten zählen Netzwerkoperationen sowie die Personalisierung der Komponente durch das CALL-Skript. Der Funktionskern jedes Mediators wird unter Verwendung der JAVA *Reflections* API bei der Instanzierung nachgeladen. Nachfolgend wird das Konzept anhand einer Operatorkomponente erläutert.

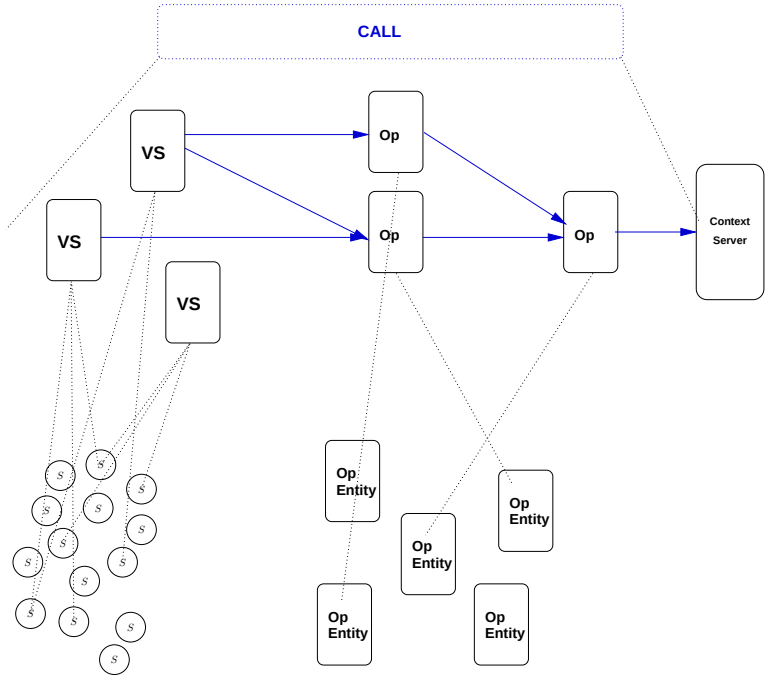


Abbildung 7.6: Konfiguration einer Overlay-Netztopologie durch CALL

7.2.2.1 Operatorkomponenten

Jede Operatorkomponente ist durch einen global eindeutigen Identifikator (*Global Unique Identifier – GUID*) bestimmt. Der Global Unique Identifier (GUID) wird aus einem Namen aus einem umgekehrten qualifizierten Domainnamensraum (*qualified reverse domain namespace name*) erstellt. Ein alternativer Ansatz ist die Verwendung des Namensschemas Intentional Naming Scheme (INS) [Sch99, Lil00]. Beide Schemata erlauben eine Gruppierung von Komponenten zu logischen Einheiten, was die Handhabung erleichtert. Jede Komponente ist über eine IP-Adresse und einem Port zu erreichen. Die IP-Adresse wird dabei entweder fest zugewiesen oder durch das Dynamic Host Configuration Protocol (DHCP) dynamisch bestimmt. Der Initialisierungteil hierzu ist in Listing 7.1. gezeigt.

Listing 7.1: Initialisierung des Operators in einem CALL-Skript

```
<?xml version="1.0" encoding="UTF-8"?>
<call xmlns="urn:ietf:params:xml:ns:call"/>
<box id="de.tud.kom.345.1">
   <address static="yes">130.83.139.161</address>
   <port>8888</port>
```

Zum Aufbau der Verbindungen werden für jede Komponente nur die ausgehenden Verbindungen spezifiziert. Für Kommunikation zwischen den Komponenten können

Sockets [Ste98] oder JAVA RMI (Remote Method Invocation) verwendet werden. Welche Operationen die Komponente bereitstellt, ist im *operation* Teil des CALL-Skripts spezifiziert. Dazu werden der Typ des Operators und anschließend die verfügbaren Operationen festgelegt. Für einen Filter wurden die folgenden Funktionen definiert und umgesetzt: min, max, avg, range, greater than, lower than.

Ein interner Timer kann verwendet werden, um während eines Zeitfensters Daten zu sammeln, bevor diese weiterverarbeitet werden. Dies ist beispielsweise zur Bildung von gleitenden Durchschnitten wichtig. Dabei wird das Zeitfenster als Vielfaches eines Timerintervalls definiert. Die Definition eines Filters, der jeweils 5 Zeitintervalle Daten sammelt und die Daten, die im Intervall $[20-30]$ liegen, dann en bloc weitersendet, ist im Ausschnitt eines CALL-Skripts in Listing 7.2 gezeigt.

Listing 7.2: Spezifikation des Operators in einem CALL-Skript

```
<operator type = "filter">
        <quantity>Temperature</quantity>
        <unit>Celsius</unit>
        <operation function = "range">
                <from>20</from>
                <to>30</to>
        <window-size>5</window-size>
</operator>
```

Während Filter einfache Komponenten darstellen, führen Aggregatoren komplexere Operationen aus. Verfahren, wie sie in Unterabschnitt 6.4.1 beschrieben sind, können als Operatoren in einem Aggregationsnetz eingesetzt werden. Im Rahmen der Arbeit wurde zum Nachweis der Machbarkeit des Ansatzes eine Fuzzy Logik Inferenz-Einheit implementiert. Diese wird beim Start der derart konfigurierten Komponente nachgeladen und ausgeführt. Die spezifische Ausführung des Funktionsblocks ist in der Anwendung gekapselt, die Konfiguration des Aggregationsnetzes bleibt davon unberührt. Bei der Erstellung der Konfiguration muss jedoch darauf geachtet werden, dass die richtige Anzahl und die richtigen Typen an Eingängen für den jeweiligen Funktionsblock vorliegen. Die Anwendungen können die Parameter `unit` und `quantity` verwenden, um die Daten zu klassifizieren.

7.3 Kontext-Server als Integrationskomponente

Wie zuvor schon erwähnt, ist ein wesentliches Entwurfsziel bei der Entwicklung von Kontext-bewussten Anwendungen die Trennung von Kontextgewinnung und Kontextnutzung. Müssen Anwendungen jeweils individuell den Kontext bestimmen, so muss jede Anwendung die ersten drei Phasen des Kontext-Spiral-Modells durchlaufen. Insbesondere die Kommunikation mit den einzelnen Sensoren ist von Heterogenität geprägt. Die Anwendungen müssen die Interaktion mit den Sensoren implementieren. Dies führt ähnlich wie beim Datenaustausch in Netzen zu einer n:m-Beziehung, bei der m Anwendungen mit n Sensoren Daten austauschen.

Ein wesentlicher Nachteil dieses Ansatzes ist, dass die Sensoren und ihre Protokolle und Eigenschaften zur Entwicklungszeit bekannt sein müssen. Weiterhin führt die schlechte Wiederverwertbarkeit bezüglich des Findens und Kommunizierens der Sensoren zu einem hohen Entwicklungsaufwand von Kontext-bewussten Anwendungen. Änderungen an der Sensorkonfiguration oder den Sensoren müssen in allen Applikationen nachgeführt werden.

Die in dieser Arbeit vorgeschlagene Lösung ist eine *Integrationskomponente*, die zwischen der Kontextgewinnung und Kontextnutzung den Dienst der *Kontextspeicherung* und *Kontextbereitstellung* zur Verfügung stellt. Dies führt, ähnlich wie bei der Verwendung von eXternal Data Representation (XDR), zum Datenaustausch zu einer n:1:m-Beziehung, wie dies in Abbildung 7.7 dargestellt ist. Die konkrete Ausprägung der Integrationskomponente, die im Rahmen der Arbeit entwickelt worden ist, ist der ContextServer.

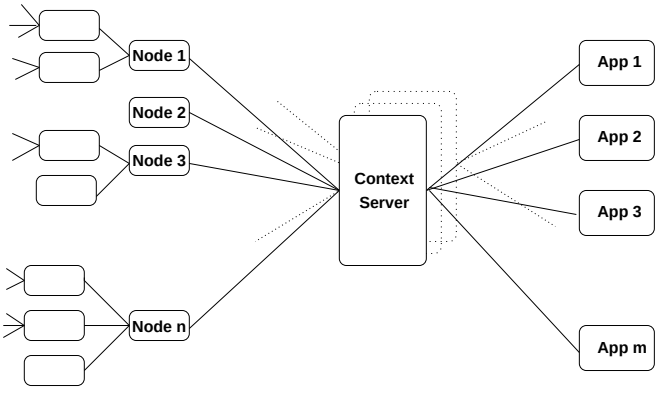


Abbildung 7.7: Einordnung des ContextServer als Integrationskomponente in die Gesamtarchitektur

Der ContextServer stellt über eine standardisierte Schnittstelle Kontexte, aber auch Sensordaten zur Verfügung. Zur Kommunikation zwischen den Anwendungen und dem ContextServer wurde ein eigenes Protokollformat namens Presence Information Data Format – Context Enhanced (PIDF-CE) definiert, welches in Unterabschnitt 7.5.2 beschrieben ist. Anwendungsentwickler können über RPC-Mechanismen auf die jeweils aktuellen Kontexte zugreifen. Das Auffinden der benötigten Sensoren und die Interaktionen mit diesen wird durch die Schnittstellen des ContextServer gekapselt, was den Entwickler entlastet. Zusätzlich müssen Änderungen am Kontextaggregationsnetzwerk oder den Sensoren nur im ContextServer berücksichtigt werden; die Kontext-bewussten Anwendungen und Dienste bleiben davon *unabhängig*.

7.3.1 Funktionsumfang

Mit einer Anwendungsfallanalyse wurden die primären Funktionen des ContextServer identifiziert und in Abbildung 7.8 dargestellt. Die wesentlichen Funktionen sind nachfolgend erklärt.

Speicherung (*storage*): Die persistente Speicherung der Kontexte, Kontextmerkmale und weiterer assoziierter Daten stellt eine wichtige Grundfunktion für weitere Funktionen dar. Die Speicherung erfolgt in einer Datenbank, die nicht notwendigerweise mit dem ContextServer ko-lokiert sein muss.

Entscheidungsfindung (*decision making*): Die Abbildung einer Menge von Sensordaten auf Kontextmerkmale und schließlich die Bestimmung des Kontexts mittels einer charakteristischen Funktion χ kann alternativ zum Context Aggregation Network (CAN) auch in einem Modul des ContextServern [s] erfolgen. Hierfür kann auch auf die in der Datenbank gespeicherten Informationen zurückgegriffen werden.

Verlauf (*history*): Die Speicherung der Kontexte und optional der Entstehungsprozesse wird verwendet, um einen Verlauf der Kontexte eines Nutzers oder einer anderen Entität aufzuzeichnen. Aus den Daten können Muster erkannt werden. Diese Information kann als Eingabe für weitere Prozesse wie die Prädiktion oder selbstlernende Verfahren dienen.

Prädiktion (*prediction*): Aus Informationen vergangener Kontexte und dem gegenwärtigen Verhalten des Nutzers kann dieses Modul mittels Prädiktionsmodelle mögliche zukünftige Kontexte vorhersagen. Diese können genutzt werden, um weitere Aktionen anzustoßen oder Informationen bereits vorzuhalten (*pre-fetching*), die von Interesse sein könnten.

Verteilung (*dissemination*): Die aktuellen Kontexte werden über unterschiedliche Mechanismen den Anwendungen und Diensten zugänglich gemacht. Über eine spezifizierte Schnittstelle ist ein Anfrage/Antwort-Mechanismus realisiert worden. Ein weiteres Verteilkonzept ist ein ereignisbasierter Dienst, bei dem sich die Anwendungen für bestimmte Kontexte registrieren und über Kontextänderungen benachrichtigt werden.

7.4 Verteilung von Kontextinformationen

Nachfolgend werden zwei Verfahren für die Verteilung von Kontexten bzw. Kontextobjekten vorgestellt. Zum einen stellt der ContextServer die gespeicherten Kontexte anderen Applikationen und Diensten zur Verfügung. Zum anderen sollen Kontexte zwischen Anwendungen direkt ausgetauscht werden können. Ein solches Szenario für einen Kontext-bewussten Dienst ist in Unterabschnitt 3.3.2 beschrieben worden.

Beide Anforderungen werden durch unterschiedliche Kommunikationsmethoden erfüllt. Der ContextServer interagiert mittels einer dienstorientierten Middleware in Form von Web Services mit anderen Anwendungen, wie dies in Unterabschnitt 7.4.2 beschrieben ist. Erweiterte SIP-User Agents, welche Kontexte In-band über das Session Initiation Protocol (SIP) austauschen können, sind in Unterabschnitt 7.4.4 erläutert.

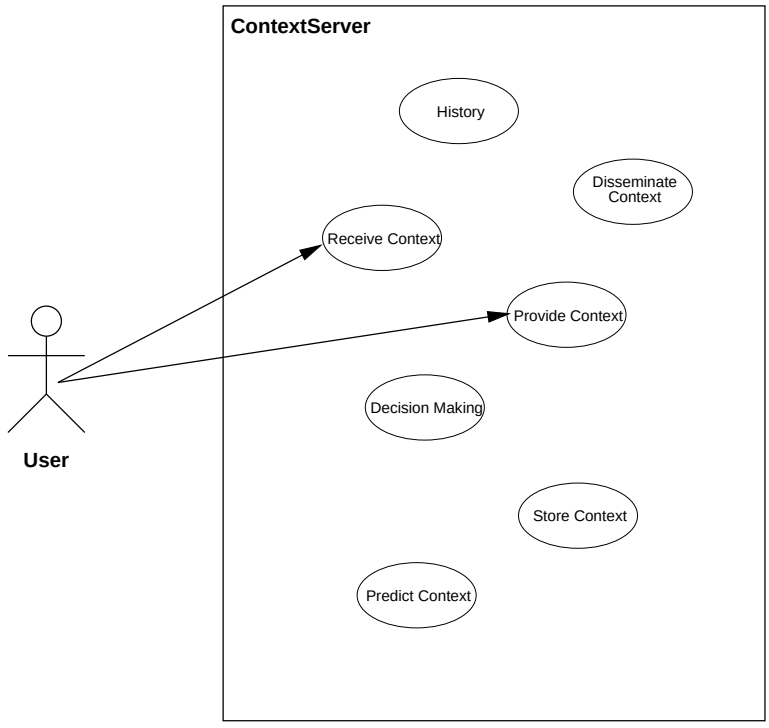


Abbildung 7.8: Primäre Funktionen des ContextServer

7.4.1 Interprozesskommunikation

Zur Kommunikation und Interaktion zwischen separaten Prozessen wurden in den vergangenen 20 Jahren eine Vielzahl an Methoden und Mechanismen unter dem Begriff *Interprozesskommunikation* (*Interprocess Communication – IPC*) vorgeschlagen. Für den lokalen Austausch von Daten haben sich *benannte Puffer* (*named pipes*) und *Shared Memory* etabliert. Zu den bekannten Ansätzen zur weitverteilten Prozesskommunikation zählen SUN's Remote Procedure Call (RPC) [Sri95], JAVA Remote Method Invocation (RMI) [www11], Common Object Request Broker Architecture (CORBA) von der Object Management Group (OMG) [www16], Microsoft's Distributed Component Object Model (DCOM) [www4] oder das OSF Distributed Computing Environment (DCE) [www5]. Jede dieser aufgeführten Methoden findet heute Einsatz in verteilten Systemen, jedoch konnte keine Methode eine übergreifende Verbreitung erreichen.

Ein aktuell stark favorisierter Ansatz ist die *dienstorientierte Architektur* (*Service Oriented Architecture – SOA*), die insbesondere in der Enterprise Application Integration (EAI) eine große Rolle spielt. Dieser Ansatz wurde als Middleware für die Bereitstellung von Kommunikation und Koordination zwischen Komponenten im Kontext-bewussten Kommunikationsdienstesystem ausgewählt. Die daraus resultierende *Service-orientierte Middleware* wird nachfolgend eingehend betrachtet.

7.4.2 Service-orientierte Middleware

Die dienstorientierte Architektur SOA stellt ein architektonisches Konzept zum Entwurf von verteilten Systemen dar. Es besteht aus *Komponenten*, die Anwendungsfunktionalität in Form von Diensten anbieten. Diese können von Anwendungen gefunden und genutzt oder mit anderen Diensten kombiniert werden. Die SOA ist eine komponentenbasierte Architektur (*component-based architecture*) auf einer hohen Abstraktionsebene, die eher auf die Zusammensetzung zu Anwendungen fokussiert als auf Implementierungsdetails.

Die einzelnen Komponenten kommunizieren mittels *Nachrichten* in einem einheitlichen Format und mittels definierter Protokolle, die auf offenen und plattformneutralen Standards beruhen. Dieser implementierungsagnostische Ansatz soll die Integration vereinfachen. Das Zusammenspiel der Komponenten wird durch Richtlinien gewährleistet. Folgende Eigenschaften zeichnen die Dienste in der dienstorientierten Architektur aus.

Wohldefiniert (*well-defined*): Jeder Dienst muss mit einer gemeinsam bekannten Definition beschrieben sein. Die Beschreibung ist dabei unabhängig von einer spezifischen Technologie.

Zustandslos (*stateless*): Die Verarbeitung der Daten in einem Dienst hängt nicht von den Zuständen anderer Dienste ab. Auch muss keine Historie an Zuständen gehalten werden.

Autonom (*autonomous*): Jeder Dienst muss selbstständig lauffähig sein. Die Verarbeitung der Daten in einem Dienst hängt auch nicht von den Zuständen anderer Dienste ab.

Eigenständig (*self-contained*): Aus der Sicht des Dienstanforderers erscheint jeder Dienst eigenständig, auch wenn dies im internen Aufbau nicht der Fall sein muss.

Lose verbunden (*Loosely-coupled*): Die einzelnen Komponenten interagieren miteinander über wohl-definierte Schnittstellenbeschreibungen und Nachrichtenformate. Diese Beschreiben die Funktionalität und die Daten und verbergen die Implementierung und Dateiformate.

Das Grundmodell der Komponenteninteraktion in einer dienstorientierten Architektur ist in Abbildung 7.9 dargestellt. Ein *Dienstanfrager* (*service requestor*) fordert einen Dienst an. Dazu setzt er eine Nachricht an den *Dienstanbieter* ab. Die Nachricht kann bereits Daten enthalten, die verarbeitet werden sollen oder mit denen andere Daten abgefragt werden können. Der Dienstanbieter sendet, nachdem der Dienst seine Verarbeitung abgeschlossen hat, die Antwort zurück. Zwischen den beiden Parteien muss zum einen Einigkeit über die Beschreibung und die Semantik herrschen und zum anderen müssen sich die Parteien vorher kennen, um die Verbindung zu etablieren.

Eine dienstorientierte Architektur wurde als Kommunikationsmiddleware zwischen den involvierten Komponenten ausgewählt. Sie erfüllt alle Erfordernisse des Systems. Darüber hinaus kann die Service Oriented Architecture (SOA) nach dem Konzept der

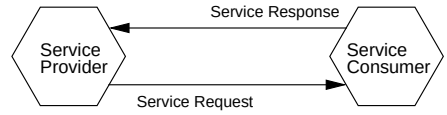


Abbildung 7.9: Austausch von Nachrichten zwischen Dienstanfrager und Diensterbringer

multiple platforms, multiple languages implementiert werden. Dadurch können sehr heterogene Systeme miteinander verbunden werden. Existierende Lösungen und Anwendungen können integriert werden. Nachfolgend wird die Realisierung der Architektur in Form von Web Services näher konkretisiert.

7.4.3 Web Services

Web Services sind eine Ausprägung der dienstorientierten Anwendungsarchitektur (*Service Oriented Architecture – SOA*). Ein *Web Service* stellt als Grundkonzept eine aus dem Netzwerk erreichbare Schnittstelle zwischen dem Nutzer und der Anwendungsfunktionalität zur Verfügung, auf die programmatisch zugegriffen werden kann. Dabei wird auf Standardtechnologien und -konzepte aus dem Internet aufgebaut. Der Web Service fungiert als Abstraktionsebene, welche die spezifischen Details der verwendeten Plattformen und Programmsprachen verbirgt.

Der funktionale Aufbau der Web Service-Architektur mit ihrem Rollenmodell ist in Abbildung 7.10 dargestellt. Im Unterschied zur Abbildung 7.9 wird ein Dienstverzeichnis in den Prozess der Dienstnutzung integriert. Der *Dienstanbieter* (*service provider*) erzeugt Beschreibungen der Dienste und publiziert (*publish*) diese bei einem *Dienstverzeichnis* (*service registry*), wo sie gespeichert werden. Der *Dienstnutzer* (*service consumer*) kann das Dienstverzeichnis nach geeigneten Diensten durchsuchen und die Adresse für den Aufruf des Dienstes erhalten.

Web Services definieren ein Framework [Boo04] für die zu verwendenden Nachrichten, interagierenden Komponenten und den Aufruf der Anwendungsfunktionalitäten. Der Aufruf entspricht dabei dem Konzept des *entfernten Methodenaufrufs* (*Remote Procedure Call – RPC*). Hierbei wird eine Methode auf einem über das Netzwerk erreichbaren Rechner aufgerufen und optional eine Antwortnachricht erhalten. Diese Techniken sind bereits in CORBA und DCOM bekannt. Jedoch zielt die Konzeption des Name-Service-Ansatzes auf die Verteilung über das Internet. Eine große Anzahl an Definitionen zu Web Services existiert mittlerweile. Die meisten beziehen sich auf die Anwendung in Geschäftsprozessen. Das World Wide Web Consortium definiert Web Services folgendermaßen [Sch02a]:

Definition 7.2 (Web Service)
*Ein **Web Service** ist eine durch einen URI eindeutig identifizierte Softwareanwendung, deren Schnittstellen als eXtensible Markup Language (XML)-Artefakte definiert,*

beschrieben und gefunden werden können. Ein Web Service unterstützt die direkte Interaktion mit anderen Softwareagenten durch XML-basierte Nachrichten, die über Internetprotokolle ausgetauscht werden.

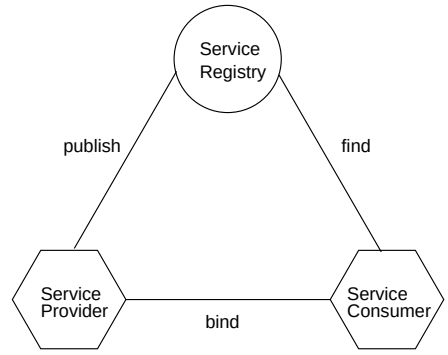


Abbildung 7.10: Funktionale Ansicht des Web Service Architektur

Der Aufbau des konzeptionellen Stacks von Web Services ist in mehrere Schichten aufgeteilt, wie dies in Abbildung 7.11 dargestellt ist. Durch die horizontale Integration der Ebenen wird eine Modularisierung für verteilte Rechnersysteme unterstützt. Der Stack lässt sich in drei horizontale Gruppen. unterteilen. Die Gruppe, die für die *Interaktion* zuständig ist, ist mit *Wire* betitelt. Die Techniken und Protokolle zur Beschreibung und Arrangierung sind in der Gruppe *Description* zusammengefasst. In der Gruppe *Discovery Agencies* sind Verfahren zum Ankündigen und Auffinden von Diensten zu finden. In der Gruppen lassen sich weitere Schichten unterteilen. Nachfolgend werden die relevanten und häufig verwendeten Schichten erläutert.

Die *Paketisierungsschicht* (*packaging layer*) stellt die Funktionalitäten (*serialization and marshaling*) zum Austausch von Daten zwischen möglicherweise heterogenen Systemen zur Verfügung. Als Datenformat wird XML und das Simple Object Access Protocol (SOAP) eingesetzt, welches XML nutzt. Die Paketierung geschieht unabhängig von den darunterliegenden Transportprotokollen (*protocol agnostic*). Nachrichten aus der Packaging-Schicht werden auf der *Transportschicht* (*transport layer*) übertragen. Hierfür stehen eine Vielzahl an Protokollen, wie HTTP, Simple Mail Transport Protocol (SMTP) oder Jabber [SA04], zur Auswahl. Gegenwärtig ist HTTP das am häufigsten eingesetzte Transportprotokoll für SOAP-Nachrichten.

Die *Richtlinienschicht* (*policy layer*) beschreibt dienstspezifische Informationen unter Verwendung von WS-Policy [BBC+04a], die die Verwendung des Dienstes näher spezifiziert. Dazu zählen Sicherheitsanforderungen, Kosten oder Dienstgüteparameter (*Quality of Service – QoS*). Web Services können miteinander verbunden werden, um einen gemeinsamen Dienst zu erfüllen. Hierzu müssen diese geeignet komponiert und orchestriert werden. Die notwendigen Techniken wie Business Process Execution Language (BPEL) [ACD+03], WS-Coordination [CCF+04] und Web Services Choreography [ABPRT03] sind hier gruppiert. Die Geschäftsbeziehungen zwischen verschiedenen Teilnehmern können mittels Service Level Agreements (SLA) spezifiziert

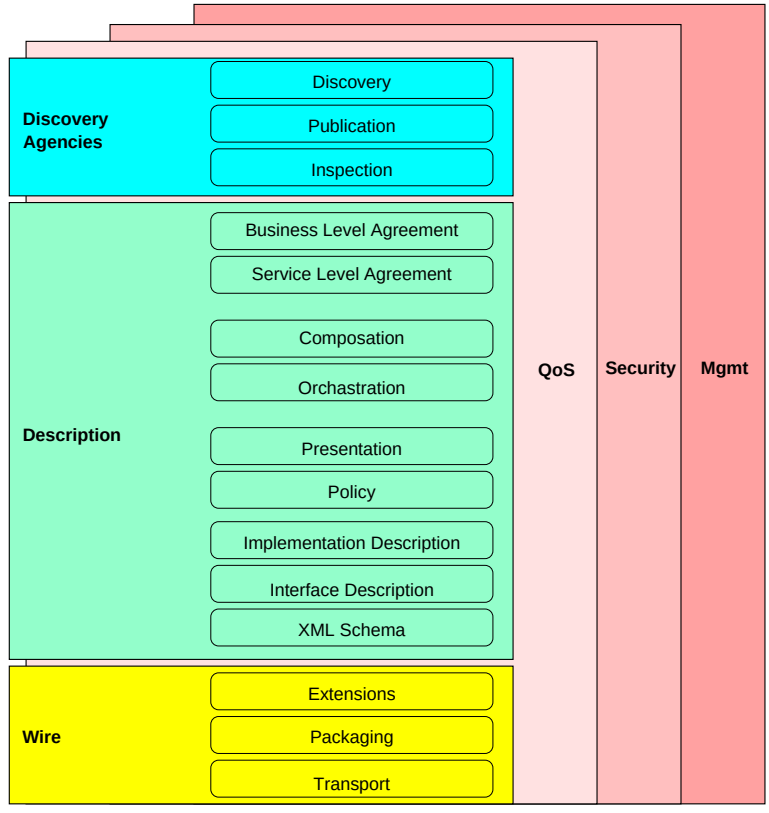


Abbildung 7.11: Schichten des konzeptionellen Web Services Stacks

werden. Hierfür gibt es erste Ansätze für die Verwendung in Web Services Umgebungen [KL03].

Auf der *Findungsschicht* (*discovery layer*) werden Mechanismen zur Unterstützung der Universal Description, Discovery and Integration (UDDI) [CHvRR04] bereitgestellt. Zur aktiven Erkundung von Diensten in Verzeichnissen (*registries*) wurde die WS-Inspection Sprache (*Web Service Inspection Language – WSIL*) [BBM+01] spezifiziert. Die Web Service Description Language (WSDL) ist zur Zeit der de-facto Standard für die Bereitstellung der *Beschreibungsschicht* (*description layer*). Ein weiterer Vorschlag hierfür ist das Resource Description Framework (RDF) [Bec04, Bri04] des World Wide Web Consortium (W3C).

Als Querschnittsebenen, die alle Schichten durchziehen seien Dienstgüte (*QoS*), Sicherheit (*security*) und Management genannt. Die Gewährleistung gewisser Parameter bei der Ausführung der Dienste wie Ausführungszeiten, Zuverlässigkeit aber auch Genauigkeiten bieten zusammen mit Preisen eine solide Basis zur Auswahl eines Dienstes durch einen Dienstanfrager. Eine Reihe von Ansätzen zur Ende-zu-Ende-Sicherheit und Vertrauen (*trust*) werden von der Organization for the Advancedment of Structured Information Systems (OASIS) [www15] standardisiert. Insbesondere für geschäftskritische aber auch für sensible Daten wie beispielsweise Kontextinformationen sind diese

Funktionalitäten sehr wichtig.

7.4.3.1 Simple Object Access Protocol – SOAP

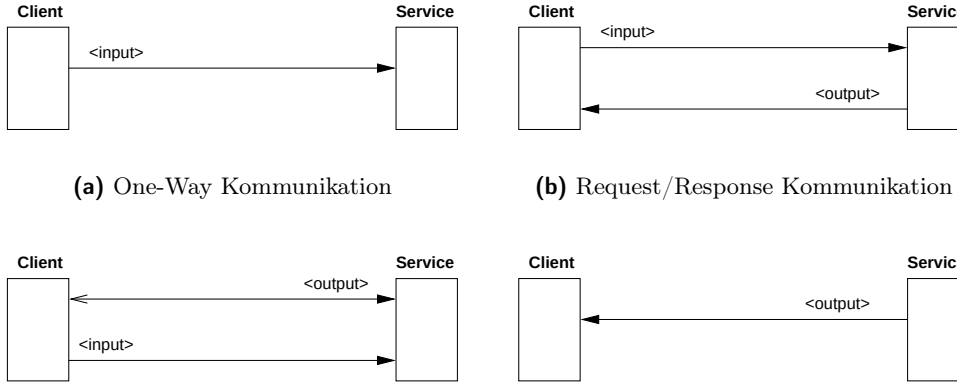
Das *Simple Object Access Protocol* (SOAP) [GHM+03] ist ein XML-basiertes Nachrichtenprotokoll. Es bietet einfache und konsistente Mechanismen, die es Anwendungen erlauben, Nachrichten auszutauschen. Hierfür wurden vier verschiedene Kommunikationsverhalten definiert.

One-way: Der Dienst benötigt nur eine eingehende Nachricht als Input für die Operation. Es wird keine Antwortnachricht vom Web Service erzeugt. Das Senden von Daten von Sensoren zur Weiterverarbeitung wäre eine Anwendung.

Request-response: Der Dienst empfängt eine Anfragenachricht und antwortet mit einer Antwortnachricht. Bei der Dienstdefinition werden dafür eine Input- und eine Output-Nachricht definiert. Klassische Anfrage/Antwort-Systeme lassen sich hiermit abbilden.

Solicit-response: Der Dienst sendet zuerst eine Nachricht, wobei der empfangende Client den Empfang der Nachricht seinerseits mit einer Antwortnachricht beantwortet. Die Reihenfolge der Input- und Output-Nachrichten dieser Operation ist bei der Dienstdefinition vertauscht.

Notification: Der Dienst sendet Nachrichten, ohne auf eine Antwort zu warten. Bei diesem Push-Dienst wird nur eine Output-Nachricht definiert.



(a) One-Way Kommunikation

(b) Request/Response Kommunikation

(c) Solicit-Response Kommunikation

(d) Notification Kommunikation

Abbildung 7.12: Kommunikationstypen zur Interaktion mit Web Services

Die SOAP Spezifikation definiert einen *XML-basierten Umschlag* für die zu transportierenden Daten sowie ein Regelwerk für die Umsetzung der anwendungs- und plattformspezifischen Datentypen in eine einheitliche XML Repräsentation. Jede SOAP Nachricht, wie Abbildung D.1 gezeigt, besteht aus einem Umschlag (*envelope*) mit einer Anzahl an optionalen Kopfzeilen und einem Nachrichtenkörper. In den Kopfzeilen können

Verarbeitungsanweisungen für den Inhalt, Routinginformationen, Authentifizierungs- und Authorisierungserklärungen (*assertions*) sowie Transaktionskontexte enthalten sein.

7.4.3.2 Web Service Description Language – WSDL

Ein Schlüsselkonzept, welches Web Services von anderen Ansätzen für einen verteilten Dienstaufruf unterscheidet, besteht in der Möglichkeit, Dienste mit *Selbstbeschreibungsfähigkeiten* zu versehen. WSDL [Chi04] ist eine XML-Grammatik und stellt dabei gegenwärtig einen de-facto Standard für die Spezifikation und Beschreibung von Diensten im Umfeld von Web Services dar. Dabei wird die WSDL wie folgt definiert:

Definition 7.3 (WSDL)
*Die **Web Service Description Language** ist eine XML-basierte Schnittstellenbeschreibungssprache, die Informationen zum operativen Betrieb des Dienstes spezifiziert. Dazu zählen Dienstschnittstellen, Implementierungsdetails, Zugriffsprotokolle und Endpunktadressen.*

Eine WSDL-Beschreibung enthält eine Schnittstellenbeschreibung des Endpunktes, unter dem der Dienst aufgerufen werden kann, eine formale Beschreibung der Anfrage- und Antwortnachrichten und eine abstrakte Notation der Bindung von konkreten Protokollen, Methoden und Datenformaten auf den Dienst. Eine Web Service Schnittstelle ist generell einer Schnittstelle von objekt-orientierten Sprachen ähnlich. Die Semantik der einzelnen Dienste wird mittels WSDL vorab definiert, in dem die Beschreibung durch informelle Informationen angereichert wird.

7.4.4 Verteilung von Kontextinformationen mit dem Session Initiation Protocol

In Unterabschnitt 3.3.2 wurde ein Verfahren zur Vermeidung von nicht-zielführender Kommunikation durch die Verteilung von Kontexten vorgeschlagen. Hierbei teilt der Angerufene dem Anrufer seinen aktuellen Kontext mit, damit dieser seine weitere Kommunikationssteuerung anpassen kann. In diesem Abschnitt wird eine *In-band Signalisierung* für ein solches Verfahren vorgeschlagen. Für die konkrete Umsetzung wurde ein SIP-basiertes Kommunikationssystem als Plattform gewählt.

Zwei prinzipielle Verfahren der Prozedur des Austauschs zwischen zwei SIP-UAs werden unterschieden. Zum einen ist hier ein *Anfrage/Antwort*-Mechanismus zu nennen, deren Ablauf in Form eines UML-Diagramms in Abbildung 7.13 dargestellt ist. Zum anderen gibt es einen *impliziten Verteilmechanismus*, den der Angerufene aktivieren kann. Der Ablauf ist in Abbildung 7.14 gezeigt. Beide Verfahren finden statt, *bevor* der angerufene Nutzer alarmiert wird („Phantomklingeln“).

Nachfolgend werden unterschiedliche Realisierungsmöglichkeiten der beiden Verfahren mittels einer Umsetzung durch Protokollsemantiken des Session Initiation Protocols untersucht und beschrieben.

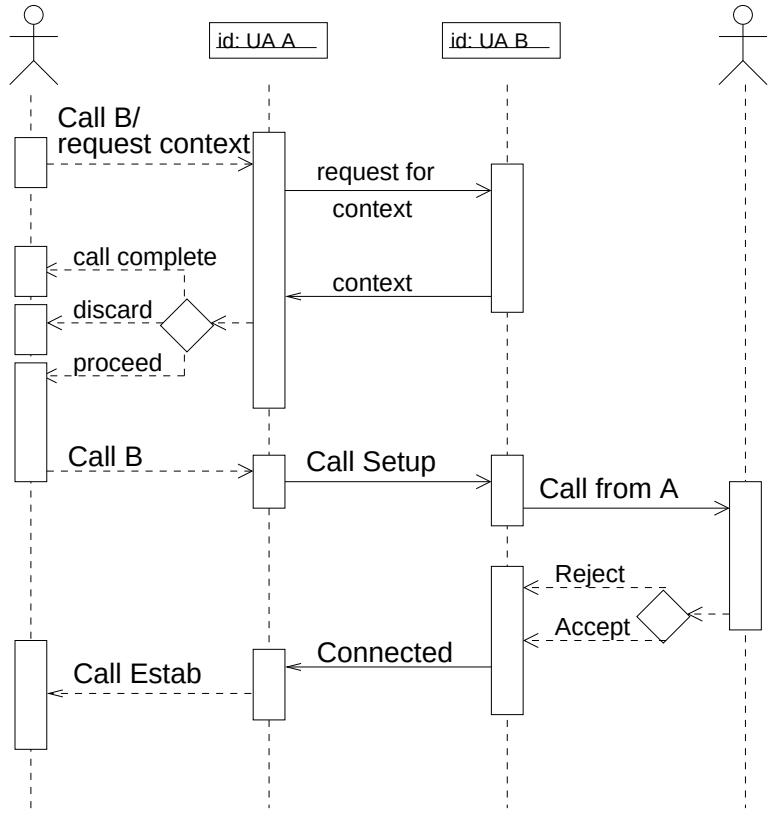


Abbildung 7.13: In-band Abfrage des Kontext

7.4.4.1 Entwurfskriterien für die Realisierung in SIP

Zwei Entwurfskriterien wurden für die Umsetzung in der Kontext-bewussten Kommunikationsinfrastruktur definiert.

1. Die notwendigen Operationen sollen transparent in den standardmäßigen Rufaufbau eingebettet sein.
2. Die Erweiterungen sollen ohne Einschränkung des Basisdienstes mit Standardkomponenten funktionieren.

Verschiedenen Methoden wurden auf die Verwendung im SIP-Umfeld und auf die Einhaltung der zwei angegebenen Entwurfskriterien analysiert. Es wurden jeweils die zwei Fälle:

1. zwischen zwei erweiterten User Agents (XUA)
2. zwischen einem standardkonformen UA und einem XUA

untersucht. Wurde eine Standard-SIP-Nachricht modifiziert, so wurde dies mit einem ‘*‘ gekennzeichnet.

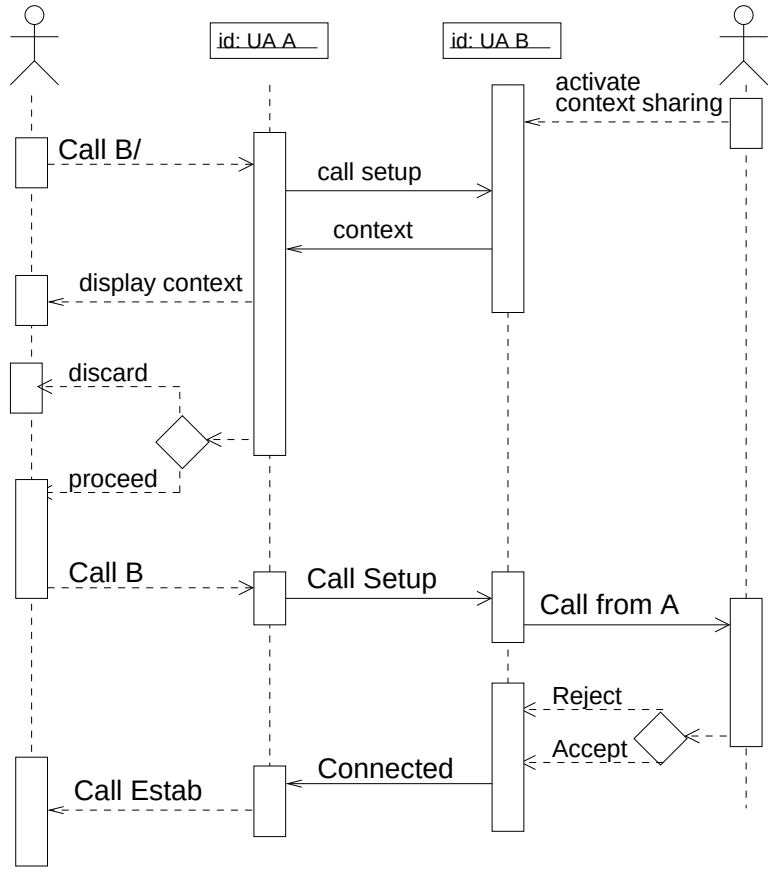


Abbildung 7.14: In-band Verteilung des Kontexts

7.4.4.2 Bewertung von Realisierungsmöglichkeiten zur Umsetzung eines Anfrage/Antwort-Mechanismus in SIP

Zur Realisierung eines direkten Anfrage-Mechanismusses nach dem Kontext des Angerufenen wurden folgende vier Alternativen untersucht.

CONTEXT: Definition einer neuen SIP-Methode, die explizit zur Anfrage des aktuellen Kontexts des Angerufenen genutzt werden kann. Der Kontext wird im PIDF-CE-Format in einer 200 Ok-Nachricht an den Anrufer zurückgesendet; andernfalls wird vom Angerufenen eine Fehlernachricht erzeugt und gesendet.

Vorteil: Der wesentliche Vorteil dieses Ansatzes liegt in der eindeutigen Signalisierung. Die Methode wird speziell für diesen Zweck eingesetzt.

Nachteil: Das Hinzufügen einer neuen Methode erfordert die Änderung der bestehenden SIP-Komponenten und des SIP-Standards. Wird die Nachricht von einer Entität empfangen, die diese neue Methode nicht unterstützt, wird eine 420 Bad Extension Fehlermeldung generiert und zurückgesendet. Wird die Methode unterstützt, so muss, basierend auf

der Entscheidung des Nutzers, eine neue INVITE-Transaktion eingeleitet werden, die vom Angerufenen als zur Kontextabfrage zugehörig erkannt wird.

OPTIONS: Die OPTIONS-Nachricht dient dem Erfragen von Informationen und Fähigkeiten eines Endsystems ohne Aufbau einer Session. Diese Methode kann verwendet werden, um den Kontext des Angerufenen als Teil der Information abzurufen. Hierzu kann das `Require` Headerfeld gesetzt werden. Auf diese Anfrage kann der User Agent mit einer Status-Nachricht antworten, die den Kontext in ihrem Nachrichtenkörper enthält. Der UA des Angerufenen antwortet mit einer erweiterten 200 Ok Status-Nachricht.

Die Kommunikation zwischen zwei erweiterten User Agents ist in Abbildung 7.15 dargestellt. In Abbildung 7.15 ist der Ablauf der Signalisierung zwischen einem XUA und einem UA gezeigt. Der nicht-modifizierte UA kann die Erweiterung der Nachricht entweder ignorieren und mit der Standard-Antwort auf eine OPTIONS-Nachricht antworten oder aber mit einer Fehlermeldung (420 Bad extension).

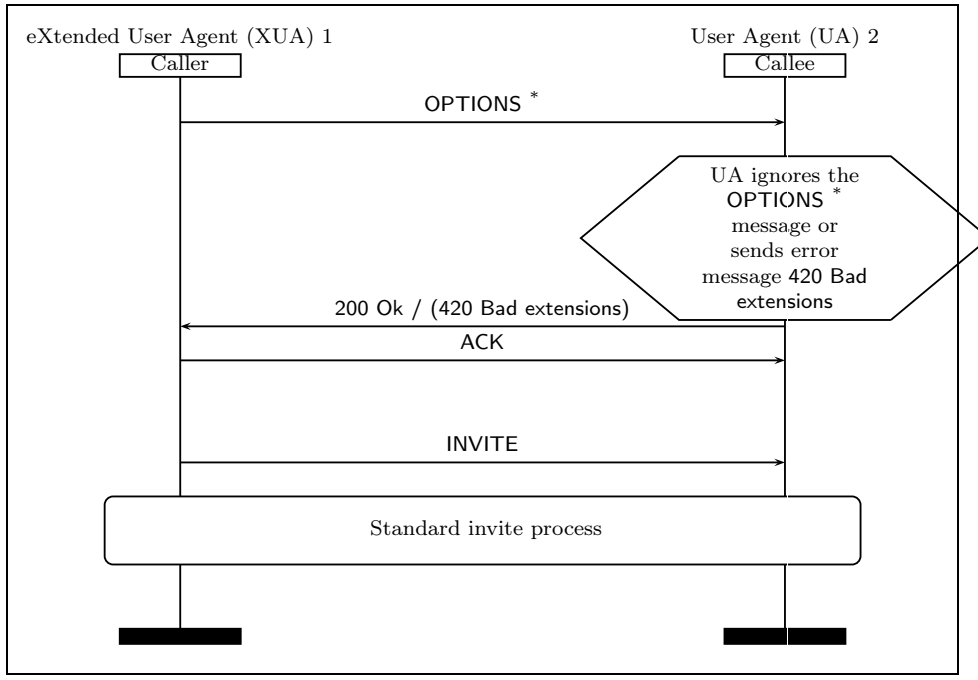


Abbildung 7.15: Abfrage des Kontexts mittels der OPTIONS-Methode zwischen UA und XUA

Vorteil: Es müssen keine Erweiterungen des SIP-Standards vorgenommen werden. Die bestehende OPTIONS-Methode wird für einen weiteren Zweck verwendet. Lediglich die beteiligten Entitäten müssen für die neue Verwendung erweitert werden.

Nachteil: Eine existierende Methode wird für einen anderen, nicht standardisierten Gebrauch eingesetzt. Dies kann zu Mehrdeutigkeiten führen.

Unterstützt ein UA die erweiterte OPTIONS-Methode nicht, so werden unter Umständen die standardmäßigen Informationen gesendet. Der anfragende User Agent muss den Inhalt auf diese Möglichkeit hin untersuchen und unterscheiden. Dies erhöht die Komplexität und die Fehleranfälligkeit.

SUBSCRIBE/NOTIFY: Das SUBSCRIBE/NOTIFY-Verfahren [Roa02] ist eine Realisierung eines asynchronen ereignisbasierten Mechanismus in SIP. Üblicherweise stellt dieses Verfahren auch den einzigen weitverbreiteten Standard hierfür dar.

Neben der Verwendung der SUBSCRIBE und NOTIFY-Nachrichten für den asynchronen Nachrichtenaustausch kann auch ein Anfrage/Antwort-Mechanismus realisiert werden. Hierbei sendet der Anrufer eine *Einmalsubskription* für den Kontext des Gesprächspartners. Empfängt der UA des Angerufenen diese Nachricht, so sendet er direkt den aktuellen Kontext in einer NOTIFY-Nachricht zurück und trägt die Subskription nicht in eine Liste offener Subskriptionen ein. Alternativ kann die Subskription gehalten werden und der angerufene UA sendet bei Änderung des Kontexts eine neue Nachricht an den Anrufer. Nach Auswertung des Kontexts auf Seiten des Anrufers kann ein standardkonformer Sitzungsaufbau ausgeführt werden. Die Signalisierung ist in Form eines MSC in Abbildung 7.16 dargestellt.

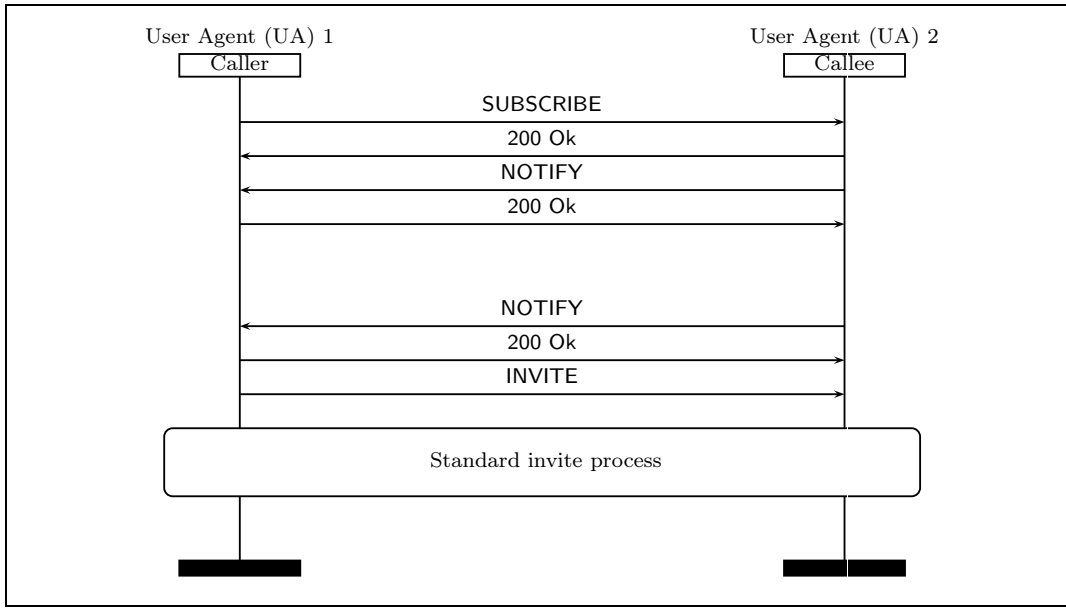


Abbildung 7.16: Abfrage des Kontexts mittels der SUBSCRIBE/NOTIFY-Methode

Vorteil: Die Methoden SUBSCRIBE und NOTIFY werden bereits zu der Erbringung eines Publish/Subscribe-Mechanismus im Rahmen der vorgeschlagenen Kontext-Verteilung verwendet. Die Verwendung zur Einmalsubskription ist technisch ohne großen Aufwand zu realisieren. Die

bereits definierten Event-Packages können in diesem Modus weiterverwendet werden.

Nachteil: Kann der angerufene UA nicht auf die in der SUBSCRIBE-Nachricht enthaltenen `Event-package` antworten, wird erst eine Fehlermeldung generiert. Danach muss vom anrufenden User Agent eine neue INVITE-Transaktion angestoßen werden. Hierzu muss die interne Verarbeitungslogik des UA erweitert werden. Zusätzlich kostet die erste Transaktion Zeit im Fehlerfall.

INVITE: Der Aufbau einer SIP-Sitzung wird mit einer INVITE-Methode eingeleitet. Die im Standard definierte INVITE-Methode wird um ein neues Headerfeld namens `Context` erweitert. In diesem Feld wird mittels des Eintrags "Request" der aktuelle Kontext des Angerufenen abgefragt. Ähnlich zum Ansatz mit der OPTIONS-Methode kann der Kontext vom XUA in eine 380 Alternative services Status-Nachricht oder eine eigens definierte $3xx$-Nachricht eingebettet werden.

In Abbildung 7.17 wird die Verwendung der erweiterten INVITE-Nachricht zwischen zwei XUA dargestellt. Ist der angefragte User Agent nicht modifiziert, so wird entweder ein standardkonformer Rufaufbau nach der INVITE-Nachricht eingeleitet oder es wird eine Fehlermeldung (z. B. 420 Bad request) erzeugt.

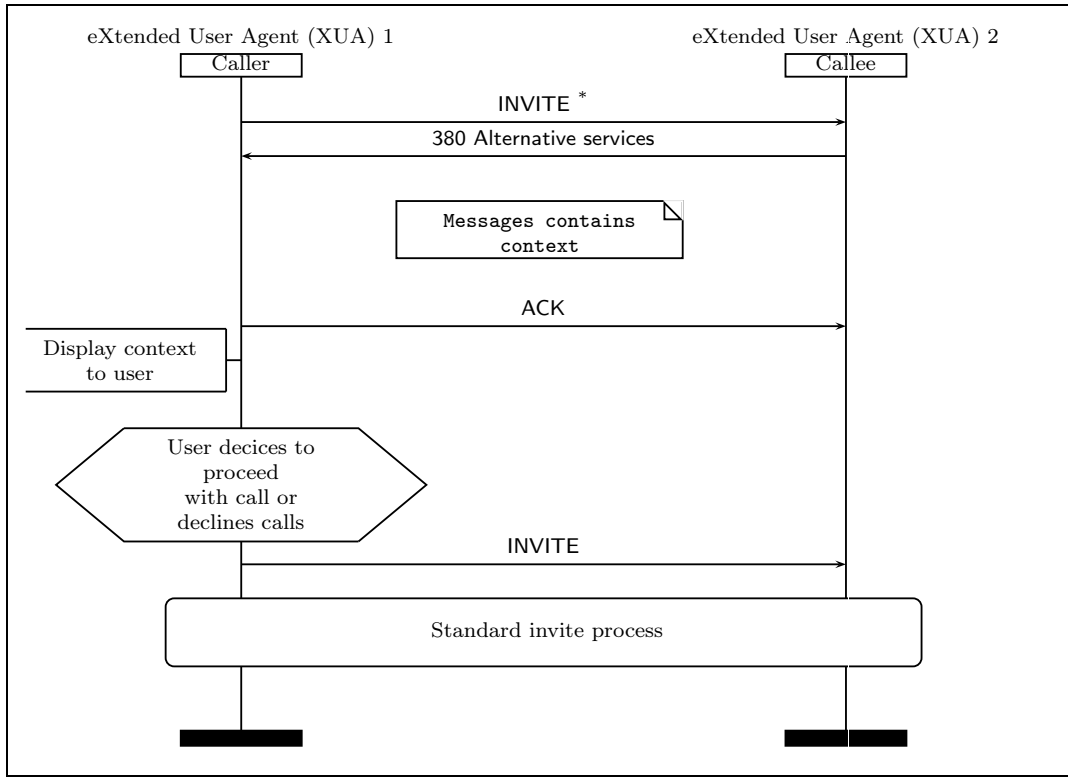


Abbildung 7.17: Abfrage des Kontexts mittels einer erweiterten INVITE-Nachricht

Vorteil: Die Verwendung des INVITE-Vorgangs fügt sich nahtlos in die Handhabung eines Standardrufaufbaus ein. Die für diese Transaktion definierten Mechanismen und Status-Nachrichten können weiterverwendet werden. Ein wesentlicher Vorteil ergibt sich im Fall der Interaktion mit einem nicht-erweiterten User Agent, da dieser das neue Headerfeld ignoriert und einen Standardrufaufbau fortführt. Dadurch entstehen keine weiteren Verzögerungen im Aufbau der Sitzung. Ein weiterer Vorteil ergibt sich bei der Interaktion mit einem CPL-Server, da dieser standardmäßig nur auf INVITE-Nachrichten ein CPL-Skript aktiviert. Bei allen anderen Ansätzen müsste die Handhabung in der CPL-Engine verändert werden. Der Angerufene kann bei diesem Ansatz Kontextinformationen ohne explizite Anforderung mitteilen. Diese werden dem Anrufer präsentiert, bevor der Sitzungsaufbau weitergeht.

Nachteil: Die Verwendung von INVITE zur Abfrage des Kontexts ist ein *implizites* Verfahren, dass die eigentliche Funktionalität überlädt. Es kann hierbei zu Uneindeutigkeiten führen. Die beteiligten Entitäten müssen erweitert werden, so dass sie das neue Headerfeld auswerten können. Ignoriert ein nicht-modifizierter User Agent die Erweiterung, so wird ein Rufaufbau initiiert. Dies kann zu einer Signalisierung des Nutzers kommen, was eigentlich verhindert werden sollte.

Bewertung der Alternativen

Die Bewertung der vorgestellten Alternativen führte zu der Verwendung der SUBSCRIBE und NOTIFY-Nachrichten. Durch Einmalsubskriptionen kann ein Anfrage/Antwort-Mechanismus realisiert werden, der eine explizite Abfrage des Kontexts ermöglicht. Zusätzlich kann diese Methoden für einen asynchronen Mechanismus, der von der Anfrage entkoppelt ist, benutzt werden. Die Abfrage über die OPTIONS bzw. CONTEXT-Methode sind als umsetzbare Ansätze zu bewerten, die bei einem nicht-modifizierten UA eine Fehlermeldung auslösen aber den Nutzer nicht stören. Abzulehnen ist die Verwendung der INVITE-Nachricht, da diese zu einem unnötigen Verbindungsaufbau führen kann.

7.4.4.3 Angerufenen initiierte Verteilung von Kontexten

Der Nutzer, der bei eingehenden Anrufen seinen aktuellen Kontext mitteilen möchte, kann dazu entweder seinen eXtended User Agent entsprechend konfigurieren oder einen CPL-Skript bereitstellen, das die in Unterabschnitt 8.2.1 beschriebene *Context Notifier*-Komponente unterstützt. Die Umsetzung durch eine In-band Signalisierung wird nachfolgend als Ablaufprozess beschrieben, der in Form eines MSCs in Abbildung 7.18 für die Kommunikation zwischen zwei erweiterten User Agents dargestellt ist. Die Signalisierung mit einem nicht-erweiterten UA ist in Abbildung 7.20 gezeigt.

Die notwendigen Änderungen zur Umsetzung in einem User Agent werden in Form von kräftig eingezeichneten Linien in der Zustandsmaschine des UA in Abbildung 9.11(b) und Abbildung 9.11(a) verdeutlicht.

Verteilung von Kontexten zwischen XUA und XUA

Der Anrufer initiiert einen standardkonformen Sitzungsaufbau durch Versenden einer INVITE-Nachricht. Da die Verteilung von Seiten des Angerufenen ausgeht, ist dies der einzige Lösungsansatz. Zusätzliche Informationen über den Grund der INVITE-Nachricht kann im *Reason-Header* [SOC02] übertragen werden. Dies verringert die Gefahr einer Fehldeutung der eingehenden Nachricht.

Der eXtended User Agent (XUA) des Angerufenen bzw. die CPL-Engine, nehmen die INVITE-Methode entgegen und erzeugen, wenn eine Authentifizierung gefordert ist, eine 401 Unauthorized (UAS) bzw. eine 407 Proxy Authentication Required Antwort (Proxy/CPL-Engine) zur Anforderung einer HTTP-Authentifizierung. Der anfordernde XUA sendet die notwendigen Zertifikate (*certificates*) in einer erneuten INVITE-Nachricht. Nach einer erfolgreichen Überprüfung wird der aktuelle Kontext des Nutzers zur Weiterverarbeitung bezogen. Dies kann ein lokal zwischengespeicherter Kontext sein, der über den Subskriptionsmechanismus mit einem ContextServer aktuell gehalten wird. Dieser Modus ist bevorzugt, da er eine schnelle Antwortzeit auf die offene Sitzungsaufbauanfrage sicherstellt. Alternativ kann der Kontext bei Bedarf durch eine Anfrage beim ContextServer bezogen werden.

Der Kontext wird vom XUA bzw. von der *Context Notifier*-Komponente in einer Beschreibung im PIDF-CE-Format in eine 183 Session progress Status-Nachricht verpackt. Dass Kontextinformationen mit der Status-Nachricht versendet werden, wird durch Einfügen eines `Context:` Headerfeldes angezeigt. Das Feld `Content-Type:` identifiziert, dass es sich bei dem Nachrichtenkörper um einen Inhalt im PIDF-CE Format handelt. Mittels des Kopfzeilenfeldes `Content-Disposition:`, das den Standard Multipurpose Internet Mail Extensions (MIME) Content-Type [TDM97] erweitert, wird spezifiziert, wie der Inhalt vom UA interpretiert werden soll. Durch Setzten des Typs auf "render" wird festgelegt, dass der Inhalt dem Nutzer angezeigt werden soll. Der Parameter `handling-param` legt fest, wie sich der UA verhalten soll, wenn der Inhaltstyp nicht unterstützt oder verstanden wird. Um eine Interoperabilität zu erreichen, wird der Wert auf "optional" gesetzt.

Die eingehende 183 Session progress Status-Nachricht wird auf Seiten des Anrufers mit einer PRACK-Nachricht quittiert. Diese vorläufige Bestätigung (*provisional acknowledgment*) hält die Sitzung offen. Diese Methoden wird vom UA des Angerufenen mit einer 200 Ok bestätigt. Nun wird der übertragene Kontext auf der Anruferseite ausgelesen und dem Nutzer angezeigt. Der Detaillierungsgrad des Kontexts entspricht dabei dem für den (authentifizierten) Nutzer zugeordneten Wert.

Der Nutzer kann, basierend auf einer Bewertung des Kontexts und des eigenen Anliegens, die weiteren Aktionen bestimmen. Um seine Entscheidung zu übermitteln, sendet der XUA eine UPDATE-Nachricht [Ros02] mit der Information, ob der Anrufaufbau fortgesetzt wird oder aber abgebrochen werden soll. Zukünftige Anwendungen können dem

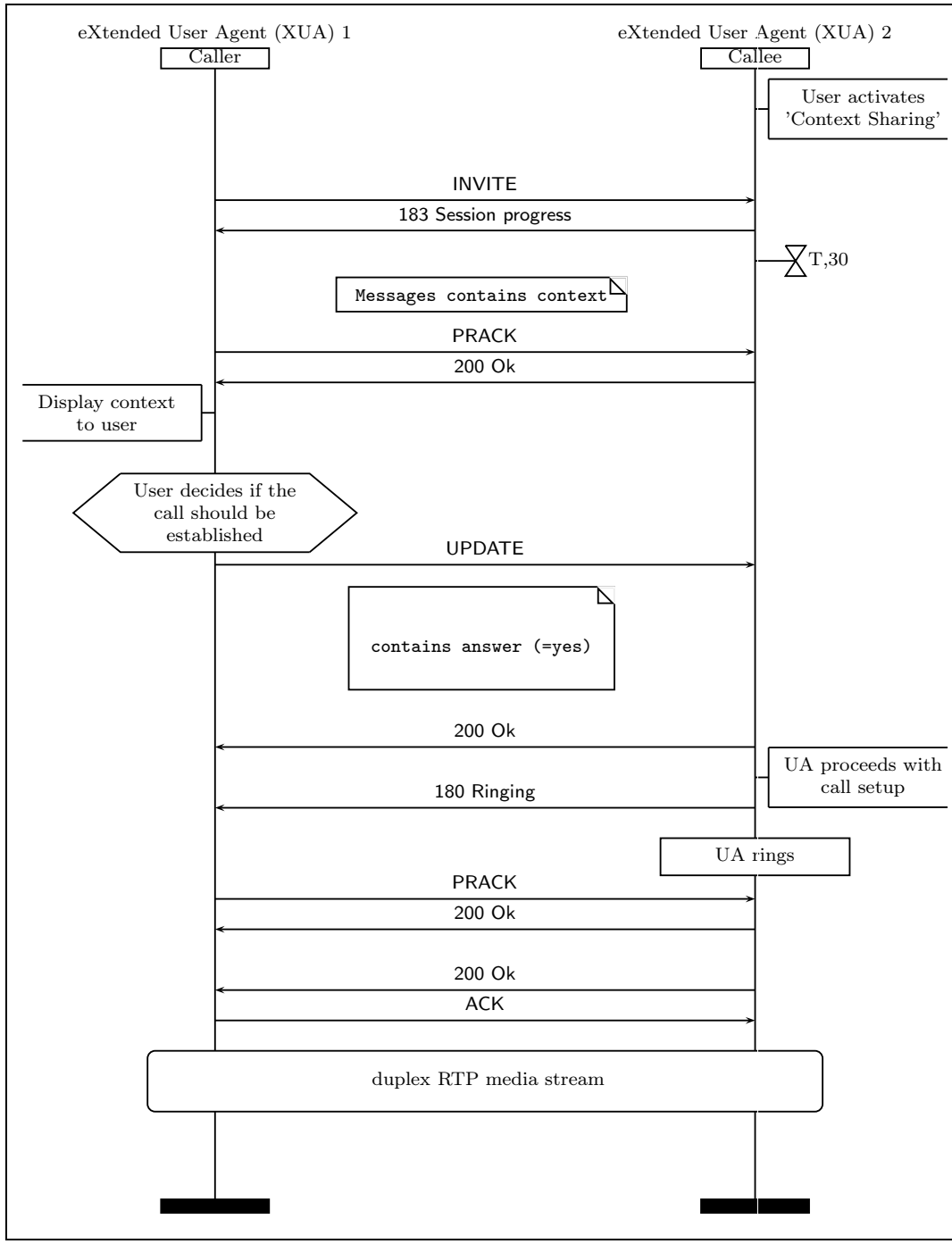


Abbildung 7.18: SIP Signalisierung zur Kontextverteilung durch den Angerufenen

Nutzer eine Reihe von weiterführenden Aktionen in Abhängigkeit der Nutzerpräferenzen und des Kontexts vorschlagen. Hierbei wird insbesondere selbstlernenden Verfahren ein großes Potenzial eingeräumt.

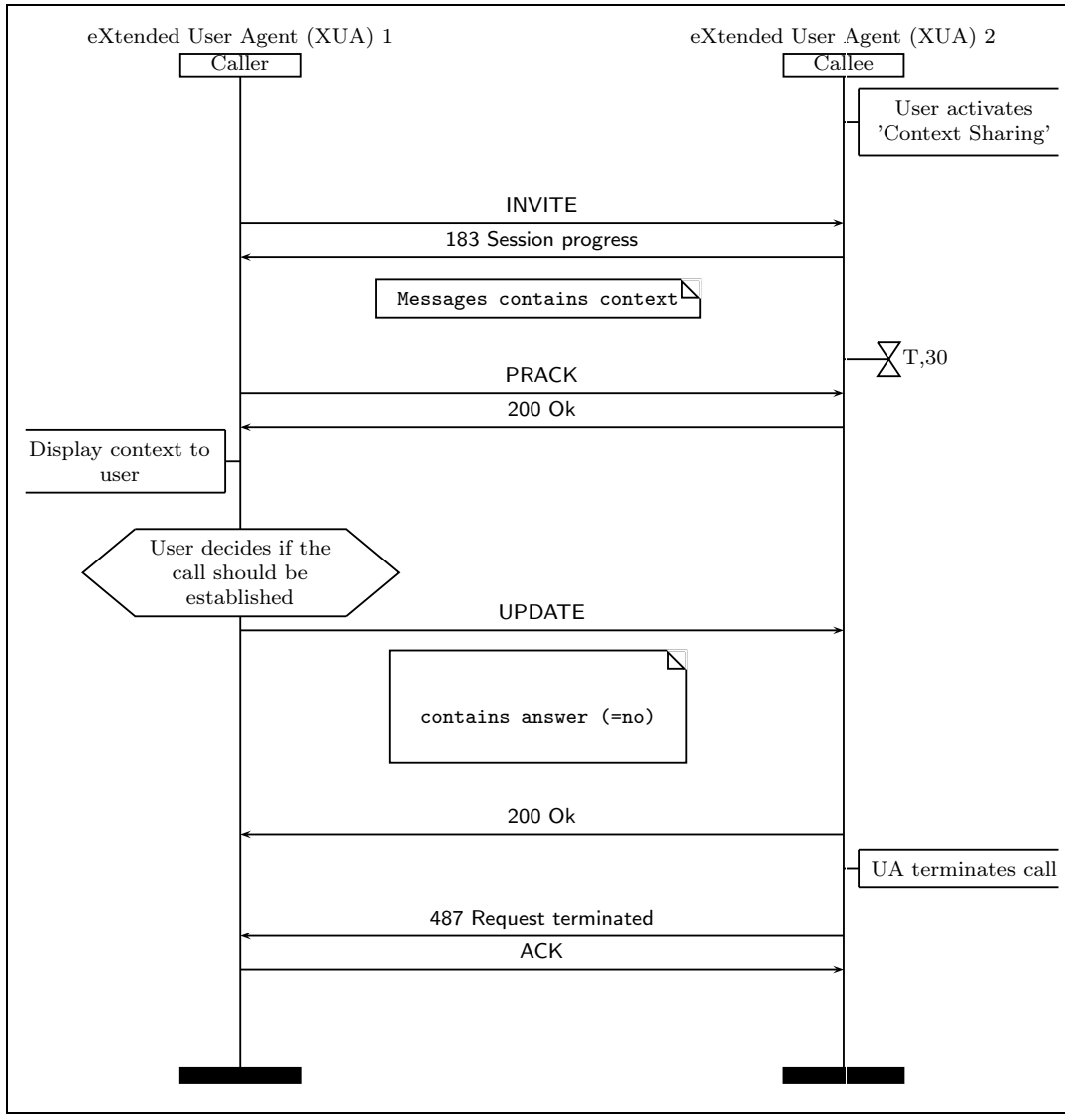


Abbildung 7.19: SIP Signalisierung zur Kontextverteilung durch den Angerufenen

Diese Methode wird ebenfalls mit einer 200 Ok-Nachricht bestätigt. Enthält die UPDATE-Nachricht die Information, dass der Anruf getätigt werden soll, so wird mit dem Rufaufbau fortgefahren. Die vom Angerufenen gesendete 180 Ringing initiiert das Rufzeichen auf beiden Seiten. Das Fortsetzen des Anrufs wird mit einer weiteren PRACK-Nachricht und 200 Ok-Nachricht bestätigt. Damit ist der provisorische Sitzungsaufbau, während dem der Kontext ausgetauscht wird, abgeschlossen und die nachfolgende 200 Ok-Nachricht und das abschließende ACK des Anrufers beenden den Sitzungsaufbau.

Soll der Verbindungsaufbau aufgrund des Kontexts des Angerufenen abgebrochen werden, so wird die UPDATE-Nachricht mit 200 Ok bestätigt. Der Aufbau der Sitzung wird danach vom Angerufenen mit einer 487 Request terminated-Nachricht abgebro-

chen und einer abschließenden ACK-Nachricht beendet. Dieser Signalisierungsvorgang ist in Abbildung 7.19 dargestellt.

Verteilung von Kontexten zwischen UA und XUA

Der Ablauf zwischen einem nicht-erweiterten UA und einem XUA ist in Abbildung 7.20 abgebildet. Der anrufende User Agent ist sich der Erweiterung nicht „bewusst“, so dass in den standard-konformen Sitzungsaufbau umgeschaltet werden muss. Nach Absenden der 183 Session progress-Nachricht, die den Kontext enthält, wird ein Timer T im UA des Angerufenen gestartet. Nach Ablauf des Timers wird davon ausgegangen, dass der anrufende UA nicht über die notwendigen Erweiterungen verfügt.

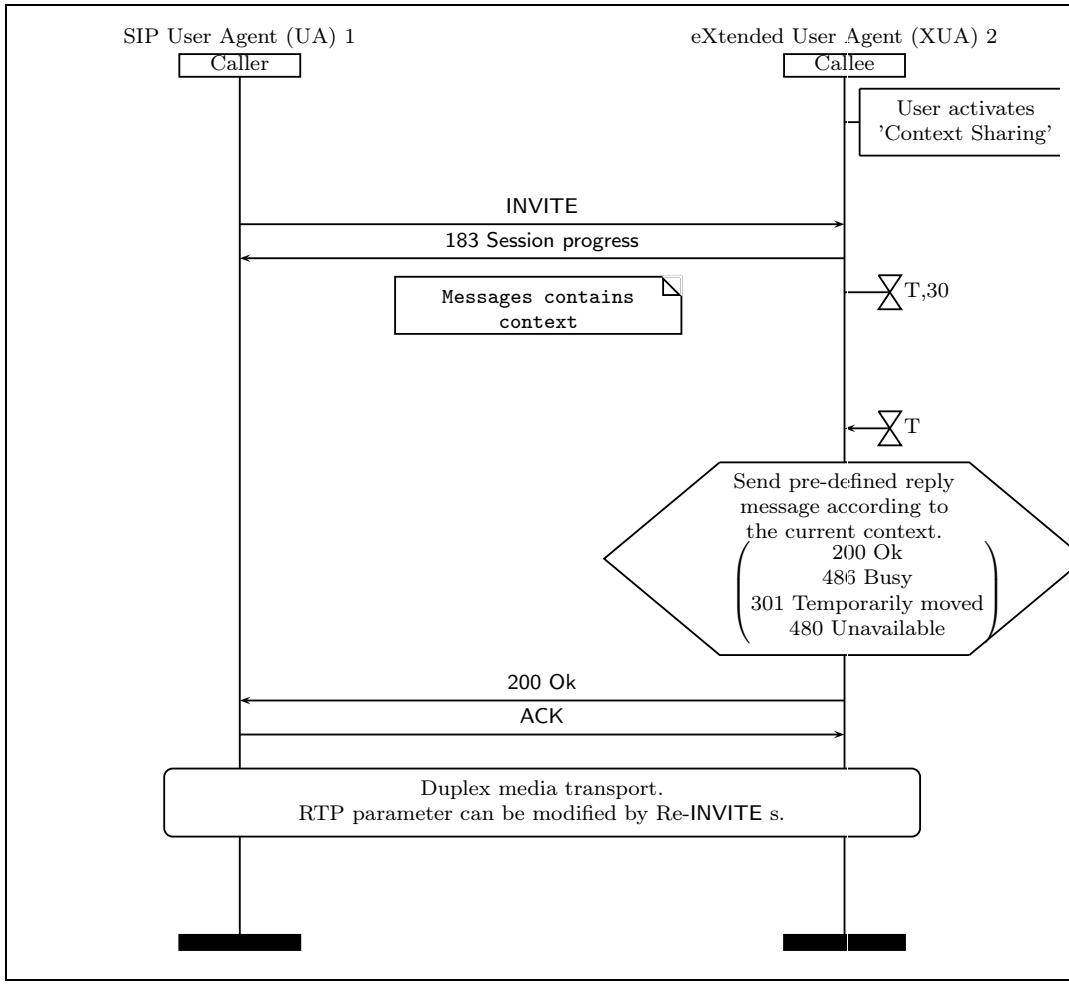


Abbildung 7.20: SIP Signalisierung zur Kontextverteilung durch den Angerufenen

Der Angerufene kann die weitere Behandlung von seinem Kontext abhängig machen. So kann er den Anrufaufbau fortsetzen (200 Ok) oder den Anruf zu einem anderen Teilnehmer umleiten (302 Temporarily moved). Weiterhin können Information über das

Ablehnen des Anrufes mitgeteilt werden (486 Busy here, 480 Unavailable). Der Anrufer beendet den Verbindungsaufbau mit einer ACK-Meldung.

Bewertung des Verfahrens

Der Vorteil dieser Methode ist, dass ein User Agent, der nicht über die beschriebene Erweiterung verfügt, nach der Status-Nachricht mit dem standardkonformen Sitzungsaufbau fortfahren kann. Diese Fallback-Strategie ist für beide Parteien akzeptabel, da diese zu keiner Verschlechterung des Status Quo führt. Alle weiteren Aktionen können durch Beibehaltung eines eindeutigen Identifikators im Headerfeld `Call-ID:` dieser Sitzung zugeordnet werden.

7.4.4.4 Sicherheitsbetrachtung

Die übermittelten Kontexte stellen sehr sensible Informationen dar. Für das Session Initiation Protocol wurden bereits eine Reihe von kryptographischen Verfahren definiert, die hierfür zum Einsatz kommen können.

Zur Authentifizierung der Teilnehmer wird im SIP-Standard vorgeschlagenen, dies mittels einer HTTP-auth Authentisierung [FHBH+99] durchzuführen. An die so authentifizierten Nutzer kann anschließend der Kontext im entsprechende Detailgrad mitgeteilt werden. Zur kryptographischen Absicherung der transportierten Daten können die Verfahren Secure MIME (S/MIME) [GMCF95, BR99, Hou99] oder TLS [DA99, Cho02] verwendet werden.

Die spezifizierten Verfahren bieten eine Ende-zu-Ende-Sicherheit. In manchen Fällen ist, es jedoch von Vorteil, dass intermediäre System auf die Daten zugreifen können. Daher gibt es zum einen die *partielle Verschlüsselung*, die Headerfelder, die für das Routing notwendig sind, unverschlüsselt lässt. Ein Ende-zu-Mitte-Sicherheit wird mit den Ansätzen [OT03, Bar03, Mah03] verfolgt. Mediendaten können über S/RTP [BMC+03] abgesichert werden. Ein Schlüsselaustauschverfahren wurde mit MIKEY [ACL+04, Euc03] definiert. Die beschriebenen kryptographischen Verfahren bieten alle notwendigen Methoden, um einen sicheren Austausch der Kontexte zu gewährleisten.

7.5 Kontextrepräsentation

Die Kontextobjekte c, welche im ContextServer gespeichert oder zwischen zwei XUA verteilt werden, müssen in einem definierten Format vorliegen. Dieses Format ist ein n-Tupel, das sowohl den Kontextbezeichner λ, als auch weitere Informationen zum Kontextobjekt, wie sie beispielsweise zur Realisierung der räumlichen und temporalen Gültigkeit notwendig sind, beinhaltet. Auch soll das Format für Kontextmerkmale ζ und auch Sensordaten s verwendet werden können.

7.5.1 Anforderung an ein Format zur Kontextrepräsentation

Ein einheitliches Format für die Kontextinformationen ist erforderlich, um einen Austausch der Daten zwischen verschiedenen Instanzen zu ermöglichen. Für die Repräsentation der Kontextinformationen muss ein Format definiert werden, das alle nachfolgend gestellten Anforderungen erfüllt.

Standardisiert: Die gewählte Repräsentationsform muss auf einem bekannten Standard basieren. Diese Forderung erhöht die Anzahl der zur Verfügung stehenden Werkzeuge und Anwendungen, die bereits mit dem standardisierten Verfahren umgehen können. Zusätzlich wird die Interoperatibilitätswahrscheinlichkeit vergrößert, was für den Austausch zwischen heterogenen und verteilten Komponenten wichtig ist.

Erweiterbar: Die Kontextrepräsentation muss erweiterbar sein und damit auch das zugrunde liegende Format. Dabei sollte das Format Abwärtskompatibilität ermöglichen, um Anwendungen (*legacy systems*), die nur das ältere Format unterstützen, nicht komplett auszuschließen. Dabei sollten die Erweiterungen mit wenig Aufwand hinzugefügt werden können, ohne das gesamte Format neu zu definieren.

Hierarchisch: Kontextinformationen sollen in hierarchischer Form repräsentierbar sein. Daher muss das zu verwendende Format die geforderten Hierarchiestufen unterstützen.

Schützbar: Kontextinformationen sind sehr sensible Daten. Daher ist es notwendig, dass diese mittels kryptographischer Verfahren gesichert werden können. Hierbei muss sichergestellt werden, dass die übermittelten Daten von einer bestimmten Person stammen (Authentifizierung) und nicht manipuliert worden sind (Integrität). Zusätzlich müssen die Daten auch vor Einsichtnahme von nicht autorisierten Personen geschützt (Verschlüsselung) werden. In einigen Fällen ist es vorteilhaft, wenn die Kontextinformationen partiell verschlüsselt werden können. Dabei bleiben bestimmte definierte Bereiche der Kontextnotation lesbar, während andere verschlüsselt sind. So können bei einer hierarchischen Notation die Basisinformationen lesbar bleiben und die weiteren Detaillierungsstufen geschützt werden.

Für die Kodierung der Daten stehen zwei prinzipielle Verfahren zur Verfügung: binäre und textbasierte Formate. Binäre Kodierungen bieten den Vorteil der sehr kompakten Darstellung und den damit resultierenden kleinen Platzbedarf. Dadurch kann die Anforderung an die benötigten Bandbreiten für die Übertragung gering gehalten werden. Häufig verwendete binäre Kodierungen sind beispielsweise die eXternal Data Representation (XDR) [Sun87] oder die Abstract Syntax Notation No.1 (ASN-1) [Int97c], welche im Umfeld von H.323 zur Kodierung der Program Data Units (PDU) eingesetzt wird. Der wesentliche Nachteil einer solchen Kodierung ist die nur schwer zu erzielende Erweiterbarkeit des Formats. Zusätzlich müssen Werkzeuge zur Verfügung stehen, um die Daten zu kodieren und dekodieren.

Textbasierte Formate, zum Beispiel in einem ISO 10646 Zeichensatz mit UTF-8 Kodierung, sind in der Regel menschenlesbar und benötigen keine speziellen Werkzeuge zur Erstellung und zum Lesen der kodierten Information. Darüber hinaus wird die Weiterverarbeitung der Daten in Programmiersprachen oder Skriptsprachen wie awk [AKW88] oder TCL/Tk [Ous94] direkt unterstützt. Nachteilig wirken sich diese Lesbarkeit sowie die üblicherweise höhere Aussagekraft auf den benötigten Speicherbedarf aus. Daraus resultiert auch eine größere Bandbreitenanforderung. Eine Repräsentation in textueller Form kann mittels einer großen Zahl an verschiedenen Transportmechanismen, wie SIP oder als MIME E-Mail Nachricht übertragen werden.

Die textuelle Repräsentation der Daten kann auf unterschiedliche Weise erfolgen. So kann der zu repräsentierende Inhalt in einer Komma-separierten Liste (*Comma Separated Values – CSV*) dargestellt werden. Alternativ sind Darstellungen gebräuchlich, in der die Daten in Form von `<type>=<value>` Paaren abgelegt sind, wie dies in Session Description Protocol (SDP)-Notationen der Fall ist. Die Verwendung von Auszeichnungssprachen ist eine weitere Notation zur Repräsentation, die an Verbreitung zunimmt, nicht zuletzt durch XML. Eine Auszeichnungssprache (*markup language*) besteht aus Zeichendaten sowie Markups. Ein *Markup* ist eine Beschreibung der Aufteilung auf Speicherungseinheiten und der logischen Struktur des Dokuments. Markups sind Marker, Entity- und Zeichenreferenzen, Kommentare, (Dokument-) Typdeklarationen sowie Verarbeitungsanweisungen (*processing instructions*). Ein Marker beginnt mit einem öffnenden Start-Tag (`<unit>`) und einem schließenden End-Tag (`</unit>`), in SGML mit spitzen Klammer gekennzeichnet. Zeichendaten des Dokuments sind derjenige Text, der kein Markup ist.

Auf der Basis der Bewertung der möglichen Repräsentationsform anhand der oben beschriebenen Anforderungen sowie der praktischen Umsetzbarkeit innerhalb des Gesamtsystems wurde die Auszeichnungssprache XML ausgewählt. Zusätzlich wird die Erstellung, Validierung, Aufbereitung (*parsing*) von XML-Dokumenten durch zahlreiche Programme und Mechanismen wie XPath [BBC+04b], XPointer [DJG+02] oder XQuery [BCF+04] unterstützt. Nachfolgend sollen die Eigenschaften dieses Datenformats beschrieben werden.

7.5.1.1 Extensible Markup Language – XML

Der Einsatz der eXtensible Markup Language (XML) [BPSM+04] als Format zum Austausch von Daten hat in jüngster Vergangenheit stark zugenommen. XML-basierende Formate werden zur Speicherung von Textdokumenten, Datensätzen oder Inhalten, die in verschiedene Darstellungsformen überführt werden können, verwendet. XML ist eine Untermenge der Standard Generalized Markup Language (SGML) [Int86] und ein Anwendungsprofil (*application profile*).

XML stellt dabei eine *Meta-Markup Sprache* dar, mittels der eine Syntax für die Spezifikation von semantischen und strukturierten Auszeichnungssprachen definiert werden kann. XML besitzt keinen festen Satz an Tags und keine Formatierungsvorschriften. XML bietet die Möglichkeit, den Inhalt eines Dokuments von der Formatierung zu

trennen. Durch die Definition von beliebigen eigenen Elementen kann die Struktur und die Bedeutung durch Verwendung der Document Type Definintion (DTD) oder deren Nachfolger XML-Schema [Fal00], beschrieben werden.

Eine *Document Type Definintion* (DTD) ist eine reguläre und formale Grammatik, die unter Verwendung der Extended Backus-Naur-Form (EBNF) [Int96a] notiert wird. Diese beschreibt die im Dokument erlaubten Elemente, Attribute, Notationen und Entitäten, sowie die Beziehungen untereinander. DTDs sind stark dokumentenorientiert und stellen eine eigenständige Informationseinheit dar. Eine Reihe von Einschränkungen der DTDs sowie die großen Unterschiede zu der Notation von XML führten zur Entwicklung von *XML Schema* durch das W3C und wurden im Mai 2001 als W3C Recommendation veröffentlicht. XML Schema erlaubt eine Datenstrukturen-orientierte Spezifikation von Struktur und Inhalt.

XML Schema bietet gegenüber der DTD eine erweiterte Datentypunterstützung, die über die vier Typen der DTD hinausgeht und ein flexibles Typensystem, das der Anwender erweitern kann. Zusätzlich erlaubt XML Schema die Wiederverwendbarkeit durch Vererbung und Namensräume.

7.5.2 Presence Information Data Format - Context Extended (PIDF-CE)

Für die Verteilung von Kontextinformationen und Kontexten wurde ein einheitliches Datenformat entwickelt, das die Anforderungen des in dieser Arbeit beschriebenen Systems erfüllt. Hierzu zählen die Unterstützung aller Attribute der virtuellen Sensoren, und die Möglichkeit, Privatsphäre durch kryptographische Methoden und Hierarchien von Informationsdetails zu gewährleisten. Auch sollen alle Informationsklassen, wie sie in Unterabschnitt 4.3.1 beschrieben sind, repräsentiert werden und die Verknüpfung der Information mit den jeweiligen Operatoren soll abgespeichert werden können.

Zusätzlich wurde beim Entwurf auf Erweiterbarkeit und Interoperabilität zu existierenden Formaten geachtet. Hierzu wurden vorhandene Formate identifiziert und auf Wiederverwendbarkeit untersucht. Es wurde eine spezielle Lösung entwickelt, da es, wie auch bei der Kontextnotation, kein oder ein nur sehr komplexes und ineffizientes Format für alle möglichen Domänen und Anwendungen geben kann. Ausgewählt und entsprechend erweitert wurde das für Instant Messaging mit SIP [CRS+02b] eingesetzte PIDF, das nachfolgend näher erklärt wird.

7.5.2.1 Presence Information Data Format – PIDF und Erweiterungen

Das Format wurde für den Austausch von Präsenzdaten (*presence data*) in einem Instant Messaging System entworfen. Das Standardformat sowie ein Rahmenwerk für Erweiterungen wurde in [DAMV00] definiert. Ein Minimalsatz an Statusinformationen für Präsenzinformationen, wie sie von der Instant Messaging and Presence Protocol (IMPP) Working Group [www10] der Internet Engineering Task Force spezifiziert wurde, ist in [DRS00] definiert. Das *Presence Information Data Format* (PIDF) [SFK+04]

wurde als ein zu Common Profile for Presence (CPP) und Common Profile for Instant Messaging (CPIM) konformes Datenformat in dieser Working Group spezifiziert. Auf diese Version wird nachfolgend mit *CPIM-PIDF* referenziert. Aktuell wird in der Working Group SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE) [www25] ein Standard entwickelt, der eine Interoperabilität zwischen den unterschiedlichen Formaten anstrebt. Eine erweiterte Version von PIDF unter dem Namen Rich Presence Information Data Format for SIP (RPIDS) [SGKR04] wurde von der Working Group veröffentlicht.

Ein PIDF-Objekt ist ein wohlformatiertes XML-Dokument, in welchem das Element `<presence>` alle Daten einer *Presentity* in Form von geordneten Tupeln (im `<tuples>` Element) enthält. Abbildung 7.21 zeigt eine solche Anordnung. Ein exemplarisches PIDF-Dokument ist in Anhang C gezeigt. Das Pflichtelement `<status>` beinhaltet den aktuellen Status der repräsentierten Entität. Die Erweiterungen der PIDF-Spezifikation können durch Verwendung von Namensräumen (`'urn:ietf:params:xml:ns:pidf'`) vom Basisnamensraum unterschieden werden. Die Spezifikation *Presence Information Data Format – Context Enhanced* (PIDF-CE) fasst die beschriebenen Erweiterungen zusammen. Die Erweiterungen werden durch den Namensraum
`'xmlns:pidfce="urn:ietf:params:xml:ns:pidfce"'`
identifiziert.

Durch die Verwendung von Tupeln können verschiedene Kontextquellen unterschieden werden und deren Daten in einem einzigen Dokument transportiert werden. Ein zwingendes Identifikationsattribut, das eindeutig sein soll, erlaubt die Unterscheidung der einzelnen Quellen. Zusätzlich wird jedes Tupel mit einem `<timestamp>`-Element versehen, was zur Archivierung und für Sicherheitsmechanismen genutzt werden kann. Eine wesentliche Einschränkung von CPIM-PIDF ist die sehr spartanische Beschreibung der Informationsquellen mittels des `<basic>`-Elements, welches nur anzeigt, ob die Entität *offen* (`'open'`) oder *geschlossen* (`'closed'`) ist. Nachfolgend wird ein Teil der Erweiterungen beschrieben.

Das PIDF-CE Format ist speziell für die Beschreibung der Aggregation von Informationen zu Kontexten konzipiert. Jeder Kanal (*channel*) entspricht dabei einer Quelle, die mit einer Reihe von Attributen, wie sie für virtuelle Sensoren in Abschnitt 6.3 spezifiziert worden sind, näher beschrieben werden kann. Der Zugriff auf einen solchen Kanal kann durch *Authorisierungsklassen* beschränkt werden. Die vordefinierten Klassen können erweitert werden.

Jede Informationsquelle ist an eine Entität oder an einen Ort gebunden. Das `<where>`-Element enthält die Informationen über den Ort. In der aktuellen Fassung ist dies eine Auflistung von symbolischen Lokationen wie `In-office`, `At-home`, `Public-area` oder eine Raumangabe wie `S3|06/345`. Eine genauere Angabe der Ortskoordinaten kann durch Verwendung des Open-GIS-Formates, von der Open Geospatial Consortium (OGC) [www17] erreicht werden.

Ein Kontext kann durch die Attribute `since` und `until` in seiner zeitlichen Gültigkeit bewusst beschränkt werden. Die optionalen Elemente `<past-context>` und

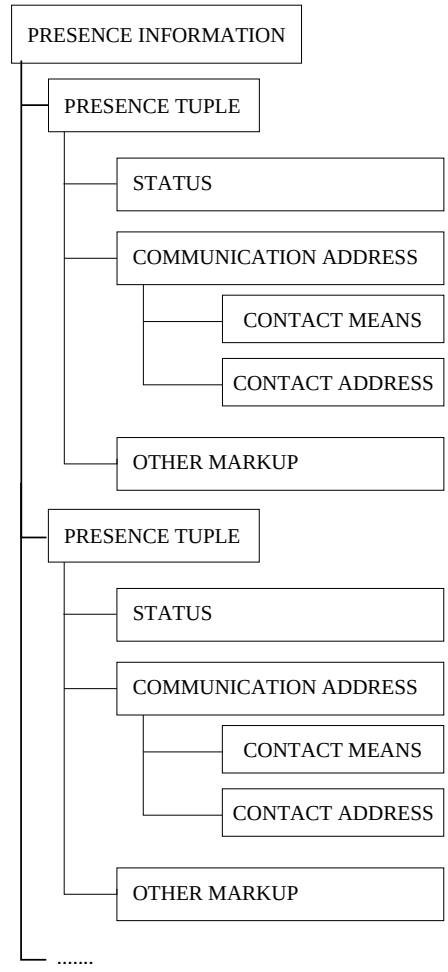


Abbildung 7.21: Struktur eines PIDF-Dokuments

`<future-status>` können verwendet werden, um weitere Informationen zum Kontext der Entität zu übermitteln.

7.6 Evaluation

Aus den an das System gestellten und in Form einer Anwendungsfallanalyse beschriebenen Anforderungen wurde eine geeignete Kommunikationsinfrastruktur abgeleitet. Diese besteht aus einer dreigeteilten Architektur, die aus einem Kontextaggregationsnetzwerk CAN, einem ContextServer als Integrationskomponente und einer Kontextverteilinfrastruktur besteht. Die einzelnen Bereiche besitzen unterschiedliche Charakteristiken, was sich in unterschiedlichen Kommunikationsanforderungen widerspiegelt. Hierfür wurde eine Kommunikationsmiddleware vorgeschlagen, welche die notwendigen Funktionalitäten leistet. Abbildung 7.22 zeigt eine Erweiterung des in Abbildung 7.2 gezeigten Funktionsschichtbildes.

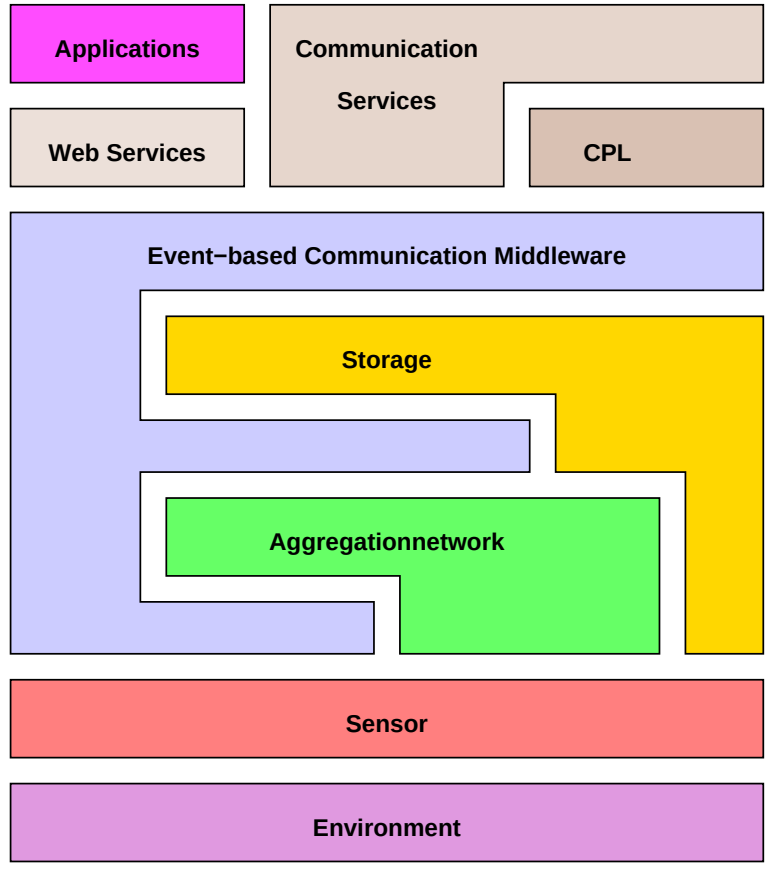


Abbildung 7.22: Schichtbild mit Kommunikationsmiddleware

Eine ereignisbasierte Middleware wurde für die Verteilung der Sensordaten, Kontextmerkmale und Kontexte vorgeschlagen. Diese erlaubt eine skalierbare und asynchrone

Kommunikation zwischen den beteiligten Entitäten. Innerhalb des Aggregationsnetzwerks wird eine sehr schnelle und effektive Push-basierte Kommunikation verwendet. Applikationen können sowohl über den Event-Mechanismus, aber auch über eine Web Service Schnittstelle unter der Verwendung von SOAP die Kontexte erhalten. Für die Verteilung zwischen zwei Applikation wurde ein In-band Mechanismus unter der Verwendung des Session Initiation Protocols vorgeschlagen. Kontextobjekte können in dem entwickelten PIDF-CE-Format repräsentiert und verteilt werden.

Kapitel 8

Vereinfachte Diensterstellung für Nutzer

> Der Mensch ist ein auf vielen Ebenen kommunizierendes Wesen, das manchmal auch spricht.
>
> Ray L. Birdwhistell

In Kapitel 3 wurden Verfahren zu einer effizienteren Kommunikation vorgestellt. Schlüsselkomponenten sind dabei die Einbeziehung von Kontexten in Kommunikationsdienste und die Verlagerung der Aufwände für den Rufaufbau vom Nutzer auf das System. In den nachfolgenden Kapiteln 4 und 5 wurde eine Kontextmodellierung entwickelt sowie ein konzeptuelles Framework entworfen, welches die Erstellung und den Betrieb von Kontext-bewussten Diensten unterstützt.

Dieses Kapitel beschreibt Verfahren und Konzepte zur Vereinfachung der Erstellung Kontext-bewusster Dienste für ambitionierte Nutzer. Dazu wurde eine regel- und skriptbasierte Erstellungsmethodik als eine geeignete Form identifiziert, um einen breiten Nutzerkreis zu erschließen. Die im SIP-Umfeld etablierte Call Processing Language (CPL) wurde als Ausgangspunkt einer Skriptsprache gewählt und bewertet. Die Limitierungen der Sprache insbesondere im Hinblick auf die Erstellung von Kontext-bewussten Kommunikationsdiensten wurden durch eine Erweiterung der Sprachsyntax aufgehoben. Ein graphischer Editor zur Erstellung erweiterter CPL-Skripte wird als Werkzeug zur Vereinfachung des Erstellungsprozesses sowie zur Reduktion der Fehlermöglichkeiten bei der Skripterzeugung vorgestellt.

Die Verlagerung der Aufwände für einen erfolgreichen Kommunikationsaufbau wird durch das vorgeschlagene Konzept des Kommunikationsbrokers ermöglicht. Die notwendigen Algorithmen sowie die dazugehörigen Anwendungen zum Auffinden von geeigneten Zeitspannen für eine Kommunikation werden entworfen und umgesetzt.

Kurzübersicht

Die Struktur des Kapitels ist folgendermaßen: Eine allgemeine Betrachtung der Methodik der Skript- und Regel-basierten Diensterstellung ist in Abschnitt 8.1 gegeben. Aus der Analyse leitet sich die Verwendung und Bewertung von CPL als Skriptsprache ab. Die Einschränkungen von CPL hinsichtlich der Kontrollstrukturen sowie das Fehlen von Komponenten zur Unterstützung der Erstellung Kontext-bewusster Dienste führt zur Erweiterung von CPL, welche in Abschnitt 8.2 dargestellt ist. Ein graphischer Editor zur Erstellung von erweiterten CPL-Skripten ist in Abschnitt 8.3 vorgestellt.

Eine weitere Form der Erstellung von nutzerzentrierten Diensten wird mit einer regelbasierten Kommunikationsanwendung vorgeschlagen. Eine Ausprägung einer regelbasierten Anwendung ist der Digital Call Assistant, der in Unterabschnitt 8.4.1 vorgestellt wird. Das Konzept des Digital Call Assistant beschreibt die Verlagerung der Aufwände für den Verbindungsaufbau vom Nutzer auf das System. Als konkretes System werden Kommunikationsbroker vorgestellt, welche unter Einbeziehung von Kontexten für beide Seiten günstige Zeitschlitze zur Kommunikation aushandeln. Die Struktur des Kapitels ist in Abbildung 8.1 dargestellt.

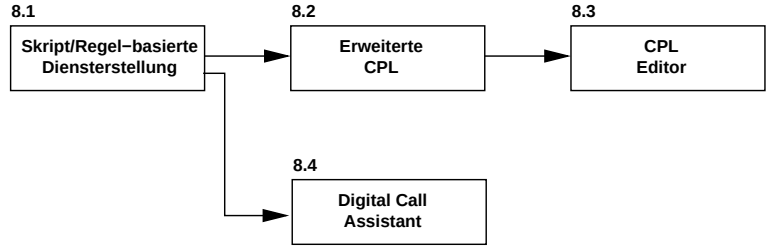


Abbildung 8.1: Struktur des Kapitels 8

8.1 Regel- und Skriptbasierte Diensterstellung

Die Erzeugung von nutzerzentrierten Diensten soll, wie in Abschnitt 3.2 gefordert, möglichst einfach zu handhaben sein. Eigene Dienste durch Modifikation existierender Anrufsteuerdienste zu realisieren ist jedoch möglich und wird in Unterabschnitt 9.5.1 für eine Kontext-bewusste Anrufumleitung gezeigt. Dennoch entsteht hierbei ein nicht unerheblicher Aufwand und ein tiefergehendes Wissen über Kommunikationsdienste und Softwaredesign ist unerlässlich. Eine vereinfachte Realisierung von nutzerzentrierten Diensten kann dem Nutzer durch Bereitstellen einer *regelbasierten* Dienstentwicklungsumgebung zur Verfügung gestellt werden.

Eine regelbasierte Beschreibung von Kommunikationsaufgaben stellt eine formale und technische Realisierung der Spezifikation von Wünschen an ein Sekretariat (siehe auch Abschnitt 1.1) dar. Der Nutzer spezifiziert hierbei die von ihm gewünschte Handhabung von eingehenden Kommunikationsanfragen in Form von *Richtlinien* (*policies*) oder

konkreter als *Regeln* (*rules*). Für ein Telefonie-System können diese Regeln mit einer Skriptsprache erfasst und in ausführbare Anweisungen übersetzt werden. Zur weiteren Vereinfachung sowie zur Fehlervermeidung kann ein (graphischer) Editor den Nutzer bei der Erstellung der Regeln unterstützen, wie dies mit Erfolg bei der Erstellung von Filterregeln in aktuellen E-Mail-Programmen angewendet wird.

Für die Realisierung von Diensten, die der Nutzer selbstständig entwerfen und aktivieren kann, wird das Konzept der E-Mail-Filterregelerstellung auf Telefoniedienste umgesetzt und erweitert. Formal wird das Konzept der regelbasierten Kommunikationsdiensterstellung als *Event-Condition-Action* (ECA)-Prozess aufgefasst. Das ECA-Framework ist in der Domäne der Aktive Datenbanken (*active database*) [WC95, Pat99, BPW02] ein gut erforschtes und eingesetztes Konzept. Eine ECA-Regel r wird geschrieben als

$$r\colon \{E\}\colon C \rightarrow A \tag{8.1}$$

wobei $\{E\}$ für die Menge an *Ereignissen* (*events*), die die Regel auslösen steht. C bezeichnet die *Bedingung* (*condition*) der Regel und A benennt die Aktionen (*action*), die ausgelöst werden.

Tritt ein Ereignis oder eine Kombination von Ereignissen ein, so wird die Regel aktiviert. Anschließend wird die Bedingungsaussage ausgewertet. Ist die Aussage erfüllt, so wird die spezifizierte Aktion im Action-Teil der Regel ausgeführt. Eine Reihe von Operatoren kann verwendet werden, um eine Kombination von Ereignissen zu beschreiben und um komplexere Ausdrücke zu erstellen [GJS92, CM93, GJS94]. Eine Disjunktion (*disjunction*) $(e_1 \vee e_2)$ ist wahr, wenn eine Ereignis von beiden oder beide Ereignisse eintreffen. Eine Konjunktion (*conjunction*) $(e_1 \wedge e_2)$ wird erfüllt, wenn beide Ereignisse innerhalb eines spezifizierten Zeitintervalls eintreten. Eine Ordnung des Eintretens wird nicht ausgewertet. Die Sequenz (*sequence*) $(e_1; e_2)$ liefert ein wahres Ergebnis, wenn die Ereignisse in der spezifizierten Reihenfolge eintreten (also e_1 vor e_2). Eine Negation (*negation*) (not e_1 [*interval*]) ist wahr, wenn das Ereignis nicht innerhalb einer definierten Zeitspanne auftritt.

Der Nutzer kann mit dem vorgeschlagenen Regelsystem die Ausführung von Diensten kontrollieren. Ein Beispiel einer Regel (hier in Pseudonotation) ist nachfolgend angegeben:

$$\begin{aligned} &\texttt{on}\ \textit{incoming call}\quad \texttt{if}\ \textit{context} = \textit{meeting} \\ &\texttt{do}\ \textit{CFNR}(\textit{addr} = \textit{secretary})\quad \texttt{else}\ \textit{VM} \end{aligned} \tag{8.2}$$

In Abhängigkeit vom gegenwärtigen Kontext eines Nutzers wird ein eingehender Anruf, wenn er nach einer spezifizierten Zeitspanne nicht beantwortet wird, entweder an das Sekretariat weitergeleitet (*Call Forwarding on No-Reply (CFNR)*) oder alternativ an einen Anrufbeantworter umgeleitet (*Voice Mail (VM)*).

Darüber hinaus kann die Verwendung einer Event-Condition-Action (ECA)-Notation zur Vermeidung von unerwünschten lokalen Dienstwechselwirkungen durch Mehrdeutigkeit (*ambiguity*) eingesetzt werden, wie dies in [GAMS03] gezeigt ist. Mehrere Dienste können dabei gleichzeitig aktiviert sein. Die Entscheidung, welcher Dienst ausgeführt

wird, wird anhand der Bedingung C getroffen. Dabei können Konfliktlösungsstrategien bereits vorab (*off-line*) auf die Regeln angewandt werden [JMM99, Han92].

8.1.1 Verwendung von CPL als Diensterstellungsskriptsprache

Die Skriptsprache Call Processing Language (CPL) wurde im Unterabschnitt 2.3.2.1 einführend beschrieben. Die Funktionalitäten der Call Processing Language ermöglichen bereits eine sehr eingeschränkte Umsetzung von ECA-Regeln. Diese werden insbesondere im Bereich der Filterung von Kommunikation im Umfeld des Session Initiation Protocols (SIP) verwendet. CPL wurde auf Grund ihrer weiten Verbreitung, der Verfügbarkeit von CPL-Servern sowie der Erweiterbarkeit ihrer Syntax als Basis für eine regelbasierte Erstellung von Kontext-bewussten Kommunikationsdiensten ausgewählt. Notwendige Erweiterungen der Sprachsyntax und die Umsetzung in einem Server für CPL-Skripte wird in Abschnitt 8.2 beschrieben. Ein graphischer Editor zur Unterstützung des Nutzers bei der Erstellung der Skripte ist in Abschnitt 8.3 ausgeführt.

Die Sprachsyntax von CPL erlaubt das Erstellen von Kontrollanweisungen für die Behandlung von eingehenden und ausgehenden Anrufen. Dies sind auch die beiden einzigen definierten Ereignisse, die ein CPL-Skript auslösen kann. Bedingungen werden mittels Kontrollstrukturen sogenannte Verzweigungen (*switches*), ausgewertet. Die Bedingungen sind Kombinationen aus Ausdrücken, die die Zeit, Adressinformationen oder Prioritäten auswerten. Die Anordnung der Bedingungen entspricht dabei einer disjunktiven Normalform (*Disjunctive Normal Form – DNF*):

$$(c_{1,1} \wedge c_{1,2} \ldots \wedge c_{1,n_1}) \vee (c_{2,1} \wedge c_{2,2} \ldots \wedge c_{2,n_2}) \vee \ldots \vee (c_{m,1} \wedge c_{m,2} \ldots \wedge c_{m,n_m}) \quad (8.3)$$

wobei $c_{i,j}$ einen Einzelausdruck darstellt, der entweder `WAHR` (*true*) oder `FALSCH` (*false*) ist.

Ausgeführt wird die Aktion, die am Ende des Pfades des Graphens steht, der das CPL-Skript repräsentiert. Hierbei handelt es sich um Rufsteuerdienste, die das aktuelle Gespräch umleiten oder terminieren können. Es können bewusst keine externen Programme aufgerufen werden. Anschließend terminiert das CPL-Skript. Die Konzeption der gewollten Einschränkung der Funktionalität prädestiniert die Call Processing Language für die Erstellung von sicheren Diensten, die auf einem Server ausgeführt werden und aktiv keine Anrufe aufbauen können.

8.1.1.1 Einschränkungen von CPL

Die intendierte Absicherung der Dienste gegen einen übermäßigen Ressourcenverbrauch und den Aufruf externer Programme bringt auch eine Einschränkung der Sprache mit sich. Eine weitere Limitierung der Einsatzmöglichkeiten von CPL ist durch die Aktivierung des Skriptes ausschließlich durch ein- und ausgehende Nachrichten bedingt. Bei einer Erweiterung der Sprachsyntax der CPL sollen die Entwurfsprinzipien der Sprache erhalten bleiben. Es muss daher zwischen der Sicherheit bei der Ausführung der Dienste und der Freiheit bei der Gestaltung der Dienste abgewogen werden.

Weiterhin ist im aktuellen CPL-Standard nur eine kleine Anzahl an Elementen zur Anrufsteuerung definiert. Diese erfüllen nicht die Kriterien, die zur Erstellung von Kontext-bewussten Kommunikationsdiensten notwendig sind. So kann weder der aktuelle Kontext abgefragt werden, noch kann eine Bewertung eines Kontexts als Verzweigung Verwendung finden.

Es wurden einige Erweiterungen für die CPL-Syntax bzw. andere CPL-ähnliche Skriptsprachen vorgeschlagen, welche die Einschränkungen von CPL aufheben wollen. Eine der Call Processing Language ähnliche Sprache ist die Service Control Markup Language (SCML) [BJ02], die auf das IN-Rufmodell (*IN call model*) abgestimmt ist und in-call-Aktionen auslösen sowie Rufe aufbauen kann. Die Sprache Language for Endsystem Services (LESS) [WS03, WS05] wurde speziell für den Entwurf und die Bereitstellung von Diensten in Endsystemen entwickelt. Eine weite Verbreitung konnten aber beide Ansätze nicht erzielen.

Eine alternative Anwendungsdomäne wird von der Sprache VoiceXML [MBC+04] adressiert. Diese Sprache wurde für die Erzeugung von Menüs, in denen mittels Sprachdialogen navigiert werden kann, entworfen. Einen ähnlichen Ansatz verfolgt die Call Control Extensible Markup Language (CCXML) [Aub05] mittels derer Call-Events kontrolliert werden können. Die Sprache verfügt im Gegensatz zu CPL über Variablen und Schleifen, sowie einen `goto`-Operator, was die Sprache Turing-vollständig macht [Len04], aber auch anfälliger für unvorhergesehene Aktionen [Dij68].

8.2 Erweiterung der Call Processing Language

Die Call Processing Language wurde als Skriptsprache zur Erzeugung von Kontrollmöglichkeiten eingehender und ausgehender Anrufe in Unterabschnitt 2.3.2.1 vorgestellt. Die wesentlichen Einschränkungen der Sprache sind das (erwünschte) Fehlen der Möglichkeit, Anrufe selbsttätig auszulösen sowie der limitierte Satz an Verzweigungen und Aktionen. Darüber hinaus sind die Bedingungen der erzeugten Skripte statisch und können sich nicht an die sich dynamisch ändernden Kontexte des Nutzers anpassen.

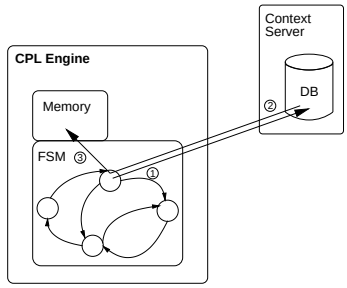
Dennoch bietet CPL eine geeignete Ausgangsbasis, um Dienste zu erstellen, die eingehende Anrufe filtern können, wie dies im Szenario in Unterabschnitt 3.3.1 vorgestellt wurde. Die in dieser Arbeit erweiterten Funktionalitäten ermöglichen eine Erstellung von umfassenden Kontext-bewussten Diensten. Hierzu wird die generelle Abfrage von nutzerbezogenen aktuellen Kontextinformationen bereitgestellt. Methoden für das Kontext-bewusste Filtern von eingehenden und ausgehenden Anrufen wurden erstellt. Zusätzlich wird das Abfragen von Kontextinformationen durch Anrufer unterstützt, wie dies für Szenarien wie in Unterabschnitt 3.3.2 beschrieben benötigt wird. Hierfür wurde die Sprachsyntax von CPL erweitert und die Verarbeitungseinheit *CPL Engine* entsprechend angepasst.

8.2.1 Sprachsyntax

Der Sprachumfang von CPL wird durch eine im Standard [LS00a, LWS04] spezifizierte Document Type Definintion (DTD) beschrieben. Mittlerweile existiert eine entsprechende Spezifikation in XML Schema. Die Erweiterungen der Sprachsyntax werden konkret an der CPL-DTD, die von Vovida für die Vovida Open Communication Application Library (VOCAL) bereitgestellt wird, durchgeführt. Hierzu wurden neue Kontrollstrukturelemente für die Sprachsyntax definiert, die die Erstellung von Kontextbewussten Kommunikationsdiensten ermöglichen. Die Elemente sind zum Nachweis der Realisierbarkeit für die VOCAL-CPL-Engine und das CPL-Modul des SER-Servers umgesetzt worden. Gegenwärtig sind dies die nachfolgend beschriebenen Elemente *Context Lookup*, *Context Switch*, *Context Notifier* und *Answer Switch*.

Context Lookup

Die *Context Lookup* Komponente stellt die logische Verbindung zwischen dem CPL-Skript und einer zentralen Kontextquelle her. Eine solche zentrale Quelle ist beispielsweise der Context Server, wie er in Abschnitt 9.4 beschrieben ist. In der technischen Umsetzung stellt diese Komponente eine Verbindung von der CPL-Engine via eines MySQL-Connectors zu einer Datenbank oder mittels einer SOAP-Kommunikation zu einem Web Service her. Die Komponente kann dazu entsprechend parametrisiert werden, wie dies in Abbildung 8.2 gezeigt ist. Der aktuelle Kontext des Nutzers liegt dann dem Skript als String vor.



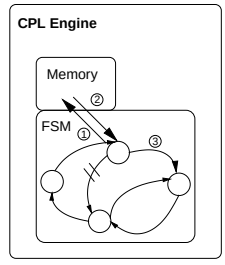

```
<context-lookup db="context"
                user="mgoertz">
   <store />
</context-lookup>
```

Abbildung 8.2: CPL Context Lookup Komponente mit CPL-Skript. Das Skript fragt den aktuellen Kontext vom ContextServer ab und stellt ihn dem System zur Verfügung.

Bei der internen Verarbeitung der Anfragen werden jeweils die passenden Structured Query Language (SQL)-Ausdrücke bzw. Methodenaufrufe für den Web Service erzeugt und ausgeführt. Da es sich um eine Kommunikation mit einer potenziell entfernten Entität handelt, mussten Timer eingeführt werden. Dieses Konzept ist neu in der CPL-Engine. Wenn innerhalb einer definierten Zeitspanne keine Antwort von der externen Kontextquelle erfolgt, wird ein Standard-Wert weitergeliefert. Nachfolgende Komponenten können diesen Wert zur Auswahl der Standard-Verzweigung wählen. Da es sich um einen Echtzeitdialog handelt, muss im Zweifelsfall das Warten auf das richtige Ergebnis dem zeitnahen Weiterleiten des Anrufs untergeordnet werden.

Context Switch

Der *Context Switch* ist eine bedingte Verzweigungskomponente, die den aktuellen Kontext des Nutzers als Bedingung für die Verzweigung bewertet. Der Nutzer kann festlegen, welchem weiteren Pfad die Kommunikationssteuerung folgen soll. So kann die Verarbeitung von Anrufen gewählt werden, die im jeweiligen Kontext geeignet ist bzw. vom Nutzer als geeignet spezifiziert worden ist. Insbesondere in der Kombination mit anderen verfügbaren Verzweigungskomponenten ergibt sich ein sehr mächtiges und nützliches Verfahren zur effektiven Steuerung der Kommunikation. Den aktuellen Kontext bezieht die Context Switch Komponente aus dem Speicher, in den die Context Lookup Komponente den Kontext vorher hineingeschrieben hat. Eine schematische Abbildung der Funktionalität der Context Switch Komponenten sowie ein Beispiel-CPL-Skript sind in Abbildung 8.3 dargestellt.




```
<context-switch field="label">
   <context contains="meeting">
      <reject status="reject" />
   </context>
   <context is="travel">
      <location url="sip:rac@kom.eu">
         <proxy />
      </location>
   </context>
   <otherwise>
      <sub ref="voicemail" />
   </otherwise>
</context-switch>
```

Abbildung 8.3: CPL Context Switch Komponente mit CPL-Skript-Beispiel. Das Skript verzweigt in Abhängigkeit des aktuellen Kontexts zu verschiedenen Aktionen

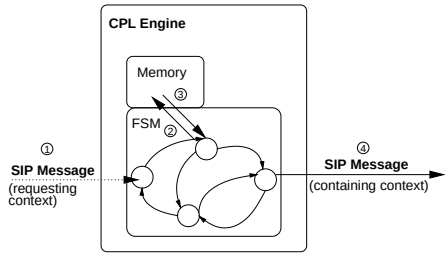
Die Context Switch Komponente kann zwischen 1 und n Verzweigungen haben. Jede dieser Verzweigungen bewertet den zurückgelieferten Kontext-Bezeichner λ. Dabei kann λ auf einen Teilstring (`contains`) oder auf die vollständige Übereinstimmung (`is`) geprüft werden. Zusätzlich kann mit dem `not`-Operator auf Nichtvorhandensein überprüft werden. Bei der ersten Übereinstimmung wird der entsprechenden Verzweigung gefolgt und die weitere Überprüfung wird abgebrochen. Sollten nach der ersten Übereinstimmung weitere Verzweigungsbedingungen vorliegen, die ebenfalls zu einer Übereinstimmung geführt hätten, so bleiben diese unberücksichtigt. Dieses Verhalten reflektiert auch, dass ein offener Gesprächsaufbau nicht wie eine Programmausführung in einer Schleife wiederholt werden kann, so dass alle Aktionen, die auf eine Überstimmung der Bedingung folgen, ausgeführt werden.

Context Notifier

Die *Context Notifier* Komponente beantwortet Anfragen von Anrufern nach dem gegenwärtigen Kontext des Nutzers. Dies wird technisch durch eine SIP NOTIFY-Nachricht realisiert. Der Nachrichtenkörper dieser Nachricht enthält den aktuellen Kontext des Nutzers im PIDF-CE-Format. Der Context Notifier ist zustandsbehaftet und merkt sich

die offenen Anrufanfragen für einen definierten Zeitraum. Um die Zustände zu halten, wird die Session-ID der INVITE-Nachricht gespeichert. Dies ermöglicht, weitere Anrufanfragen der ersten Anfrage zuzuordnen und danach geeignet weiter zu verarbeiten.

Die Abfrage des Kontexts geschieht ohne Interaktion des Nutzers und damit ohne potenzielle Störung. Die Mitteilung des Kontexts kann durch den Parameter `detail` und eine vorhergehende Authentifizierung (`auth`) kontrolliert werden. Die Context Notifier Komponente ist in Abbildung 8.4 mit einem exemplarischen Ausschnitt eines CPL-Skripts dargestellt.




```
<context-lookup db="context"
                user="mgoertz">
   <location url="sip:je@tum.de">
      <notify detail="3" auth="yes" />
   </location>
</context-lookup>
```

Abbildung 8.4: CPL Context Notifier Komponente mit CPL-Skript-Beispiel. Das Skript sendet den aktuellen Kontext an denjenigen, der diesen abgefragt hat.

Die Detailstufen, die sich mittels des Parameters `Detail-Level` wählen lassen, entsprechen den Hierarchiestufen, die in Unterabschnitt 7.5.2 für die Kontextrepräsentation PIDF-CE eingeführt wurden. Durch eine vorgeschaltete Adress-Verzweigungskomponente, die auf Nutzer- oder Domain-Adressen des Anrufers filtert, kann innerhalb dieser Granularität der jeweilige gewünschte Detailgrad eingestellt werden.

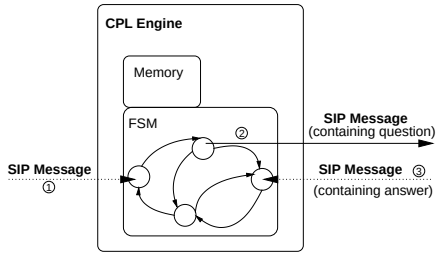
Bei der Authentifizierung werden die jeweiligen im SIP-Standard vorgeschlagenen Verfahren berücksichtigt. Mittels einer HTTP-auth Authentisierung können der anrufende Nutzer authentifiziert und der entsprechende Kontextdetailgrad mitgeteilt werden. Eine weitere kryptographische Sicherheit wird durch die Aktivierung der Verschlüsselung des Inhalts der NOTIFY-Nachricht erreicht. Die Verschlüsselung setzt auf S/MIME auf.

Bei der ersten Kommunikation mit einem Gesprächspartner ist ein Schlüsselaustausch zur Initialisierung der Authentifizierung und der Verschlüsselung notwendig. Die Schlüssel werden gespeichert und bei einer erneuten Kommunikation wird nur noch die Gültigkeit des Schlüssels geprüft. Dieses Vorgehen reduziert den Zeitaufwand für die sichere Kommunikation. Die Verwendung der kryptographischen Verfahren setzt die Existenz einer Public Key Infrastructure (PKI) voraus, welcher beide Parteien vertrauen.

Answer Switch

Mit der *Answer Switch* Komponente wird die Möglichkeit geschaffen, dem Anrufer mittels Rückfragen eine gezielte Entscheidung über die Verarbeitung seines Anrufes

zu überlassen. Dazu wird von der CPL-Engine eine SIP-Nachricht an den Anrufer geschickt. Die Frage wird auf dem Endgerät angezeigt und der Nutzer bekommt dann eine vordefinierte Auswahl an Antworten präsentiert. In der Regel sind dies 'ja/nein'Fragen, die beantwortet werden müssen. Die gewählte Antwort wird anschließend wieder zurück an den Server geschickt und dort weiterverarbeitet. So kann der Angerufene mitteilen, unter welchen Umständen er für eine Konversation zur Verfügung steht. Die Answer Switch Komponente ist zusammen mit der CPL-Syntax in Abbildung 8.5 dargestellt.




```
<answer-switch question="'urgent?"'>
   <answer is="yes">
      <proxy />
   </answer>
   <otherwise>
      <reject status="reject" />
    </otherwise>
</answer-switch>
```

Abbildung 8.5: CPL Answer Switch Komponente mit CPL-Skript-Beispiel. Wird die Frage mit 'yes' beantwortet, so fährt das Skript fort, andernfalls wird der Verbindungsaufbau abgebrochen.

8.3 Editor zur Nutzerunterstützung bei der Bearbeitung von CPL-Skripten

Die Erstellung von Regeln zur Beschreibung der Steuerung von Kommunikationsflüssen oder die Spezifikation von Kommunikationsdiensten erfordert die Einarbeitung und die Anwendung einer Beschreibungssprache. Im Rahmen dieser Arbeit wurde die Call Processing Language als eine Sprache identifiziert, mit der es möglich ist, solche Regeln zu erstellen. Darüber hinaus erhöht die manuelle Erstellung von CPL-Skripten die Fehlerwahrscheinlichkeit und damit auch die Zeit, die zur Fehlersuche aufgewandt werden muss. Eine geeignete graphische Unterstützung im Erstellungsprozess ist ein wichtiges Kriterium dafür, ob die Verwendung von CPL-Skripten für Kontext-bewusste Dienste vom Nutzer angenommen wird.

In dieser Arbeit wurde ein vorhandener *CPL-Editor* als graphisches Frontend zur Erzeugung und Modifikation von CPL-Skripten weiterentwickelt. Der Basiseditor wurde im Projekt X-ING an der HU-Berlin in Kooperation mit der Siemens AG im Bereich Information and Communication Networks – COM (vormals ICN) erstellt [www21]. Der Editor unterstützt den Umfang der CPL-Spezifikation Version 0.4. Implementiert wurde der Editor in der Sprache JAVA.

8.3.1 Funktionsumfang des Basis-CPL-Editors

Der Editor erlaubt das Öffnen bestehender CPL-Skripte und deren Modifikation, sowie die Erstellung von neuen Skripten. Die erstellten Skripte können entweder als Datei abgespeichert werden oder per Lightweight Directory Access Protocol (LDAP) direkt an einen Server übertragen werden, der CPL-Skripte verwaltet. Es kann immer nur ein CPL-Skript auf einmal angezeigt und bearbeitet werden. Bei der Erstellung eines neuen CPL-Skriptes erhält der Nutzer eine leere Arbeitsfläche, auf der bereits der Wurzelknoten des Graphens eintragen ist.

8.3.2 Aufbau des Basis-CPL-Editors

Zu den Stärken des Designs des Editors zählt vor allem seine weitgehende Steuerung durch eine Konfigurationsdatei. In der Konfigurationsdatei sind alle Kontrollstrukturen definiert. Die Klassen im Editor sind generisch gehalten und passen ihre graphische Repräsentation und Funktion der Definition des Skriptes an. Die Konfiguration legt den Typ, das Aussehen, die Anzahl der Ausgänge und die möglichen Parameter fest. Zusätzlich ist festgelegt, wie die Parameter des Elements in einer äquivalenten CPL-Notation beschrieben werden. Eine Konfiguration für den hinzugefügten *Context-Switch* ist in Listing E.1 abgebildet.

Für die Handhabung der XML-Dokumente wird das Castor XML Paket [www31] genutzt. Dieses bietet ein Databinding-Framework, das im Unterschied zu Document Object Model (DOM) oder Simple API for XML (SAX) Methoden bereitstellt, um XML-Dokumente als Datenobjekte zu repräsentieren und zu manipulieren. Ein XML-Schema kann verwendet werden, um die syntaktische Gültigkeit des erstellten Skriptes zu validieren.

8.3.3 Erweiterung des CPL-Editors

Der Basis-Editor wird um die Unterstützung der in Abschnitt 8.2 beschriebenen Erweiterungen der CPL-Syntax erweitert. Die neuen Kontrollstrukturen sind *Context Lookup*, *Context Notify*, *Answer Switch* und *Context Switch*. Die Parametrisierungsmöglichkeiten der neuen Kontrollstrukturen sind nachfolgend beschrieben. Definiert wurden die Elemente und ihre Notation in einem CPL-Skript durch Erweiterung der Basis-Konfigurationsdatei des Editors. Zusätzlich werden neue Symbole (*icons*) für die Boxen bereitgestellt.

Ein Screenshot des erweiterten CPL-Editors mit einem Beispielskript, das die neuen CPL-Elemente enthält, ist in Abbildung 8.6 dargestellt. Die dunklen rechteckigen Kästen sind *Parametrisierungsmenüs* der einzelnen Boxen. Diese Boxen sind Icons, die CPL-Elemente wie *Actions* und *Switches* darstellen. Die Verwendung des CPL-Editors mit den Erweiterungen erleichtert die Erstellung von Kontext-bewussten CPL-basierten Kommunikationsdiensten signifikant. Insbesondere die graphische Darstellung

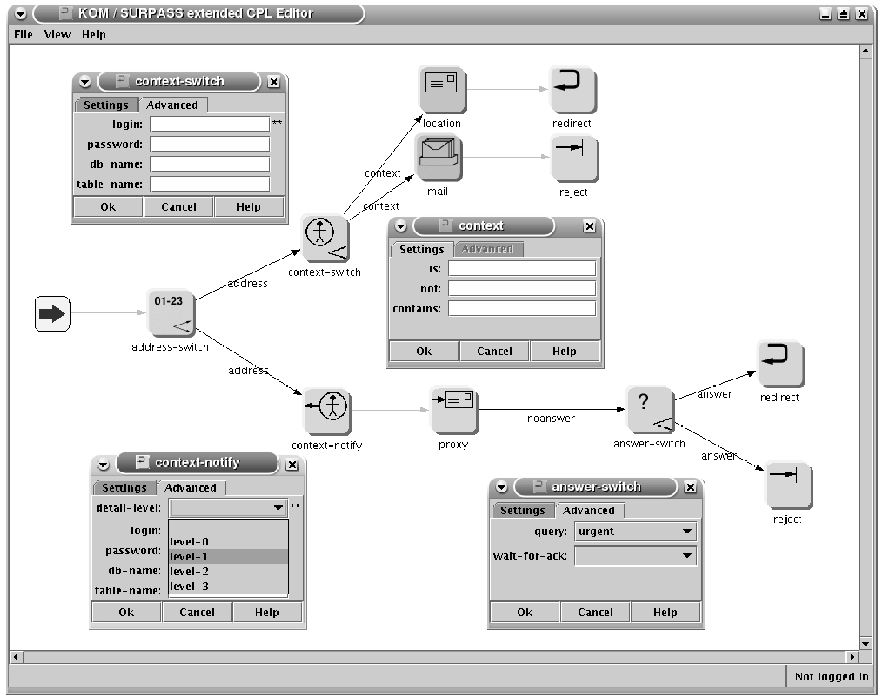


Abbildung 8.6: Bildschirmfoto des erweiterten CPL-Editors

der Skripte erlaubt die rasche Erfassung von deren Struktur und deren Funktionsweise.

8.4 Verwendung von regelbasierter Kommunikation mit Kontextinformationen zur Effizienzsteigerung

Eine Analyse von täglichen Kommunikationsabläufen führt zu der Beobachtung, dass es bei dem Versuch den Gesprächspartner zu erreichen, oftmals zu einer Schleife von nicht-erfolgreichen Anrufversuchen kommen kann. Listen aller entgangenen Anrufe, wie sie von modernen Telefonen zur Verfügung gestellt werden, können zu einer Verschiebung der Verantwortlichkeit für den nächsten Anrufversuch führen. Der initiierende Anrufer überlässt dabei oftmals den nächsten Verbindungsversuch dem Angerufenen. Ein Rückruf des Angerufenen kann wiederum zu einem Eintrag in der Liste des ursprünglichen Anrufers führen. Eine Möglichkeit, solche erfolglosen Anrufe zu minimieren, soll das Konzept *Digital Call Assistant* bieten.

8.4.1 Digital Call Assistant

Das Grundprinzip des Digital Call Assistant [GAS04b] ist das Aushandeln und Anzeigen von freien Zeitschlitzen (*time slots*), in denen eine Kommunikation für beide Teil-

nehmer geeignet ist. Dabei werden die Zeitschlitze derart ausgewählt, dass kompatible Kommunikationskanäle verfügbar sind. Die Aushandlung basiert auf dem beiderseitigen willentlichen Einverständnis, die notwendigen sensitiven Informationen auszutauschen. Da Tagesabläufe und damit auch die Verfügbarkeit bezüglich der Kommunikationsbereitschaft dynamischer Natur sind, ist es zwingend notwendig, den aktuellen Kontext der Teilnehmer in die Planung mit einzubeziehen.

Die Anwendung macht ausführlichen Gebrauch von Kalendereinträgen, die in elektronischen Kalendern gespeichert sind. Es wird davon ausgegangen, dass Nutzer des Digital Call Assistant ihre täglichen Termine und Erinnerungen mittels Kalendern verwalten. Dabei besitzen Angestellte von Firmen in der Regel einen unternehmensweiten *gemeinsamen* Kalender und einen eigenen *privaten* Kalender. Jeder Kalendereintrag besteht aus einer Start- und Endzeit, einer deskriptiven Beschreibung sowie einer Kategorisierung und gegebenenfalls einer Lokationsbeschreibung.

Die Kalendereinträge sind oftmals in einem herstellereigenen Format abgelegt, so dass ein Austausch dieser Daten nicht problemlos möglich ist. Für den interoperablen Austausch von Kalendereinträgen findet das Format des Internet Calendaring (iCal) [DS98] zunehmend Verbreitung. Neben dem Format zur Repräsentation der Kalendereinträge wurden auch Protokolle für den Austausch über verschiedene Protokolle, wie SMTP oder HTTP definiert. Das iCal-Format wurde aufgrund der weiten Verbreitung sowie der Erweiterbarkeit als Basis für die Digital Call Assistant Applikation gewählt.

Die Planungsapplikation des Digital Call Assistant ist in die Infrastruktur der Kontextbewussten Kommunikationsdienste eingebunden. Die Einordnung in das Gesamtsystem ist in Abbildung 8.7 gezeigt. Zur Instanzierung der Kommunikation wird eine 3rd-Party Anrufsteuerungskomponente, wie z. B. der SIP Back-2-Back User Agent angesteuert. Die einzelnen Schritte des Algorithmus zur Bestimmung von geeigneten Zeitschlitzen für eine Kommunikation sind nachfolgend beschrieben.

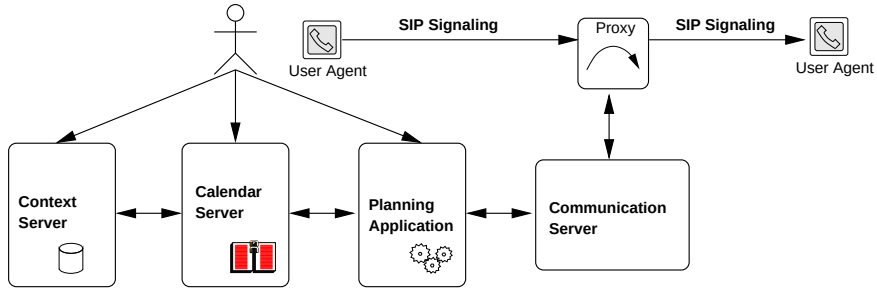


Abbildung 8.7: Digital Call Assistant im Gesamtsystem

8.4.1.1 Phasen des Digital Call Assistant-Algorithmus

Der Algorithmus, der vom Digital Call Assistant genutzt wird, um die Zeitpunkte für die Kommunikation zu bestimmen, ist in die Phasen *Vorbereitung*, *Initialisierung*, *Kommunikationsanfrage*, *Abgleich*, *Aktualisierung* und *Rufaufbau* unterteilt. Diese werden

nachfolgend im Detail beschrieben. Ein Ablaufdiagramm der einzelnen Phasen des statischen Falles ist in Abbildung 8.8 gezeigt. Die Anwendung nutzt die Funktionalitäten einer iCal-kompatiblen Kalenderapplikation. Eine Erweiterung des Algorithmuses ist in Unterabschnitt 8.4.1.2 beschrieben.

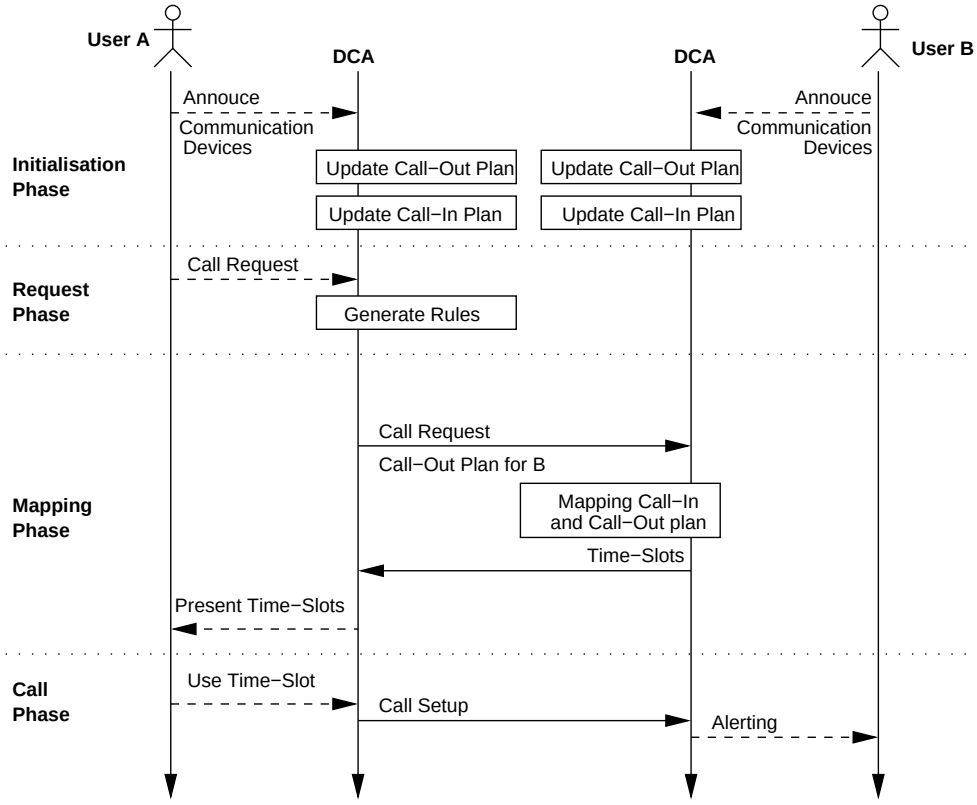


Abbildung 8.8: Ablaufsequenzen des Algorithmus des Digital Call Assistant

Vorbereitungsphase In der Vorbereitungsphase werden eine *Kompatibilitätsliste* aller verfügbaren Gerätetypen sowie die *Nutzerpräferenzen* definiert. Der Ablauf ist schematisch in Abbildung 8.9 gezeigt. Diese Schritte müssen nicht vom Nutzer erledigt werden, sondern können beispielsweise unternehmensweit vom Administrator durchgeführt werden. Zusätzlich wird ein Satz an vordefinierten hochwertigen Kontexten wie *Besprechung*, *Arbeitsplatz* oder *Mensa* angelegt.

In der Kompatibilitätsliste werden diejenigen Ressourcen wie Kommunikationsgeräte und -typen einander zugeordnet, die miteinander kommunizieren können. So ist ein Endgerättyp A fähig, mit einer Reihe anderer Gerätetypen zu kommunizieren, dargestellt durch

$$A \multimap \{A, C, D\} \tag{8.4}$$

Die Kommunikation kann direkt (z. B. Mobiltelefon $\rightleftarrows$ ISDN-Telefon) oder mittels eines Gateways (z. B. Voicemail $\rightarrow$ E-Mail oder Short Message Service (SMS)) erfolgen.

In den Nutzerpräferenzen wird eine Zuordnung der bevorzugten Kommunikationskanäle zu den verschiedenen definierten Kontexten abgelegt. Dabei kann für jeden Kommunikationstyp eine Priorität festgelegt werden.

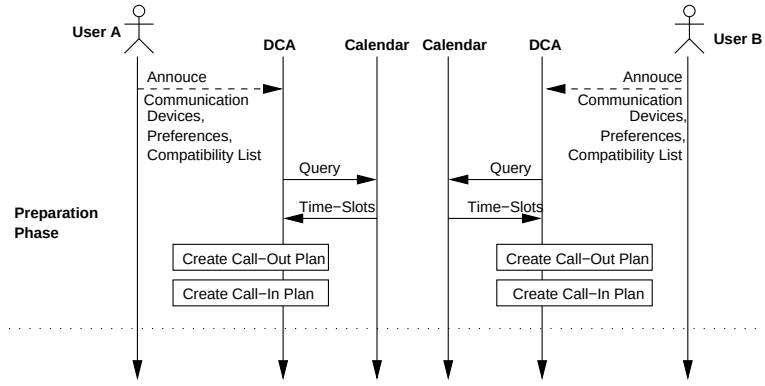


Abbildung 8.9: Vorbereitungsphase des Digital Call Assistant

Jeder Nutzer des Digital Call Assistant kontrolliert zwei Kommunikationslisten. Eine Liste, der *Call-Out Plan*, beschreibt, welche Kommunikationsressourcen zu welcher Zeit voraussichtlich für ausgehende Kommunikationswünsche zur Verfügung stehen. Dieser Prozess lässt sich weitestgehend durch die Informationen aus den zuvor angelegten Präferenzen und Kompatibilitätslisten automatisieren. Eine komplementierende Liste, als *Call-In Plan* bezeichnet, enthält die Zeitschlitze, in denen die Ressourcen für eingehende Kommunikationsanfragen verfügbar sind. Das Format der Listen folgt den Spezifikationen für das Internet Calendaring (iCal) [DS98]. Das existierende Format wurde entsprechend der Anforderungen dieser Anwendung erweitert.

Initialisierungsphase Ein Nutzer, der den Digital Call Assistant verwenden möchte, schreibt sich bei der Applikation ein. Während dieses Vorgangs sendet der Nutzer seine ihm zur Verfügung stehenden Kommunikationsressourcen. Damit kann die initial erstellte Liste aktualisiert werden. Die Kategorisierung der Ressourcen erfolgt nach dem oben beschriebenen Prinzip.

Die Planungsapplikation fragt die Kalender (z. B. den Gruppenkalender oder den privaten Kalender) des Nutzer nach den aktuellen Einträgen ab. Der Abfragehorizont kann zeitlich (z. B. 12 Stunden) oder nach Anzahl der Einträge limitiert werden. Mittels dieser Daten sowie den oben beschriebenen Zuordnungen wird ein aktueller *statischer* Rufplan erstellt, der eine grobe Struktur für den Algorithmus liefert.

Die einzelnen Listen der Ressourcen werden in einem azyklischen gerichteten Graphen (DAG) gespeichert. Jede Entität E kontrolliert dabei einen Satz an Ressourcen Ψ. Eine Ressource ihrerseits verwaltet die Zeitschlitze, in denen die Kommunikation mit dieser Ressource möglich und gewünscht ist. Zeitschlitze werden als Knoten im Graphen gespeichert und mit v_n bezeichnet. Jeder Zeitschlitz besteht mindestens aus den

Informationen seiner Startzeit t_s und seiner Endzeit t_e, geschrieben als

$$v_n = (t_{s_n}, t_{e_n}) \qquad n \in |\Psi| \tag{8.5}$$

Kommunikationsanfrage Möchte der Nutzer jemanden erreichen, so spezifiziert er seinen Kommunikationswunsch. Dazu übermittelt er dem Digital Call Assistant die Adresse des Gesprächspartners sowie den Typ der Kommunikation. Eine Liste von Typen ist für die Anwendung definiert worden. Darüber hinaus kann der Nutzer die Art der Kommunikation beschreiben. Diese Attribute beschreiben die Kommunikationsanfrage und werden vom Planungsalgorithmus in die Bewertung einbezogen. Die folgenden Attribute werden von der Applikation unterschieden:

asynchronous: Beschreibt Kommunikation wie Fax oder Store-and-Forward Dienste wie E-Mail.

synchronous: Attribut für interaktive Echtzeitkommunikation, z. B. Telefonie.

uni-directional: Einwegaustausch von Informationen, der keiner Rückantwort bedarf.

bi-directional: Ein Dialog ist erforderlich bei dieser Anfrage.

informative Der Zweck der Kommunikation ist rein informativer Natur.

urgent: Die Anfrage soll mit der höchsten Priorität behandelt werden.

private: Der Gesprächspartner muss direkt erreicht werden.

public: Wenn der gewünschte Teilnehmer nicht erreicht werden kann, so kann ein ebenfalls informierter Kollege als Gesprächspartner gewählt werden.

Jede Kommunikationsanfrage wird in eine Regel r gefasst, die als Tupel dargestellt werden kann:

$$r = (\mathrm{id}, \mathrm{target}, \mathrm{attribute}, t_s, t_e, p, \Psi) \tag{8.6}$$

Hierbei bezeichnet t_s die Startzeit und t_e die Endzeit, p die Priorität der Regel und Ψ den Satz an Ressourcen. Die aufgestellten Regeln werden in die Regelliste R aufgenommen. Dabei können unterschiedliche Sortierungscharakteristiken wie zeitliche Sortierung oder Sortierung nach Priorität gewählt werden. Zusätzlich können Operatoren genutzt werden, welche die Beziehungen zwischen den Regeln näher spezifizieren. So kann eine Reihefolge $(r_1; r_2)$ von Regeln oder aber eine Disjunktion von Regeln $(r_1 \vee r_2)$ festgelegt werden.

Abgleichphase Während der Planungsphase werden der Call-Out-Plan des Anrufers mit dem Call-In-Plan des Angerufenen verglichen und die verfügbaren überlappenden Zeitschlitze bestimmt. Dazu wird der aktuelle Call-In Plan des Gesprächspartners angefordert. Der Austausch der Informationen folgt einem ähnlichen Konzept wie beim Austausch von Kontextinformationen [GAS04a]. Ein geeignetes Konzept, das die Privatsphäre der Nutzer respektiert und das Vertrauen der Teilnehmer berücksichtigt, ist notwendig, um die sensiblen Daten zu schützen. Eine Verknüpfung von Vertrauensbeziehungen (*trust*) und Kontexten wurde in [MGA+04] untersucht.

Abbildung 8.10(a) zeigt die Listen Call-In-Plan und Call-Out-Plan mit jeweils drei Kommunikationsressourcen, die hier jeweils exklusiv zur Verfügung stehen. Diese beiden Listen, mit Ψ und Ψ' bezeichnet, stellen die Eingabe des Zeitschlitzfindungsalgorithmusses (Algorithmus 8.1) dar.

Algorithmus 8.1 Zeitschlitzfindungsalgorithmus

```
Require: Ψ und Ψ' als nach Startzeit geordnete Liste
 1: function MAPPING(Ψ, Ψ')
 2:     i ← 0
 3:     for all v ∈ Ψ do
 4:         for all v' ∈ Ψ' do
 5:             if t_s ≥ t'_e then
 6:                 next
 7:             else if t_e ≤ t'_s then
 8:                 next
 9:             else
10:                 T[i] ← MATCH(v, v')
11:                 i ← i + 1
12:             end if
13:         end for
14:     end for
15:     return T
16: end function
```

Der Algorithmus ruft die Methode MATCH auf, die die maximal überlappende Zeitspanne zweier Knoten (v_n, v'_n) findet, oder andernfalls eine leere Menge zurückliefert. Die Funktion ist in Algorithmus 8.2 beschrieben.

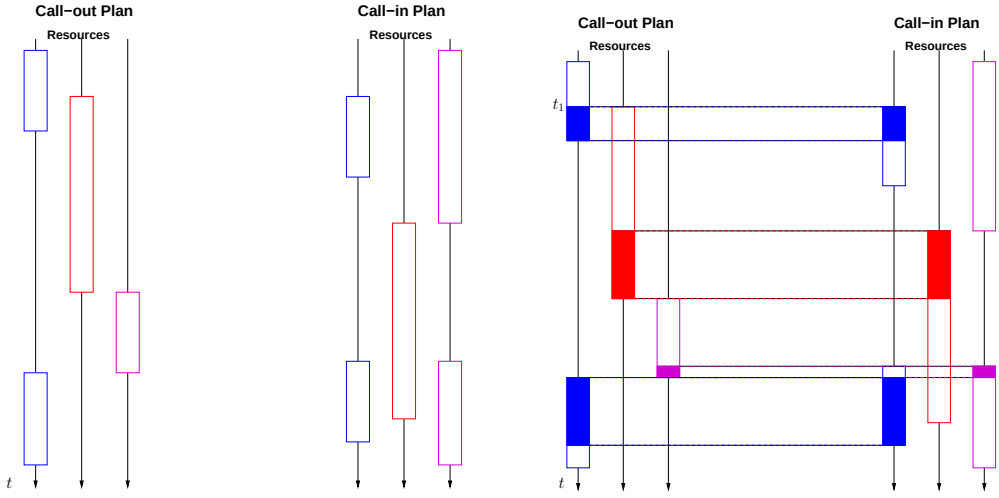
Ein Nachteil des Algorithmus 8.1 liegt in seiner exponentiellen Laufzeit von $O(n^2)$. Diese entsteht durch den paarweisen Vergleich aller Knoten aus Ψ mit allen Knoten aus Ψ'. Durch Modifikation des Algorithmusses kann allerdings die Laufzeit deutlich reduziert werden. So ist kein kompletter Vergleich notwendig, wenn die Ressourcen miteinander nicht kompatibel sind. Darüber hinaus kann die erste Übereinstimmung zweier Ressourcen als *Pivotelement* gespeichert werden. Anschließend muss kein folgender Knoten aus Ψ mehr mit denjenigen Knoten aus Ψ' verglichen werden, deren Startzeit vor der Startzeit des Pivotelements liegt.

Das Ergebnis der Planungsphase ist eine zeitlich geordnete Liste von überlappenden Regionen für jede Ressource. Eine graphische Darstellung der Überlappung ist in Abbildung 8.10(b) dargestellt. Die Startzeit der ersten Überlappung ist mit t_1 bezeichnet.

Rufaufbauphase Der Digital Call Assistant zeigt dem Nutzer den jeweiligen Status für die angestoßenen Kommunikationsanfragen an. Während erfolgsversprechender Zeitschlitze wird dem Nutzer die Möglichkeit angeboten, einen Rufaufbau auszuführen. Sollte der Ruf nicht erfolgreich ausgehandelt werden können, kann die Anfrage wieder in die Liste der offenen Anfragen eingetragen werden.

Algorithmus 8.2 Überlappungsalgorithmus

```
1: function MATCH(v_1, v_2)
2:     s ← min(t_{s_1}, t_{s_2})
3:     e ← max(t_{e_1}, t_{e_2})
4:     if e > s then
5:         return (s, e)
6:     else
7:         return null
8:     end if
9: end function
```



(a) Call-In Plan und Call-Out Plan

(b) Mögliche gemeinsame Zeitschlitze nach Anwendung des Matching Algorithmusses

Abbildung 8.10: Überdecken der Ressourcen des Call-In-Plans und des Call-Out-Plans

8.4.1.2 Erweiterung des Digital Call Assistant zur Anpassung an dynamische Änderungen

Der Ablauf, der in Abbildung 8.8 dargestellt ist, zeigt eine statische Aushandlung von Zeitslots auf Basis der in Initialisierungsphase bereitgestellten Informationen. Eine Verbesserung der ausgehandelten Zeiten kann durch die Adaption an die aktuellen Kontexte der Nutzer erreicht werden. So kann ein aus dem elektronischen Kalender gewonnene Information über die Dauer einer Besprechung oder Konferenz hinfällig werden, wenn diese länger dauert.

Jeder Kalendereintrag sowie die freien Zeiten und die zur Verfügung stehenden Kommunikationsressourcen können durch assoziierte Kontexte angereichert werden. Der Digital Call Assistant subskribiert sich dazu beim ContextServer für die Änderungen des Kon-

texts des Nutzers. Bei einer Änderung des Kontexts, wird die Anwendung informiert und kann die Berechnungen für die Call-In und Call-Out-Pläne erneut durchführen. Auch ausgehandelte Zeitschlitze können durch Neuaushandlung zwischen den beiden Digital Call Assistant-Komponenten angepasst werden.

Die Erweiterungen finden in der *Aushandlungsphase*, die nachfolgend beschrieben ist statt. Ein Ablauf mit jeweils zwei Kontextänderungen ist in Abbildung 8.11 gezeigt. Mit ≈ wurde gekennzeichnet, dass Schritte übersichtshalber weggelassen wurden.

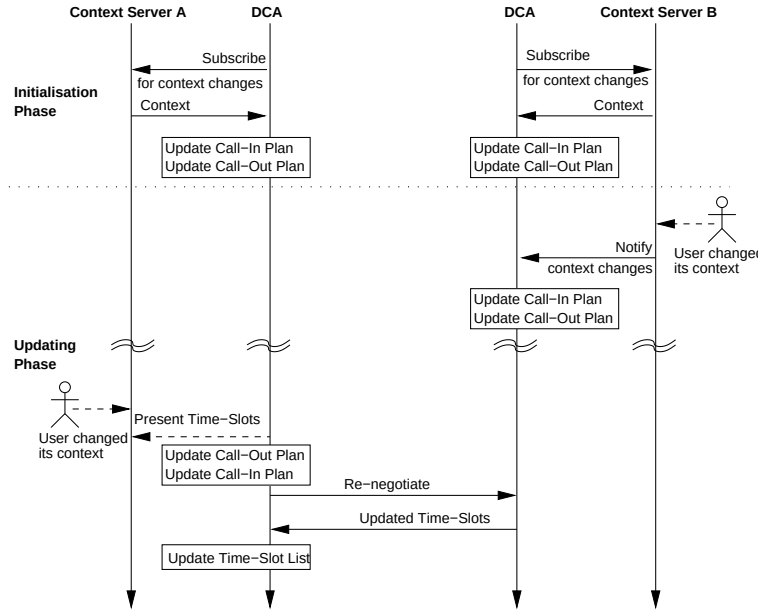


Abbildung 8.11: Erweiterung des Digital Call Assistant zur Adaption an Kontextänderungen

Aktualisierungsphase Nach der Bestimmung der Zeitschlitze kann es zu einer Änderung der verfügbaren Zeiten und Ressourcen kommen. Daher werden bei einer wartenden Kommunikationsanfrage Änderungen zwischen den beiden Applikationen ausgetauscht. Nach dem Austausch werden die neuen Informationen in eine neue Bestimmung der Zeitschlitze mit einbezogen.

8.4.2 Umsetzung des Digital Call Assistant

Das beschriebene Konzept des Digital Call Assistant wurde in einer prototypischen Implementierung umgesetzt. Der Umfang der Realisierung erfüllt nicht alle zuvor aufgeführten Funktionen. Dennoch ist ein Abgleich der Kommunikationsressourcen sowie die Verwendung eines elektronischen Kalenders möglich. Der Digital Call Assistant wurde wie in Abbildung 8.7 gezeigt, in das bestehende SIP-basierte IP-Telefonie-System integriert. Als Basis für den Digital Call Assistant wurde das Open-Source Projekt *Mozilla Sunbird* [www27] gewählt. Ziel des Projektes ist die Realisierung eines eigenständigen

(*standalone*) Cross-Plattform Kalenders. Dieser baut auf einer Software-Bibliothek auf, die Werkzeuge für die Erstellung und Modifikation von iCalendar-konformen Kalendern zur Verfügung stellt. Darüber hinaus können mittels der Nutzerschnittstellensprache (*user interface language*) Mozilla XUL graphische Komponenten erstellt werden.

Der Digital Call Assistant ist eine Erweiterung einer Entwicklerversion (build 2.4) von Sunbird. Als Format wurde das iCal-Format gewählt und um weitere Funktionen erweitert, was in Unterabschnitt 8.4.2.1 beschrieben ist. Die Einträge in diesem Format werden zusammen mit den Nutzerpräferenzen in Regeln r umgewandelt, welche für den weiteren Ablauf verwendet werden. Für die in Unterabschnitt 8.4.1.1 beschriebene Abgleichphase werden die Daten im iCalendar-Fomat übertragen. Als Transportprotokoll wurde das iCalendar Transport-Independent Interoperability Protocol (iTIP) [SMDH98] verwendet.

8.4.2.1 Erweiterung des iCalendar-Formats

Die Basisspezifikation des Internet Calendaring beschreibt einen Satz an Datenfeldern für gemeinsam genutzte und verteilte Kalenderfunktionen. Für das definierte Format wurde ein MIME Typ gewählt. Die folgenden Kalenderkomponenten wurden definiert: `VEVENT`, `VTODO`, `VJOURNAL`, `VFREEBUSY` und `VTIMEZONE`. Diese Komponenten wurden, wann immer möglich, für den Digital Call Assistant verwendet. Zur Übertragung der Kalenderevents wurde iTIP als Protokoll spezifiziert.

Der Calendar User Type (`CUTYPE`) erlaubt die Spezifizierung des Kalendernutzers und wurde im Rahmen der Arbeit um die Parameter `ROOM`, `RESOURCE` und `CONTEXT` erweitert. Die Erweiterungen erlauben es, die Lokation, die verfügbaren Kommunikationsressourcen und den Kontext des Nutzers zu spezifizieren. Zur Kennzeichnung eines freien oder eines belegten Zeitschlitzes wurde der Parameter `FBTYPE` verwendet. Dieser erlaubt die Unterscheidung von verschiedenen Typen von belegten Zeitschlitzen. Daten können zwischen Anwendungen, welche die beschriebenen Modifikationen an der Syntax des iCal-Formats verarbeiten können, ausgetauscht werden. Soll ein globaler Austausch mit Anwendungen, mit denen vorher keine Vereinbarungen bezüglich des Formats getroffen werden soll, erfolgen, so müssen die neuen Typen bei der Internet Assigned Numbers Authority (IANA) angemeldet werden.

Listing 8.1: iCalendar data used for the Digital Call Assistant

```
BEGIN:VEVENT
DTSTAMP:20040613T220546Z
ORGANIZER:MAILTO:mgoertz@kom.tu-darmstadt.de
CREATED:20040613T215649Z
UID:DigitalCallAssistant
LAST-MODIFIED:20040613T220437Z
SUMMARY:Meeting
LOCATION:S3|06/348
CLASS:PUBLIC
PRIORITY:3
CATEGORIES:Conference
```

```
CUTYPE:RESOURCE: sip-phone, mobile-phone
FBTYPE=BUSY:20040616T109000Z/20040616T103000Z
DTSTART:20040616T090000Z
DTEND:20040616T100000Z
TRANSP:OPAQUE
END:VEVENT
```

8.4.3 Unerwünschte Interaktion in den Regeln

Die Verwendung einer regelbasierten Kommunikation führt zu einer Vereinfachung der Erstellung von nutzerzentrierten Diensten. Der Nutzer kann seine Aktionen in einer sehr abstrakten Sprache formulieren. Die technischen Details werden verdeckt und durch einen Interpreter betrachtet und umgesetzt.

Während die einzelnen Regeln durch syntaktische Prüfungen relativ einfach auf mögliche Fehler überprüft werden und durch die graphische Eingabe unerlaubte Operationen verhindert werden können, kann das gesamte Regelwerk schwerer auf Korrektheit untersucht werden. Die einzelnen Regeln können sich widersprechen oder durch ihre Kombination zu unerwünschten Resultaten führen. Dies entspricht dem in Unterabschnitt 2.1.2 vorgestellten Problem der unerwünschten Dienstinteraktion, jedoch auf der Regelebene.

Dies ist ein Phänomen, welches auch in Policy-Sprachen wie Ponder [Dam02], RuleML [BTW01] oder in [KFJ03, Kag04] auftritt. Das Problem hat mittlerweile auch Beachtung erfahren [LS99, CLN00, HW03, RMT04, BMR04]. Einzelne spezielle Ansätze, wurden insbesondere im Bereich Sicherheit entwickelt [AKGM00, DCG+04, NLiMK04, NLiMK03]. Eine Erweiterung des vorgestellten regelbasierten Kommunikationssystems ist die Einführung solcher Konflikterkennungs und -vermeidungsverfahren.

8.5 Zusammenfassung

Die in diesem Kapitel dargestellten Diensterstellungsmethoden stellen eine im Vergleich zum Service Engineering-Prozess klassischer Telefoniedienste neuartige Form der Diensterstellung dar. Der Nutzer kann mittels einer Skriptsprache, wie mit der in Abschnitt 8.2 beschriebenen erweiterten Call Processing Language, sichere, eigene Dienste erstellen und ausführen lassen.

Ein weiterer Schritt in Richtung einfacher nutzerzentrierter Diensterstellung wird mit der Bereitstellung eines graphischen Editors erreicht. Über diesen kann der Nutzer aus einer vordefinierten Anzahl von Komponenten auswählen und diese geeignet verknüpfen. Dabei können bereits Plausibilitätsprüfungen durchgeführt werden. Ein Editor, der die erweiterte CPL-Syntax unterstützt, wurde in Abschnitt 8.3 vorgestellt.

Der Einsatz eines Kommunikationsbrokers ermöglicht es dem Nutzer, die oftmals zeitaufwändige Kommunikationsaushandlung auf das System zu verlagern. Der Nutzer spezifiziert dabei seine Kommunikationswünsche in Form von Regeln. Er beschreibt dabei,

wann und mit wem er mit welchen Kommunikationsgeräten und welcher Priorität kommunizieren möchte. Der Digital Call Assistant handelt dazu Zeitschlitze für eine Kommunikation aus, die für beide Gesprächspartner günstig sind. Dadurch kann die Anzahl der erfolglosen oder störenden Anrufe verringert werden.

Zur Zeit müssen die Regeln vom Nutzer manuell erstellt werden und bei einer Änderung seiner Wünsche wieder modifiziert werden. Auch wenn die Erstellung von Kommunikationsdiensten mittels eines graphischen Editors einfach, schnell und fehlervermeidend durchgeführt werden kann, so muss der Nutzer die notwendigen Schritte ausführen. Die Verwendung *selbst-lernender Verfahren* stellt die nächste Stufe in Richtung „unsichtbarer Helfer“ für die Unterstützung der Kommunikationssteuerung dar. In einer Lernphase werden die Aktionen und das Verhalten des Nutzers aufgezeichnet und analysiert. Aus diesen Daten werden Muster abgeleitet, die in Regeln übersetzt werden können. In der Trainingsphase schlägt das System dem Nutzer seine Entscheidungen vor, die dieser bewertet. Im Idealfall wird danach die Nutzerinteraktion auf ein Minimum reduziert.

Wichtige Punkte sind die Berücksichtigung der Privatsphäre sowie arbeitsrechtliche Richtlinien. Die Überwachung des Nutzers, um sein Verhalten aufzuzeichnen sowie die gewonnenen Sensordaten sind einschneidende Vorgehensweisen, welche der umfassenden Zustimmung über den Umfang der gesammelten Daten bedürfen. Dies gilt ebenso für die Speicherung der Informationen und deren Zugriffsrechte. Ein durchdachtes Sicherheitskonzept und eine vertragliche Regelung sind unabdingbare Vorbedingungen für den Einsatz der dargestellten regel-basierten bzw. selbst-lernenden Kontext-bewussten Kommunikation.

Kapitel 9

Entwickelte Komponenten & Verfahren

> Die Idee ist das Absolute, und alles
> Wirkliche ist nur Realisierung der Idee.
>
> Georg W. F. Hegel

In Kapitel 3 wurden Lösungsansätze zur Steigerung der Effizienz bei der Steuerung von Kommunikationsflüssen sowie zur Findung von geeigneten Zeitpunkten für eine Kommunikation vorgeschlagen. Die notwendigen Verfahren wurden konzeptionell in den Kapiteln 6, 7 und 8 herausgearbeitet und gegen andere Ansätze abgegrenzt. Dieses Kapitel stellt einen vertikalen Abriss von Umsetzungen ausgewählter Konzepte aus allen Phasen des Kontext-Spiral-Modells dar.
Exemplarisch werden Auswertungen von Messungen sowie Implementierungsdetails für ein Innenraumlokationsverfahren mittels Signalstärkenmessung zur Erfassung von Kontextmerkmalen sowie ein System mit Fuzzy Logik Regeln als Inferenzverfahren zur Synthetisierung von Kontexten vorgestellt.
Die Architektur und die Schnittstellen des ContextServers als Integrationskomponenten zur Speicherung und Verteilung von Kontextobjekten werden dargelegt. Als repräsentative Kontextnutzer werden erweiterte SIP User Agents (XUA), die Kontext-bewusste Rufsteuerdienste bereitstellen, sowie eine modifizierte CPL-Engine, die Komponenten für die Erstellung Kontext-bewusster Kommunikationsskriptdienste bereitstellt, ausgewählt und beschrieben.

Kurzübersicht

Das vorliegende Kapitel besitzt folgende Struktur. In Abschnitt 9.2 wird ein Innenraumlokationsverfahren und dessen Umsetzung vorgestellt. Das Verfahren beruht auf

der Messung von Signalstärken in einer Wireless-LAN-Infrastruktur und dem Ähnlichkeitsmaß zu Referenzpunkten. Die Messungen der Signale sowie die Bewertung der erzielten Genauigkeiten bei der Bestimmung von Räumen sind angegeben. Ein Prototyp, der auf einem PDA lauffähig ist, wurde entworfen und umgesetzt.

Ein Programm, das ein auf Fuzzy Logik beruhendes Inferenzverfahren zur Kontextsynthetisierung realisiert, wird in Abschnitt 9.3 vorgestellt. Die Architektur und die Funktionsweise des ContextServers sowie seine Web Service-Schnittstellen sind in Abschnitt 9.4 beschrieben. Erweiterungen des SIP-Protokolls zur In-band Signalisierung von Anfragen nach dem Nutzerkontext und der Kontextverteilung führten zur Modifikation der SIP-Entitäten wie dem eXtended User Agent (XUA). Die Erweiterungen und Implementierungsdetails sind in Unterabschnitt 9.5.1 aufgeführt. Die Erweiterungen umfassen auch die Anpassung der CPL-Engine. Abbildung 9.1 stellt die Struktur des Kapitels dar.

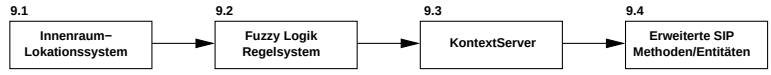


Abbildung 9.1: Visuelle Darstellung der Struktur des Kapitels 9

9.1 Übersicht der Komponenten des Gesamtsystems

Das entworfene Gesamtsystem besteht aus einer Vielzahl an verteilten Komponenten. Für einen Großteil der in den bisherigen Kapiteln vorgestellten Verfahren wurden prototypische Implementierungen durchgeführt. Nachfolgend werden ausgesuchte Komponenten, die umgesetzt und evaluiert wurden, vorgestellt. Diese gibt einen Eindruck davon, wie hoch der Aufwand der Umsetzung eines solchen Systems zum Nutzen ist.

Eine Übersicht der Komponenten und ihrer Beziehungen zueinander ist, in vereinfachter Form, in Abbildung 9.2 dargestellt. Es wurde bewusste auf einem zu hohen Detailgrad verzichtet, die die Komponenten ausführlich in den nachfolgenden Abschnitten behandelt werden.

9.2 WLAN-basierte Lokationsgewinnung

Die Bestimmung der aktuellen Position[1] eines Nutzers oder eines Gerätes ist ein wichtiger Bestandteil des emergenten Mobile Computings. Eine Vielzahl an unterschiedlichen Positionsbestimmungstechnologien und -systemen wurden entwickelt. Eine umfangreiche Übersicht der Ansätze ist in [HB01] und [Nel98b] zu finden.

[1] Unter *Ort* wird in dieser Arbeit eine räumliche Ausdehnung, wie ein Raum oder ein Platz verstanden. Mit *Position* wird die exakte Lokalität einer Entität beschrieben. Der Begriff *Lokation* beinhaltet beide Bezeichnungen.

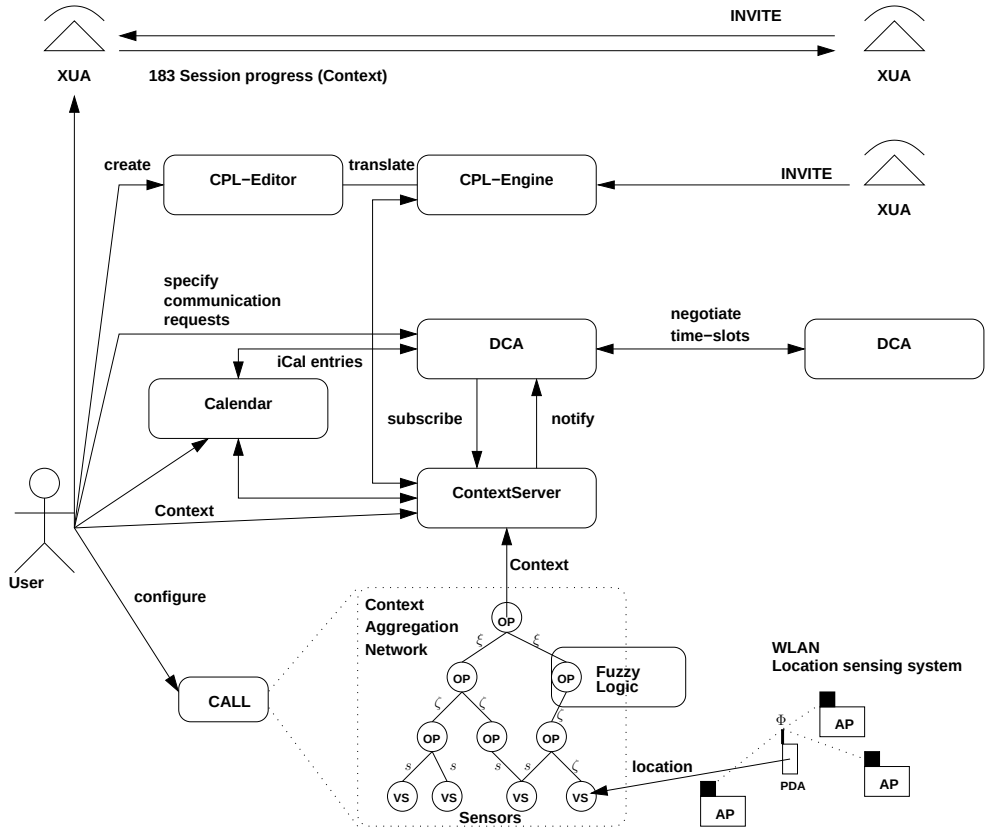


Abbildung 9.2: Entworfenes Kontext-bewusstes Gesamtsystem

9.2.1 Verfahren zur Lokationsbestimmung

Die Lokation einer Entität stellt ein wichtiges Kontextmerkmal dar, das oftmals bereits, für sich betrachtet, eine sehr aussagekräftige Information darstellt. Lokation wird daher auch als *primärer Kontext* eingestuft. Eine Reihe von Diensten bezieht die Lokalität mit in die Parametrisierung ein. Diese ortsbezogenen Dienste werden als Location Based Services (LBS) bezeichnet. Elektronische Touristenführer [Mit98, CDMF00, CDM+00, LKRF99, YYDW99, DCME01, AAH+97] oder Restaurant- und Shop-Finder [ACK94] sind häufig referenzierte Dienste für *Location Based Services* (LBS).

9.2.1.1 Taxonomie Lokationsbestimmungstechniken

Die betrachteten Lokationsbestimmungstechniken haben verschiedene Eigenschaften und Einsatzgebiete. Dennoch haben sich einige grundlegende Eigenschaften bei allen Verfahren herauskristallisiert. Ein Lokationsbestimmungssystem operiert entweder in einem *absoluten* oder in einem *relativen* Bezugssystem. In einem absoluten System

teilen alle Objekte dasselbe Koordinatensystem zur Lokalisierung. Ein klassisches absolutes Lokalisierungssystem ist das Global Position System (GPS), welches auf einem Koordinatensystem nach Universal Transverse Mercator (UTM) operiert und Daten bezüglich der Länge, Breite und Höhe liefert. In einem relativen System hat jedes Objekt seinen eigenen Referenzbezug; so liefert ein Abstandssensor die Distanz zum Objekt relativ zu seiner eigenen Position.

Die Lokationsinformation eines Lokationsbestimmungssystems kann als *physikalische* Messgröße oder als *symbolische* Information geliefert werden. Ein GPS-Empfänger zeigt z. B. an, dass das Gerät die Koordinaten 49°86'07"N by 8°65'00"E bestimmt hat. Ein Innenraumlokationssystem, das Rauminformation in Form von periodisch versendeten Textstrings erhält, würde beispielsweise einen Raum über die Information „$(S3|06/345)$“ identifizieren. In der Regel sind symbolische Lokationsinformationen Ortsinformationen auf einer gröberen Auflösungsebene, während physikalische Informationen eine genauere Positionierung erlauben. Physikalische Lokationsinformationen können durch Zuordnung mit einer symbolischen Darstellung, die in einer zentralen Datenbank gehalten wird, angereichert werden.

9.2.1.2 Verfahren zur Bestimmung von Lokationen

Zur Messung von Lokationen können eine Reihe von Verfahren angewendet werden. Eine häufig angewendete Technik ist die *Triangulation*. Bei dieser werden die geometrischen Eigenschaften eines Dreiecks ausgenutzt, um die Position eines Objektes zu bestimmen. Das Verfahren funktioniert mit Distanzmessungen zu mehreren bekannten Referenzpunkten (*Lateration*) und alternativ mit Winkeln (*Angulation*). Für eine Positionsbestimmung in 2D werden 3 nicht-kollineare Punkte, für eine Bestimmung in 3D werden 4 nicht-koplanare Messpunkte benötigt.

Für die Distanzmessung kommen in den meisten Verfahren drei Techniken als generelle Ansätze zum Einsatz:

Direkte Messung: Die Distanz wird durch eine physische Messung bestimmt. Ein Messstab könnte z. B. so weit ausgefahren werden, bis er auf einen Widerstand trifft.

Laufzeit: Bei Laufzeitverfahren (*Time-of-Flight – TOF*) wird die Zeit gemessen, die notwendig ist, um von einem Objekt zu einem bestimmten Punkt P zu kommen. Dabei wird üblicherweise die Laufzeit eines gesendeten Signals zwischen Sender und Empfänger gemessen. Bei der Zeitmessung spielen genau synchronisierte Uhren eine wesentliche Rolle.

Dämpfung: Die Intensität von emittierten Signalen nimmt mit der Entfernung von der Quelle ab. Die Abnahme in Relation zur Anfangsintensität wird als *Dämpfung* bezeichnet. Die Dämpfung kann aus der empfangenen Intensität am Punkt P der Messung bestimmt werden. So nimmt ein Radiosignal im offenen Feld mit einer Dämpfung, die proportional zu $1/r^2$ ab, wobei r die Distanz zwischen dem Sender und dem Punkt P ist.

Indirekte Messung: Messverfahren müssen Hindernisse in der Umgebung, Reflektionen, Brechungen und Mehrwegausbreitungseffekte bei der Signalverbreitung berücksichtigen können. Diese Faktoren können sich negativ auf die Genauigkeit der Messung auswirken.

Nachbarschaft: Eine weitere Technik zur Ortsbestimmung ist das *Nachbarschaftsverfahren* (*proximity*), bei dem bestimmt wird, ob sich ein Objekt in der Nähe eines Punktes befindet, dessen Position bekannt ist. Üblicherweise werden zur Messung physische Phänomene verwendet, die eine eingeschränkte Reichweite haben. Provider von Mobilfunknetzen bieten Location Based Services an, welche die Ortsinformation des Nutzers aus der Information beziehen, welches Gerät sich in welcher Global System for Mobile Telecommunication (GSM)-Zelle befindet. Die meisten Badge-basierten Verfahren nutzen eine ähnliche Technik.

9.2.2 Lokationsbestimmung mit PDA und WLAN

Ein Innenraumlokationsverfahren, das die weite Verbreitung von IEEE 802.11 Wireless Local Area Network (WLAN) Installationen nutzt, wurde in [BP00, BBP00] unter dem Namen RADAR vorgestellt. Die Grundidee des Ansatzes ist, dass der Energielevel eines Signals in einem Radionetzwerk (*Radio Frequency – RF*) eine Funktion des Abstands zwischen dem Sender und dem Empfänger ist. Hieraus kann auf den Ort des Empfängers geschlossen werden. Als entscheidende Größe zur Bestimmung des Ortes wird die *Signalstärke* bzw. die Signal-to-Noise Ratio (SNR) zwischen dem Objekt, welches die Messung durchführt und einer Basisstation (*Access Point – AP*) verwendet.

Zwei Verfahren, die eine Abbildung der Signalstärke auf eine Ortsinformation erlauben, wurden vorgeschlagen. Ein *Nachbarschaftsansatz* (*neighborhood*) vergleicht das gemessene Tupel aus Signalstärken zu n Basisstationen mit Einträgen in einer Datenbank, hier als *Radio Map* bezeichnet, die vorher gemessene Referenzpunkte beinhaltet. Der Ort der messenden Entität ist der des Referenzpunktes mit der kleinsten Distanz zu den Messwerten. Ein zweites Verfahren bestimmt aus der Signalstärke direkt die Distanz zur Basisstation, indem die Dämpfung der Signalstärke durch Wände und andere Objekte als Funktion des Abstands mit einbezogen wird. Das erstellte Signalausbreitungsmodell verwendet einen Wanddämpfungsfaktor (*Wall Attenuation Factor – WAF*) und berücksichtigt auch Mehrwegeausbreitungsprobleme. Die Genauigkeiten werden in [BP00] mit 3 m bzw. 4.3 m bei 50% Wahrscheinlichkeit angegeben.

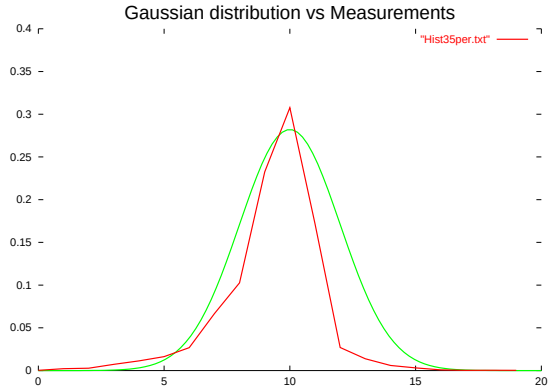
Ein eigenes Lokalisierungssystem, basierend auf dem RADAR-Ansatz wurde entwickelt, um präzise Aussagen zur Handhabbarkeit und Genauigkeit treffen zu können. Die Details der Umsetzung sowie in diesem Kapitel nicht aufgeführte Ergebnisse der durchgeführten Messungen sind in Anhang A aufgeführt. Einige ausgewählte und relevante Messungen sind nachfolgend erläutert.

9.2.2.1 Untersuchung der Charakteristika der Signale

Mit einem Messprogramm wurden in einer Langzeitmessung die Charakteristika der Signale von Access Points bestimmt. Eine getroffene Annahme ist, dass die Signalstärke jedes Signals *unabhängig* von der Anzahl und der Verteilung anderer Basisstationen und Endgeräte ist. Der zentrale Grenzwertsatz (*central limit theorem*) sagt aus, dass die Verteilung einer großen Anzahl an unabhängigen Vorkommnissen zu einer Gauss*'schen Normalverteilung* tendiert. Die dazugehörige Wahrscheinlichkeitsdichtefunktion (*Probability Density Function – PDF*) ist in (9.1) angegeben.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}, \qquad -\infty \leq x \leq +\infty \tag{9.1}$$

Die Verteilung (*distribution*) der gemessenen Signalstärken von einer Basisstation an einen bestimmten Ort ist in Abbildung 9.3 dargestellt. Die Form der Kurve zeigt, dass sie näherungsweise eine Normalverteilung darstellt. Die getroffenen Annahmen der Unabhängigkeit der Signalstärken kann aufrecht erhalten werden. Die Abweichung von der typischen Glockenform kann durch die limitierte Auflösung des Treibers der WLAN-Karte von 1 dBm sowie durch die doch gegebene minimale Interferenz mit anderen Basisstationen oder Sendern erklärt werden.



Auswertung	Signal-stärke (dBm)
Mittelwert	-73 (-72,716)
Median	-73
Mode	-72
Std. Abw. s	2,25

Abbildung 9.3: Verteilung der Signalstärke sowie statistischer Eigenschaft für die Auswertung der Messung der Verteilung der Signalstärke

Um die Beziehung zwischen der Signalstärke und der Entfernung zwischen Sender (Basisstation) und Empfänger (mobiles Endgerät) zu untersuchen, wurden anhand der realen Infrastruktur und Verteilung der Basisstationen Messungen unternommen. Die Ergebnisse dieser Messreihen sind in Abbildung 9.4 dargestellt. Wurde kein Signal von einer Basisstation empfangen, so wurde dieser Wert als -100 dBm aufgetragen. Die Kurven zeigen das erwartete Verhalten, dass die empfangene Signalstärke von der Entfernung zwischen den beiden Entitäten abhängig ist. Aus dieser Relation kann somit näherungsweise die Entfernung zwischen den beiden Entitäten bestimmt werden. Diese Beziehung ist die Grundvoraussetzung, um ein Lokationsbestimmungsverfahren auf Basis der Signalstärken zu realisieren.

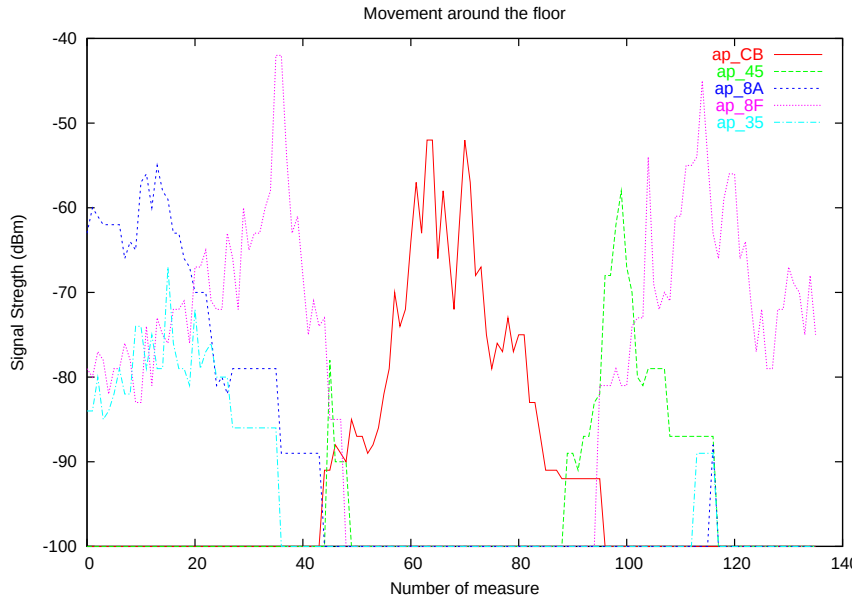


Abbildung 9.4: Gemessene Signalstärken der Access Points gemessen während der Bewegung innerhalb der WLAN-Infrastruktur

Eine Reihe weiterer Untersuchungen der Charakteristika der Signale wurde durchgeführt, um zu klären, ob ein Ortungssystem auf Basis der Signalstärken und mit der vorhandenen Infrastruktur zu verlässlichen und genügend genauen Ergebnissen führt. Im wesentlichen wurden die folgenden Fragen betrachtet:

- Ist die *Stabilität* der Signalstärken über einen längeren Zeitraum und zu verschiedenen Tageszeiten konstant?
- Ist eine genügend hohe *Trennschärfe* zwischen Signalstärken in einzelnen Räumen gegeben?
- Wie wirkt sich die *Orientierung* des Endgerätes auf die Messung der Signalstärken aus?

Eine Messung über 27 Stunden (8:30 bis 11:30 des nächsten Tages) wurde durchgeführt, um die Stabilität des Signals über einen längeren Zeitraum zu betrachten. Hierzu wurde alle 10 s ein Messwert aufgezeichnet, während das Messgerät in einer festen Position blieb. Um ein Konfidenzintervall von 95% mit einem maximalem Fehler von $\varepsilon = 1$ dBm gewährleisten zu können, wurde mittels Gleichung (9.2) die Anzahl n der notwendigen Messwerte berechnet. Als z-Quantille wurde die Normalverteilung gewählt, als Standardabweichung $s = 3,25$ wurde der höchste gemessene Wert eingesetzt. Die Anzahl der notwendigen Messwerte von $n = 41$ wurde bei allen Messungen um ein Vielfaches übertroffen.

$$n' = \lceil n \rceil = \left\lceil \left(\frac{zs}{\varepsilon} \right)^2 \right\rceil \tag{9.2}$$

Die Messung der Signalstärken ist in Abbildung 9.5 dargestellt. Drei signifikante Bereiche lassen sich an der Kurve ablesen. Der erste Bereich erstreckt sich über die Arbeitszeit am Lehrstuhl (8:30 bis 19:30). Während dieser Zeit variiert das Signal sehr stark. In den Nachtstunden (19:30 bis 8:30) zeigt das Signal ein wesentlich konstanteres Verhalten. Das letzte Drittel des Bildes zeigt wieder einen Bereich während der Arbeitszeit. Da sich beide Bereiche so stark unterscheiden, wurden jeweils zwei eigene Histogrammauswertungen erstellt, die in den Abbildungen 9.6(a) und 9.6(b) dargestellt sind.

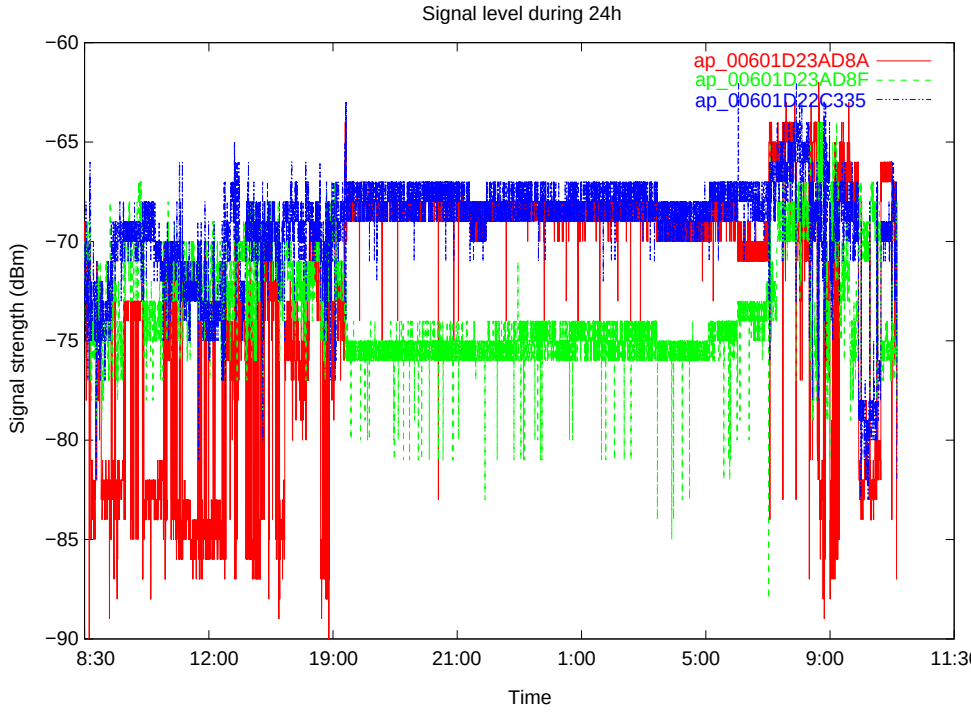
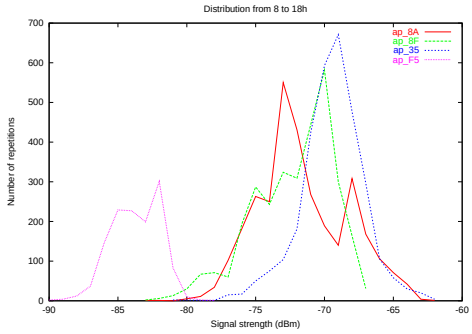
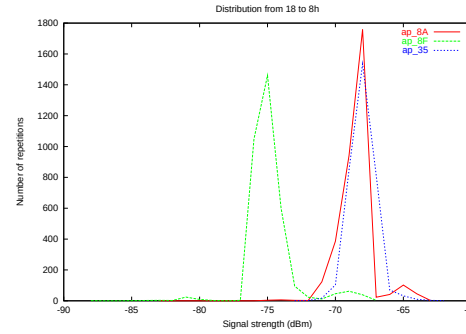


Abbildung 9.5: Messung über 27 Stunden

Eine statistische Auswertung mit den relevanten Indizes ist in Tabelle A.1 angegeben. Die konstantere Signalform in der Nacht wird auch durch die schärfere Verteilung reflektiert. Überdies kommt es zu einer Verschiebung des Mittelwerts. Eine Referenzkarte, die Mittelwerte als Fingerprints verwendet, müsste die Unterschiede in den Tageszeiten mitberücksichtigen. Die Signalstärke, die von der Basisstation 8F bei Nacht empfangen wird ist niedriger als tagsüber. Die Basisstation F5 ist nachts am Messpunkt nicht zu empfangen. Diese Effekte sind auf das Vorhandensein von Türen mit Metallgittern zurückzuführen. Der Einfluss der Infrastruktur auf die Signalstärken ist deutlich zu erkennen. Die Positionierung der Basisstationen sowie der Einfluss von Objekten muss für jedes Setup individuell bestimmt werden.



(a) Verteilung der Signalstärken der Messung von 08:30 bis 19:30



(b) Verteilung der Signalstärken der Messung von 19:30 bis 11:30

Abbildung 9.6: Verteilung der Signalstärken der Messung

9.2.2.2 Effekt der Orientierung des mobilen Endgerätes auf die gemessenen Signalstärken

Das Endgerät, das die Signalstärken misst, ist mobil und kann ein Telefonie-Endpunkt sein. Der mobile Client wird daher in verschiedenen *Orientierungen* gehalten, während die Messungen durchgeführt werden. Eine Untersuchung der Effekte der Orientierung auf die Messergebnisse der Signalstärken wurde durchgeführt. Hierzu wurde ein t-Test durchgeführt um die *Signifikanz* der Unterschiede in den Messungen in Bezug auf die Orientierung des Messgerätes zu testen. Details zum Versuch finden sich im Anhang A.3.2.2.

Die Analyse der Orientierung des Endgerätes auf die Messungen hat ergeben, dass es einen messbaren und signifikanten Unterschied zwischen den verschiedenen Orientierungen gibt. Die immanent vorhandenen Schwankungen der Signale bei fester Orientierung befinden sich in der selben Größenordnung wie die Effekte unterschiedlicher Orientierung; jedoch sind die Anforderungen an die zu erzielende Genauigkeit der Ortsbestimmung derart, dass die Berücksichtigung der Orientierung bei der Erstellung der Referenzkarte und bei der Verwendung des Systems vernachlässigt werden kann.

9.2.2.3 Prozessphasen des WLAN-basierten Lokationssystems

Das Lokationsverfahren besteht aus zwei Phasen: In der *Initialisierungsphasen* wird eine Referenzkarte der Lokationen erstellt und in der *Bestimmungsphase* wird die Lokation anhand eines Vergleichs mit der Referenzkarte bestimmt. Die beiden Phasen enthalten mehrere Schritte, die nachfolgend erläutert werden.

Initialisierungsphase Mit dem entwickelten Messprogramm werden in jedem Raum an mindestens 5 Positionen die Signalstärken zu den erreichbaren Basisstationen gemessen

Algorithmus 9.1 Pre-measurement

```
1: procedure PreMeasure                          ▷ Measure SS of each room
2:     for all r_i ∈ R do                        ▷ Measure center of room
3:         R[i] ← Fingerprint(r_i)               ▷ Store in table
4:     end for
5:     Sort(R)                                   ▷ Sort R starting with lowest MAC address
6: end procedure
```

und über einen Web Service in eine Datenbank gespeichert. Algorithmus 9.1 zeigt das Verfahren in Pseudonotation. Alle Räume, die vermessen und in der Referenzkarte gespeichert werden, sind definiert als

$$R = \{r_i \mid i \in \mathbb{N}, \quad 1 \leq i \leq N\} \tag{9.3}$$

Jeder einzelne Raum wird durch

$$r_i = \{(A_j, \Phi_j) \mid j \in \mathbb{N}, \quad 1 \leq j \leq M\} \tag{9.4}$$

beschrieben. Hierbei werden Access Points mit A_j und die zu dem Access Point gemessene Signalstärke mit Φ_j bezeichnet.

Für jeden Referenzpunkt wird ein charakteristischer *Fingerprint* $\hat{\Phi}$ erstellt, wie dies in Algorithmus 9.2 gezeigt ist. In dem eingesetzten System wurde der *Mittelwert* der Messungen verwendet. Aktuelle Verfahren, die ebenfalls auf der Messung von Signalstärken in WLAN-Umgebungen basieren, verwenden das *Histogramm* der Messungen als Fingerprint [YAS03], was zu einer höheren Genauigkeit führt.

Algorithmus 9.2 Fingerprint

```
1: procedure PreMeasure ▷ Measure a series of SS values and calculate the mean
2:     for all (A_j, Φ_j) ∈ r do
3:         for k = 0 to 100 do
4:             H[k] ← Measure(SNR_k)
5:         end for
6:         Φ_j ← Mean(H)
7:     end for
8: end procedure
```

Die Referenztabelle mit den Signalstärken-Fingerprints, die zuletzt bestimmten Ortsinformationen der Clients und ähnliche Daten werden in einer Datenbank gehalten. Das Entity-Relationship-Modell der Datenbank ist in Abbildung A.6 abgebildet. Die Verwendung einer Karte mit Referenzdaten wurde gewählt, da die bauliche Substanz der Räume sehr unterschiedlich ist und ein Verfahren, welches die mathematische Beschreibung der Ausbreitung der Signale verwendet, nicht gangbar war. Nachteile der eingesetzten Methode sind zum einen der Aufwand, die Referenzkarte zu erstellen und

zum anderen die Anfälligkeit der Messungen der Referenzpunkte gegenüber Veränderungen der Umgebung.

Bestimmungsphase Ein Nutzer, der seinen aktuellen Ort wissen möchte, sendet den aktuell gemessenen Signalstärken-Fingerprint Φ von seinem auf einem mobilen Endgerät laufenden Clientprogramm an einen Web Service. Der Web Service liefert die zu dem Fingerprint gehörende Lokationsinformation zurück. Jeder mobile Client m ist dabei durch

$$m = \{(A'_k, \hat{\Phi}_k) \mid k \in \mathbb{N}, \quad 1 \leq k \leq K\}$$

beschrieben.

Der Web Service berechnet die Distanzen zu den Referenzpunkten unter Verwendung von Algorithmus 9.3. Aus der Distanzmenge D, gegeben durch

$$D = \{d_n \mid n \in \mathbb{N}, \quad 1 \leq n \leq \min(j, k)\}$$

wird die kleinste Distanz gewählt. Der Ort, der mit diesem Referenzpunkt assoziiert ist, wird als aktueller Ort des Nutzers zurückgegeben.

Für die Suche in multidimensionalen Daten existiert im Umfeld von Datenbanken eine Reihe von effizienten Algorithmen, wie R-Tree [Gut84], X-Tree [BKK96] oder optimal k-nearest neighbor search [SK98]. Diese besitzen jedoch eine hohe Komplexität und sind für sehr große Datenmengen entworfen worden. Für den Einsatz im Ortungssystem wurde daher der einfachere Algorithmus, der k-nächsten Nachbarn ausgewählt.

Algorithmus 9.3 Closest Match

```
 1: procedure CLOSEST MATCH(r,A)                ▷ Find closest matches of m
 2:     i, j, n ← 1
 3:     repeat
 4:         if (A'_j < A_i) then
 5:             j ← j + 1
 6:         else if (A'_j == A_i) then
 7:             d_n ← DISTANCE(A'_j, A_i)
 8:             n ← n + 1
 9:             i ← i + 1
10:             j ← j + 1
11:         else if (A'_j > A_i) then
12:             i ← i + 1
13:         end if
14:         D[n] ← d_n
15:         SORT(D)
16:     until !END(i, j)
17: end procedure
```

Tabelle 9.1: Distanzen zu den Fingerprints in den Räumen

Sichtbare APs	**Distanzen zu Fingerprints**									
Raum (8F,8A,35,45)	343	344	345a	345b	347	348	352	352a	353	auf
343 (-83,-51,-60,-)	**230**	325	534	889	-	675	-	-	-	-
344 (-84,-51,-69,-)	**138**	211	324	563	-	1021	-	-	-	-
345a(-77,-64,-74,-)	29	45	**11**	136	-	986	-	-	-	-
345b(-69,-67,-79,-)	227	114	57	**49**	-	1845	-	-	-	-
347 (-38,-88,-,-86)	-	-	-	-	**108**	-	-	-	-	-
348 (-84,-76,-49,-)	653	906	969	1538	-	**36**	-	-	-	-
352 (-74,-,-,-)	36	**1**	4	9	676	144	16	4	144	9
352a(-76,-,-,-)	16	9	**0**	25	784	100	4	**0**	196	1
353 (-66,-,-,-)	196	49	100	25	484	400	144	100	**16**	121
Auf (-74,-,-,-86)	-	-	-	-	580	-	-	-	-	**10**

In Algorithmus 9.3 wird die Methode DISTANCE aufgerufen. Zwei verschieden Distanzmaße wurden als Methode implementiert. Die Euklid'sche Distanz d_e wird als

$$d_e = \sqrt{\sum_{i=1}^{n}(\Phi_i - \hat{\Phi}_i)^2} \tag{9.5}$$

berechnet. Für die Implementierung wurde eine modifizierte Metrik verwendet, da diese durch die Nicht-Ausführung der Wurzeloperation schneller zu berechnen ist, als die Distanz in Gleichung (9.5). Das zweite eingesetzte Distanzmaß ist die Summe der absoluten Differenzen d_a (auch als "Manhattan Distanz" bezeichnet [CLR90]). Es bestimmt sich zu

$$d_a = \sum_{i=1}^{n}|\Phi_i - \hat{\Phi}_i| \tag{9.6}$$

9.2.2.4 Ergebnisse

Die Ergebnisse eines realen Versuchs der Bestimmung der Signalstärken-Fingerprints mit dem entwickelten Messclient und einem Vergleich zu den charakteristischen Fingerprints der Räume ist in Tabelle 9.1 aufgeführt. Die Tabelle zeigt die sichtbaren Access Points und ihrer Signalstärke (in dB) für jeden Raum. Die Distanzen, die nach Gleichung (9.6) zwischen dem gemessenen und dem gespeicherten Fingerprint berechnet worden sind, sind aufgetragen. Die jeweils niedrigste Distanz ist fett gedruckt.

Wie man Tabelle 9.1 entnehmen kann, besteht eine hohe Übereinstimmung zwischen der niedrigsten Distanz zu dem aktuell gemessenen Fingerprint Φ und dem jeweiligen Fingerprint des Raumes $\hat{\Phi}$. Zu Bestimmungen des falschen Raumes kommt es insbesondere in den Fällen, in dem nur ein einziger Access Point sichtbar ist, wie z. B. in den Räumen 352 und 352a. Im letztgenannten Raum kommt es zu zwei Minima, so dass meist mit einer weiteren Messung eine Entscheidung getroffen werden kann.

Tabelle 9.2: Ergebnisse der Erkennung der Räume

Raum	Richtig	Falsch	Unsicher	Bemerkung
343	9	3	1	
344	3	4	-	Die Raumcharakteristiken machen eine genau Erkennung schwierig
345a	7	2	-	
345b	1	7	-	Die Raumcharakteristiken machen eine genau Erkennung schwierig
347	9	0	-	
348	9	0	-	
352	2	4	3	Nur ein Access Point ist sichtbar. Hohe Unsicherheit bei der Entscheidung
352a	0	4	3	Nur ein Access Point ist sichtbar. Hohe Unsicherheit bei der Entscheidung
353	7	1	1	
Aufzug	5	2	1	
Gesamt	52	27	9	59,09% Erkennungsrate

9.2.2.5 Bewertung des WLAN-basierten Lokationsverfahrens

Der Einsatz des beschriebenen Ortungsystems in einer realen Umgebung führt zu dem Ergebnis, dass die Genauigkeit bei einer Erkennung des Raums, in dem sich das Endgerät gegenwärtig befindet, im akzeptablen Bereich liegt. Die Erkennungsrate hängt in starkem Maße von der Anzahl der sichtbaren Basisstationen ab. Tabelle 9.2 listet die Messungen und die Erkennungswahrscheinlichkeiten für eine Vielzahl an Testläufen auf. Die mittlere Genauigkeit des Systems liegt bei ca. 60%, wie dies auch in Tabelle 9.2 zu sehen ist. Sind jedoch 3 oder mehr Basisstationen sichtbar, so liegt die Erkennungsrate bei ungefähr 85%. Die starke Schwankung der Rate entspricht den Ergebnissen über die Anzahl an notwendigen Basisstationen für eine gute Abdeckung, wie sie in [CK02b] beschrieben sind.

Die Verteilung der Basisstationen in der Infrastruktur wurde nicht den Anforderungen des Ortungssytems angepasst, vielmehr wurde das real existierende Funknetzwerk für einen weiteren Dienst – die Ortsbestimmung – verwendet. Diese Randbedingung ist zum einen ein Vorteil dieser Methode, da sie eine vorhandene Infrastruktur ohne die Installation weiterer Hardware nutzt. Die Nachteile liegen darin, dass für eine höhere Genauigkeit die Positionen der Basisstationen entsprechend ausgerichtet werden müssten. Ein weiterer Nachteil dieses Ansatzes ist der Aufwand, eine Referenzkarte zu erzeugen sowie die Sensibilität gegenüber Veränderungen der Umgebung. Insgesamt liefert das vorgeschlagene System akzeptable Informationen für die weitere Verwendung von Lokation als Kontextmerkmal.

9.3 Fuzzy Logik Applikation zur Kontextaggregation

Fuzzy Logik als Inferenzverfahren wurde in Abschnitt 6.5 vorgestellt. Das Verfahren zeichnet sich durch die Möglichkeit aus, intuitiv und dennoch formal Regeln zu ersstellen. Dies wurde bei der Verwendung von Fuzzy Logik Regeln als Methode zur Aggregation von Kontextmerkmalen zu Kontexten genutzt. Nachfolgend wird die praktische Umsetzung des Verfahrens beschrieben.

9.3.1 Fuzzy Logik-Aggregationsapplikation für Kontextinformationen

Eine auf Fuzzy Logik Regeln basierende Aggregationskomponente wurde als eine neuartige Möglichkeit zur Kombination von Kontextinformationen entworfen, implementiert und evaluiert.

Zur Erprobung des Konzeptes wurden sieben, als repräsentativ erachtete, Kontextinformationen ausgewählt, die wiederum zur Bestimmung von fünf verschiedenen Kontexten ausgewertet werden. Als Eingabevariablen wurden ausgewählt: Tag der Woche (*Days of the week*), Datum/Zeit (*Calendar entry*), Helligkeit (*Light*), Keyboardaktivität oder Mausbewegung (*Typing or Mouse Activity*), Anzahl der umgebenden Personen (*Presence of Surrounding People*), Bewegung (*Movement*) und Lokationsinformation (*Location*). Folgende Kontexte sollten durch die Aggregation der Kontextinformationen erkannt werden: *Freizeit, unterwegs, Besprechung, Arbeitsplatz, Pause*.

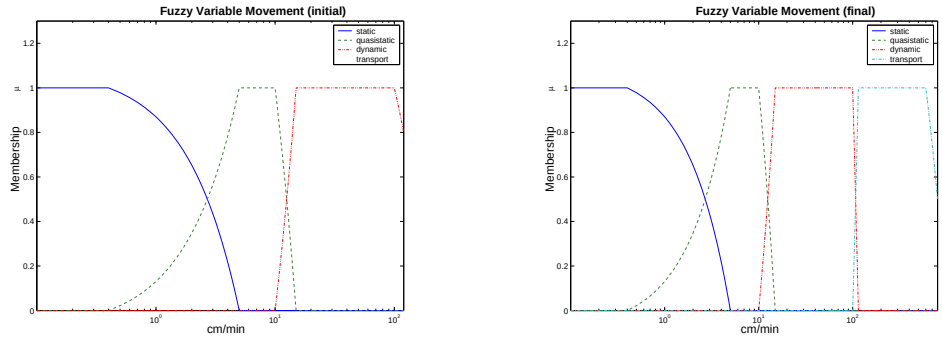
Jede Eingangsvariable wurde als linguistische Variable $\mathcal{L}_{1...11}$ modelliert und in einem ersten Schritt wurden realistische Wertebereiche (*universe of disclosure*) bestimmt. Die einzelnen Bereiche wurden dann in linguistische Terme unterteilt. Die entsprechenden Zugehörigkeitsfunktionen $\mu_{1...11}$ sind in den Abbildungen B.1 - B.7 im Anhang dargestellt. Zur Beschreibung der Kurven stehen im FuzzyJ-Toolkit vordefinierte Funktionen zur Verfügung. Insbesondere die trapezoide Form wird häufig verwendet und wird durch ihre vier charakteristischen Punkte definiert. Zur näheren Spezifikation der Terme werden *unscharfe Modifikatoren* (*FuzzyModifier*) wie *very*, *slightly* oder *somewhat* bereit gestellt.

Aus den linguistischen Termen werden Fuzzy Regeln erstellt, die anschließend ausgewertet werden. Das Toolkit stellt hierbei eine Reihe von Implikationsoperatoren zur Verfügung, von denen üblicherweise der *Mamdani Mininference Operator* [MA75] eingesetzt wird. Insgesamt wurden elf Regeln erstellt, ausgewertet und auf Übereinstimmung überprüft.

Die Aufteilung der Variablen auf die Terme sowie die Aufstellung der Regeln stellte im Vergleich zur Implementierung der Applikation den größten zeitlichen Aufwand dar [For04]. Die Verwendung des FuzzyJ-Toolkits erleichtert die technische Erstellung der Regeln. Die Identifikation der zu betrachtenden Kontextmerkmale und ihrer Modellierung bestimmt jedoch die Güte der Regelbasis und erfordert Aufwand. Die iterativ durchlaufene Designphase benötigt das Wissen eines Experten, der die zu detektierenden Kontexte mittels linguistischer Ausdrücke beschreiben und die Ergebnisse des

Fuzzy Logik Systems überprüfen kann. Um die Güte des entwickelten Systems zu testen, wurden mehrere Testläufe durchgeführt.

In einem ersten Schritt wurden charakteristische Werte gewählt, um zu prüfen, ob der zu bestimmende Kontext auch erkannt wird. Anschließend wurden Änderungen an den linguistischen Termen vorgenommen. Nach dem Positiv-Test wurde getestet, ob der Ergebniswert bei untypischen Eingaben nahe Null bleibt. Die Ergebnisse dieser Negativ-Tests wurden ebenfalls verwendet, um das Regelsystem zu kalibrieren. Dieser Prozess kann in mehreren Iterationsstufen durchgeführt werden. So wurde beispielsweise die Variable *Movement* von anfangs vier Termen, wie in Abbildung 9.7(a) dargestellt, auf sechs Terme, wie in Abbildung 9.7(b) abgebildet, erweitert, da im ersten Fall mehrere Regeln aktiviert wurden und das Ergebnis nicht eindeutig war. Eine ausführliche Darstellung der Testergebnisse mitsamt der gewählten Eingangsvariablen und Resultate in tabellarischer Form findet sich in Anhang B.



(a) Initiale Fuzzy Variable für Bewegungsmuster

(b) Erweiterte Fuzzy Variable für Bewegungsmuster

Abbildung 9.7: Fuzzy Variable für Bewegungsmuster

Insgesamt lässt sich die Verwendung eines Fuzzy Logik-Systems zur Kontextaggregation als positiv bewerten. Die Erkennungsrate in dem ausgewählten Beispiel war sehr zufriedenstellend. Ein Vorteil gegenüber anderen Verfahren wie z. B. dem Dempster-Shafer-Reasoning, die ebenfalls Unsicherheiten der Eingangswerte modellieren und zur Aggregation eingesetzt werden [Wu03] können, liegt in der intuitiven Nutzung und adaptiven Modellierung der Kontextinformationen. Unterschiedliche Zugehörigkeitsfunktionen können verwendet werden, wobei die Verwendung von Singletons insbesondere zur Abbildung von symbolischen Werten, wie Lokationen (z. B. Raumnummern) geeignet ist, was mit anderen Verfahren nicht oder nicht einfach möglich ist.

Die Ausführungszeiten des Java-Programms sind schnell genug, um in einem echten Szenario eingesetzt zu werden. Die Performanzbetrachtung ist ausführlich in Anhang B.4 dargestellt. Die Zeiten für die Auswertung der Fuzzy Regeln beträgt im Durchschnitt $\bar{t} \approx 0,7$ ms Die hier beschriebene Fuzzy Logik Anwendung wurde sowohl aus eigenständige Applikation als auch als Aggregationsoperator realisiert. Diese kann dann in

dem in Abschnitt 7.2 beschriebenen Kontextaggregationsnetz (Context Aggregation Network) eingesetzt werden.

Eine Verbesserung des vorgeschlagenen Konzepts kann durch eine Vereinfachung der Entwurfsphase erzielt werden. So kann durch den Einsatz eines graphischen Editors, wie er von kommerziellen Fuzzy Logik Entwicklungsumgebungen angeboten wird, die Erstellung der linguistischen Terme und der Fuzzy Regeln erleichtert werden. Jeder Nutzer des Systems könnte eine eigene Aufteilung der linguistischen Variablen in einer intuitiven Weise spezifizieren. Weitere Vorteile ergeben sich durch die Verknüpfung von Ansätzen der Neuronalen Netze mit dem Fuzzy-Ansatz. Arbeiten hierzu sind unter dem Begriff *Neuro-Fuzzy* [HG94, Vuo95, NKK97, Nel99] erschienen.

9.4 Kontext-Server — ContextServer

Der interne Aufbau des ContextServers ist in Abbildung 9.8 aufgezeigt. Der ContextServer besteht aus drei Hauptkomponenten. Eine *Web Service Schnittstelle* stellt öffentliche Funktionen anderen Anwendungen zur Verfügung. Die *Funktionalitätskomponente* realisiert die angebotenen Funktionalitäten und interagiert dabei mit einer *Datenbank*. Diese dient zur konsistenten Speicherung und zum gezielten Auffinden (*retrieval*) der Kontextdaten.

Der ContextServer basiert auf einem Apache Tomcat/Axis Applicationserver [www1], der Web Services vorhalten und ausführen kann. Die einzelnen Dienste sind mittels einer WSDL-Beschreibung spezifiziert. Andere Anwendungen können aus dieser Beschreibung die notwendigen Stubs- und Skeleton-Dateien für den entfernten Methodenaufruf automatisch generieren lassen. Neben den öffentlich bereitgestellten Methoden sind private interne Methoden für die Funktionalität zuständig. Die Methoden werden auch auf dem Applicationsserver ausgeführt.

Die Funktionskomponente interagiert direkt mit der Datenbank. Als Datenbank wurde die freie relationale MySQL Datenbank [www14] gewählt. Diese liefert eine ausreichend hohe Performanz und Stabilität. Über die Einbindung des MySQL Connector/J, der einen JDBC-Treiber (JAVA Database Connector (JDBC)) für MySQL zur Verfügung stellt, werden die Interaktion zwischen den Web Services und der Datenbank realisiert. Die Details der Datenbank und der Zugriff auf die Daten wurde durch einen Web Service gekapselt. Fremde Applikationen können mit der Datenbank nur über die bereitgestellten Methoden kommunizieren.

9.4.1 Architektur

Nachfolgend sollen die Architektur und die relevanten Komponenten des ContextServers, wie sie in Abbildung 9.8 abgebildet sind, beschrieben werden. Die drei Hauptkomponenten des Servers sind die *Schnittstellen*, die *Datenbank* und die *Web Services*. Die Schnittstellen stellen einen standardisierten Zugriff auf die Web Services zur Verfügung. Die Web Services bieten spezifische Funktionalitäten, wie eine lokale Aggregation, eine

Historie an Kontexten und den aktuellen Kontext an. Dafür greifen sie auf die interne Datenbank zurück, die die Speicherung von Kontexten und den dazugehörigen Daten zur Verfügung stellt.

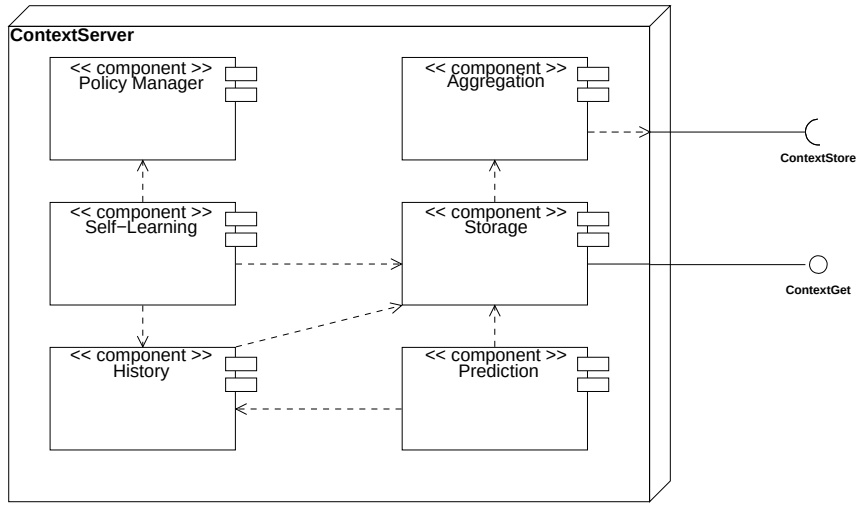


Abbildung 9.8: Aufbau der Architektur des ContextServers

9.4.2 Funktionsbeschreibung

Die entwickelten Web Services sind jeweils über eine WSDL-Beschreibung spezifiziert. Über eine Einbindung in ein öffentliches Register wie UDDI kann der Dienst offeriert (*announce*) und von anderen Applikationen gefunden werden. Zwei generelle Methoden, die von den Web Services bereitgestellt werden, sind das Speichern und das Beziehen von Kontexten. Hierzu stehen die in Unterabschnitt 7.4.3.1 beschriebenen Kommunikationstypen zur Verfügung.

Da es sich bei Kontexten um sehr sensible Daten handelt, die geschützt werden sollten, findet vor der eigentlichen Kommunikation eine Authentifizerungs- und Authorisierungsphase statt, die sicherstellen soll, dass nur authorisierte Anwendungen Daten schreiben und beziehen können. Zusätzlich können die Daten auch öffentlich zugänglich gemacht werden. Die Verwendung eines hierarchischen Formats erlaubt es, auch eine gruppenspezifische Informationsverteilung zu realisieren. Das Schreiben von Daten kann in ähnlicher Form realisiert werden. Eine geeignete Richtlinienkomponente (*policy component*) kann auch hier die Prozesse überwachen. Die Kommunikation zwischen den Anwendungen und dem ContextServer kann optional verschlüsselt werden.

Zusätzlich zum Anfrage/Antwort-Prozess, wie er in Abbildung 7.12(b) dargestellt ist, können Anwendungen auch einen Subscribe/Publish-Mechanismus wählen. Ein solcher Mechanismus ist nicht in den standardisierten Kommunikationsformen von SOAP definiert, kann aber durch die Kombination der Kommunikationsformen One-Way (Ab-

bildung 7.12(a)) und Notification (Abbildung 7.12(d)) realisiert werden. Die Vorteile letzterer Methode wurden bereits erörtert. Die Anwendungen werden asynchron benachrichtigt, wenn sich ein Kontext, für den sie sich subskribiert haben, geändert hat. Neben dem aktuellen Kontext einer bestimmten Entität (Nutzer, Ort, Artefakt) kann auch auf ältere Kontexte zurückgegriffen werden.

Neben den beiden wesentlichen Web Services wurden weitere Funktionalitäten durch interne Web Services realisiert. Diese bieten lokale Datenfusionsmechanismen und können zur Erstellung von *Kontextvorhersagen* eingesetzt werden. Ansätze bieten Verfahren aus den Sprungvorhersagen in Prozessoren [PBTU03, PBT+04] oder aus der Betrachtung aller erfassbaren Aspekte und der Erkennung von Mustern und Zusammenhängen im Benutzerverhalten [MRF03, May04]. Alternativ kann der Kontext aus Bewertung der Kontexthistorie, Bewegungsmustern (*motion pattern*) und Kalenderinformationen vorhergesagt werden [GAS+03].

Zur Speicherung der Daten wurde eine MySQL-Datenbank [www14] gewählt. Sie erfüllt die geforderten Eigenschaften und ist frei verfügbar. Aus Performanzgründen wurde keine XML-Datenbank wie Xindice [www2] oder Tamino [www28] gewählt, obwohl die Verwendung des XML-Formats PIDF-CE dies nahelegen würden. Für große Datenmengen jedoch skaliert der XML-Datenbank-Ansatz nicht mehr, da immer der gesamte XML-Baum traversiert werden muss und eine gezielte Indizierung nur unzureichend realisiert werden kann.

9.5 Erweiterte SIP Entitäten

Eine Nutzungskomponente von Kontexten ist das Telefonie-Endgerät UA. Die prinzipielle Funktionsweise des User Agents wurde in Unterabschnitt 2.2.2.3 beschrieben. Der UA stellt einen Satz an Kommunikationsmehrwertdiensten zur Verfügung. Im Rahmen der Arbeit wurden zwei dieser Dienste derart erweitert, dass sie Kontexte in ihren Ablauf mit einbeziehen. Diese Dienste sind der *Kontext-bewusster Rückruf* (*Context-aware Call Completion*) und die *Kontext-bewusste Anrufweiterleitung* (*Context-aware Call Diversion*). Desweiteren wurde der UA um die Möglichkeit der *In-band Verteilung von Kontexten*, wie es konzeptionell in Unterabschnitt 3.3.2 beschrieben wurde, erweitert.

9.5.1 Erweiterter SIP User Agent XUA

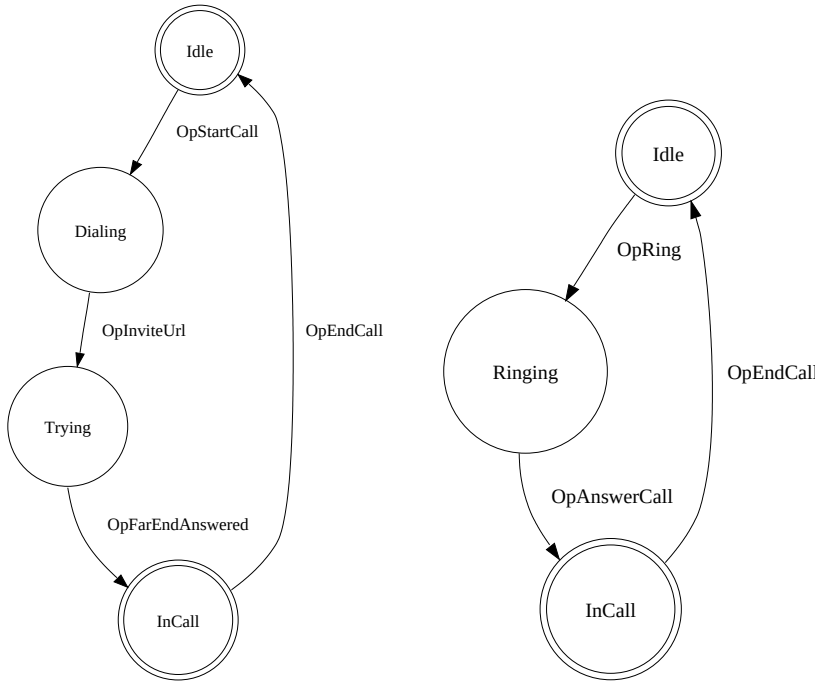
Als User Agent wurde der im VOCAL vorhandene UA als Basis für die Erweiterungen ausgewählt. Dieser ist in der verwendeten Version vollständig kompatibel zum SIP-Standard wie er in dem mittlerweile abgelösten RFC 2543 [HSSR99] beschrieben ist. Die Erweiterungen lassen sich jedoch ohne Einschränkung auch auf einen zu RFC 3261 [RSC+02] kompatiblen UA anwenden. Die Details der Erweiterung sind nachfolgend erläutert.

9.5.1.1 Funktionsbeschreibung des Prototypen

Der UA verfügt nicht über ein Graphical User Interface (GUI), sondern nur über eine Kommandozeilenschnittstelle, über die der Nutzer mittels Tasteneingabe das Programm steuern kann. Ein graphisches Frontend zur Steuerung des UA wurde mittels Tcl/Tk [www29] und Expect [Lib97][www7] entwickelt. Der erweiterte User Agent wird im nachfolgenden als XUA bezeichnet.

Bei der Erweiterung des User Agents wurden bereits implementierte Mehrwertdienste um neue Funktionalitäten ergänzt. Hierzu wurde die interne Ablauflogik des jeweiligen Dienstes entsprechend verändert. Dazu wurde die interne Struktur des User Agents mittels softwaretechnischer Werkzeuge wie Doxygen [www6] (zur Visualisierung der Klassenstruktur) analysiert, um die Ansatzpunkte zu identifizieren. Der VOCAL UA ist in Form eines endlichen Zustandsautomaten (FSM) realisiert. Eine (fast) vollständige graphische Darstellung der FSM des UA ist in Abbildung F.4 im Anhang dargestellt.

Zwei relevante Ausschnitte aus der FSM sind die Zustände bei einem Anruf sowohl aus der Sicht des Anrufers (Abbildung 9.9(a)) als auch auf der Angerufenenseite (Abbildung 9.9(b)). Diese beiden FSM-Teile wurden gewählt, da an ihnen die Erweiterungen deutlich gemacht werden.



(a) Zustandsmaschine des UA als Anrufer

(b) Zustandsmaschine des UA als Angerufener

Abbildung 9.9: Zustandsmaschinen des UA für den Basisruf

Von jedem Zustand (*state*) führen eine Reihe von Transitionen zu den nächsten Zuständen. Bei beim Eintreffen einer definierten Bedingung wird die Operation der jeweiligen Transition ausgeführt. In VOCAL werden die Transitionen mit „Operator“ bezeichnet. Es wird jeweils der Operator ausgewählt, der für die Bedingung definiert ist, bei der sich die erste Übereinstimmung ergibt. Eigene Operatoren müssen in einer eigenen Klasse dem Prozess, der die FSM während des Programmstarts aufgebaut hat, bekannt gegeben werden. Danach können die Operatoren in allen Zuständen aufgerufen werden.

9.5.1.2 Erweiterung des User Agents um Kontext-bewusstes Anrufweiterleiten

Bei diesem Dienst spezifiziert der Nutzer die Weiterleitung eines eingehenden Anrufs in Abhängigkeit von einem bestimmten Kontext. Die Weiterleitung erfolgt ohne weitere Interaktion mit dem Nutzer, was insbesondere bei Weiterleitungen während eines Gespräches oder bei Abwesenheit eine sehr nützliche Funktion ist. Der im VOCAL-System enthaltene User Agent bietet den Dienst *Anrufweiterleitung* nicht an.

Der im Rahmen dieser Arbeit erweiterte User Agent XUA wurde um die Basisfunktionalität, wie sie beispielsweise für H.323 in H.450.3 [Int98b] als *Call Diversion – CD* definiert ist, erweitert. Zusätzlich wurde der Kommunikationsdienst *Kontext-abhängige Anrufweiterleitung* bzw. *Context-aware Call Diversion* konzipiert und implementiert. Die notwendige Erweiterung zur Eingabe des Kontexts und der Zieladresse (SIP-URL oder E.164-Nummer [Int97b]) ist in Abbildung 9.10 dargestellt. Die Erweiterung wird aus dem Zustand *Idle* durch Drücken der Taste 'd' aufgerufen und führt den Nutzer durch ein Kommandozeilenmenü. Alternativ kann die Parametrisierung durch eine graphische Oberfläche erfolgen.

Die Parametrisierung des Dienstes wird in einer parallel zum XUA laufenden Komponente namens *ServiceManager* gespeichert. Nach der Initialisierung ist der Dienst so lange aktiv, bis der Nutzer den Dienst bzw. einzelne Parametrisierungen wieder deaktiviert. Der aktuelle Kontext des Nutzers wird in der FSM lokal vorgehalten, um den Zeitbedingungen beim Gesprächsaufbau Rechnung tragen zu können. Für die Interaktion mit dem ContextServer ist ein asynchroner *Context Manager*, ähnlich dem Subscription Manager, vorgesehen. Der aktuelle Kontext kann von diesem über internen Methodenaufruf aus der FSM aufgerufen werden. Dazu wird eine Subskription zu einer vordefinierten Kontextdatenbank bzw. einem ContextServer verwaltet. Ändert sich der Kontext, so wird diese Änderung über einen asynchronen Ereignismechanimus mitgeteilt.

Kommt ein Anruf bei dem XUA an, so wird im Zustand *Idle* bzw. im Zustand *Incall* geprüft, ob ein Weiterleitungsdienst aktiv ist. Ist dem der Fall, so werden die Bedingungen geprüft. Auf Basisbedingungen wie *Busy* oder *No-Reply* kann mit diesem Verfahren genauso geprüft werden, wie auch auf einen Kontextbezeichner. Hierbei wird auf das Enthaltensein der Bedingungszeichenkette in dem aktuellen Kontext geprüft. Kommt es zu einer Übereinstimmung, wird der Anruf an die spezifizierte Adresse weitergeleitet. Dazu sendet der XUA eine 302 Moved temporarily-Nachricht, welche die Zieladresse im `Contact`-Headerfeld enthält, an den Anrufer zurück.

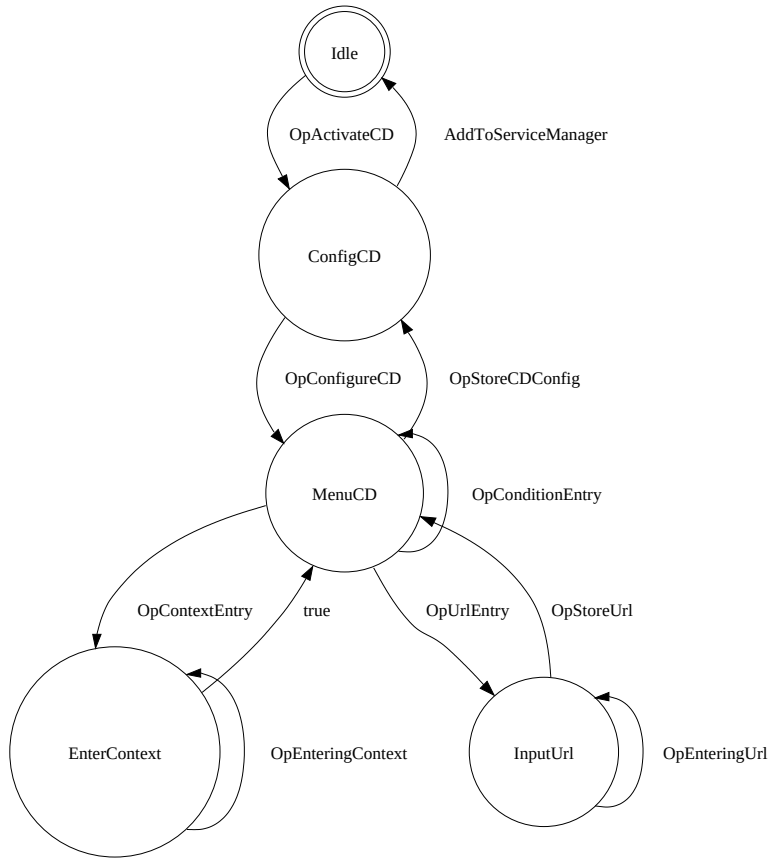


Abbildung 9.10: FSM zur Eingabe der Kontexte und der Zieladresse

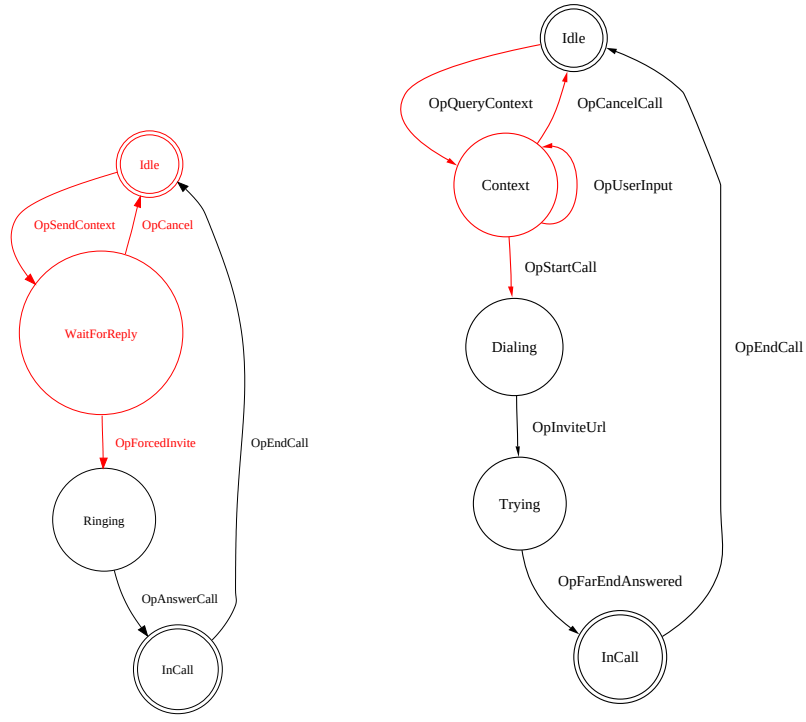
9.5.1.3 Erweiterung des User Agents zur Kontext-Verteilung

Das Konzept zur Verteilung von Kontexten ist in Unterabschnitt 3.3.2 vorgestellt worden. Der Kontext wird vom Angerufenen an den Anrufer nach einer Rufanfrage übermittelt, ohne dem Angerufenen bereits den Ruf zu signalisieren. Zwei generelle Arten zur Mitteilung von Kontexten an Anrufer werden unterschieden: Zum einen ist hier die Verteilung im *Peer-to-Peer*-Modus zu nennen, bei der Kontexte direkt zwischen zwei User Agents ausgetauscht werden. Zum anderen gibt es die Verteilung durch einen CPL-Server, was durch eine Erweiterung, die in Unterabschnitt 9.5.2 beschrieben wird, möglich ist.

Die Signalisierung für die Abfrage der Kontexte wie auch der Transport der Kontexte soll In-band mit SIP-Methoden erfolgen. Verschiedene Alternativen für eine In-band Signalisierung wurden in Unterabschnitt 7.4.4 identifiziert und bewertet. Die Verwendung der INVITE-Nachricht zur impliziten Verteilung von Kontexten und die dafür notwendigen Erweiterung der FSM des XUA werden nachfolgend beschrieben. Darüberhinaus können zur expliziten Abfrage des Kontexts *Subskriptionen* und *Notifikationen* in Form der SIP-Methoden SUBSCRIBE und NOTIFY verwendet werden. Es wurde hierfür

die *SubscriptionManager*-Komponente erweitert, welche Subskriptionen mit dem neuen *Event-Package* "context-request" verwaltet.

Bei der impliziten Verteilung startet der Anrufer einen Standard-Verbindungsaufbau. Der Empfänger, bei dem die Kontextverteilung aktiviert ist, sendet den Kontext zurück. In der Zustandsmaschine des Angerufenen, die in Abbildung 9.11(a) dargestellt ist, wurde daher der Zustand *Idle* derart verändert, dass auf eine eingehende INVITE-Nachricht der Prozess der Kontextverteilung aktiviert wird. Optional kann an dieser Stelle eine SIP-konforme Authentifizierung erfolgen, um den Sender der Nachricht eindeutig zu identifizieren. Der Kontext wird lokal aus dem bereits kurz beschriebenen *ContextManager* bezogen.



(a) Zustandsmaschine des Angerufenen

(b) Zustandsmaschine des Anrufers

Abbildung 9.11: Zustandsmaschine des UA für die Abfrage des Kontexts

Nachdem der Kontext der FSM zur Verfügung steht, wird im selben Zustand eine 183 Session progress-Nachricht erzeugt, welche den Kontext enthält. Diese wird an den Anrufer zurückgesendet. Anschließend wartet die FSM bis zum Ablauf eines Timers auf die weiteren Nachrichten des Anrufers. Der *Idle*-Zustand des Anrufers würde normalerweise diese Nachricht ignorieren, da es sich um eine informative (*provisional*) Response handelt. Um den Inhalt dem Nutzer anzeigen zu können und weitere Aktionen zu initieren wurde der Zustand modifiziert. Der relevante Ausschnitt der FSM des Anrufers ist in Abbildung 9.11(b) im Anhang dargestellt.

Die Ankunft der 183 Session progress-Nachricht löst den Übergang in den neuen Zustand *DisplayContext* aus. In diesem wird dem Nutzer der übertragene Kontext angezeigt. Anschließend kann der offene Anrufaufbau fortgesetzt oder abgebrochen werden. Der Nutzer kann für diese Entscheidung den Kontext des Angerufenen einbeziehen. Die Nutzerinteraktion erfolgt dabei über Tastendruck bzw. mittels der Auswahl eines Menüs in der GUI. Wird der Aufbau abgebrochen, so wird eine CANCEL-Nachricht generiert und die Zustandsmaschinen des Anrufers als auch des Angerufenen kehren wieder in ihren Ausgangszustand zurück.

Soll der Anrufaufbau vollendet werden, so wird vom Anrufer eine re-INVITE-Nachricht mit der gleichen `Session-Id` erzeugt und versandt. Der Angerufene kann anhand der Tatsache, dass es sich um eine re-INVITE-Nachricht handelt, den Bezug zur offenen Sitzung herstellen und mit dem Standardverfahren für den Sitzungsaufbau, wie er in Abbildung F.2 abgebildet ist, fortfahren. Erst jetzt wird der Angerufene über den eingehenden Anruf benachrichtigt.

9.5.1.4 Fazit

Die durchgeführten Erweiterungen wurden am User Agent des Open Source Projektes VOVIDA [www34] vorgenommen. Die gewonnenen Erfahrungen hinsichtlich der Umsetzung der Ansätze für Kontext-bewusste Kommunikationsdienste beziehen sich daher auf diesen speziellen UA. Prinzipielle Aussagen über die Komplexität und den Arbeitsaufwand der Erweiterungen können dennoch getroffen und auf andere Clients übertragen werden.

Der stringente Aufbau des VOVIDA UA als FSM bietet jedoch Vorteile bei der Erweiterung der internen Logik, da sich diese gut nachvollziehen lässt. Neue Funktionen müssen in Form von Zuständen bzw. Operatoren hinzugefügt werden. Die jeweiligen Zustände, von denen die neuen Operatoren ausgeführt werden sollen, können schnell identifiziert werden. Der Implementierungsaufwand für jedes weitere Feature ist wesentlich kleiner als für die erste Erweiterung, so dass der Aufbau als durchdacht und effektiv bezeichnet werden kann.

Ein großes Manko des VOVIDA UA ist seine nicht existierende Oberfläche und seine sehr spartanische Nutzerinteraktionsmöglichkeiten. Das Kommandozeilen-Menü zur Eingabe der Parameter für den Kontext-bewussten Anrufweiterleiten-Dienst ist nicht sehr nutzerfreundlich. Eine mittels Tcl/Tk und Expect entwickelte GUI kann zur Bedienung des UA verwendet werden. Das Programm gibt hierbei über Expect die Kommandozeilenbefehle an den UA weiter.

Die realisierten Erweiterungen erfüllen die prinzipiellen Funktionalitäten zur Umsetzung der in Unterabschnitt 3.3.1 und Unterabschnitt 3.3.2 beschriebenen Ansätze. Die Implementierung erfolgte in der Form einer prototypischen Umsetzung. Das Anwachsen der Codegröße des UA um knapp 2% ist bei der Betrachtung hinsichtlich des, insbesondere beim Einsatz auf einem PDA, kritischen Speicherbedarfs zu vernachlässigen.

9.5.2 Erweiterte CPL-Verarbeitungseinheit

Die Erweiterung der CPL-Sprachsyntax muss eine entsprechende Umsetzung in einer CPL-Verarbeitungseinheit (*CPL-Engine*) erfahren, um die gewünschten Funktionalitäten zu realisieren. Dazu wurde die *CPL Engine* des *Feature Server* (FS) aus dem VOCAL-Projekt als Ausgangsbasis gewählt. Das VOCAL-System besteht aus einer Reihe von Servern, die unterschiedliche Funktionalitäten von SIP-Entitäten erfüllen.

9.5.2.1 Prinzipieller Aufruf von CPL-Diensten

Für die Umsetzung von CPL-Skripten interagiert der Feature Server (FS) mit dem *Provisioning Server* (PS), wie dies in Abbildung 9.12 dargestellt ist. Der PS hält die jeweiligen Dienste – hier *Features* genannt – für jeden Nutzer bereit. Bei der Aktivierung eines Features für einen Teilnehmer, werden Port und CPL-Skript des Features im Provisioning Server gespeichert. Bei einem Anruf werden die gespeicherten Informationen des Nutzers aus dem Provisioning Server geladen und die Aktionen der aktiven Features werden an den entsprechenden Ports aufgerufen.

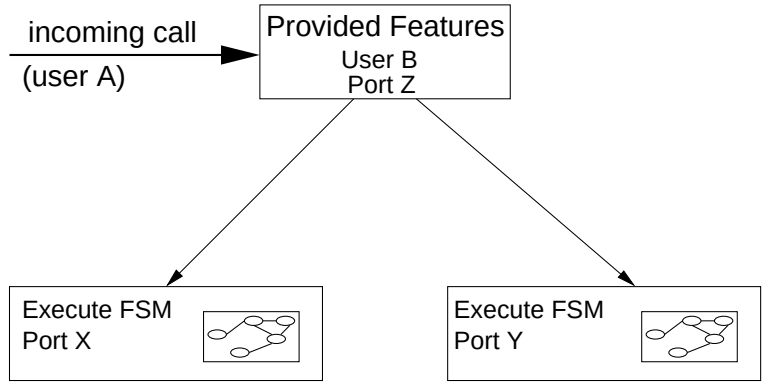


Abbildung 9.12: Ausführen einer FSM in einem Feature Server

9.5.2.2 Umsetzung der Aufrufe von CPL-Diensten

Beim Start der CPL-Engine wird die CPL-DTD-Datei eingelesen und für jedes CPL-Element wird eine entsprechende C++-Objektrepräsentation initialisiert. Für jedes neu eingeführte CPL-Element werden die jeweiligen C++-Objekte erzeugt. Bei dem ersten Aufruf eines jeden Features wird das korrespondierende CPL-Skript vom CPL-Interpreter benutzt, um eine interne Finite State Machine aufzubauen. Die CPL-DTD wird dabei für das Überführen der CPL-Elemente in interne Objekte genutzt. In Abbildung 9.13 ist dieser Prozess graphisch dargestellt.

Die erstellten FSMs führen anschließend die entsprechenden Aktionen durch. Bei späteren Aufrufen eines bereits initialisierten Features werden nur noch entsprechende

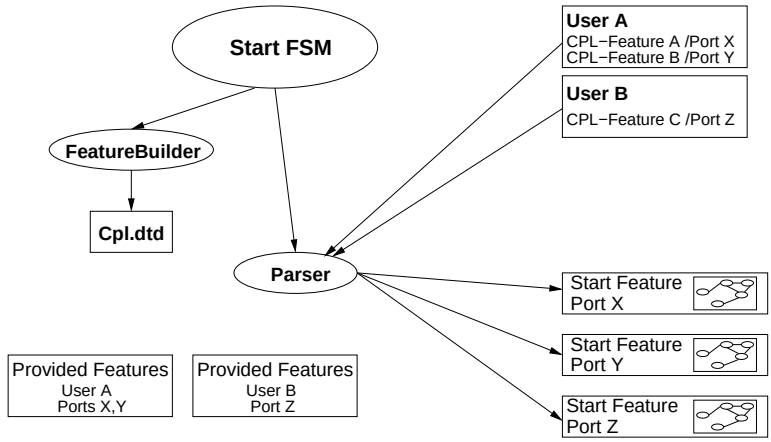


Abbildung 9.13: Verarbeitung eines CPL Skriptes

FSM aus dem Speicher geladen und ausgeführt. Zwei Arten von Ereignissen, bei denen ein Feature ausgeführt wird, sind definiert. Eingehende Anrufe werden mit den Tags `<incoming> ... </incoming>` in der CPL-Syntax gekennzeichnet. Ausgehende Anrufe werden entsprechend mit `<outgoing> ... </outgoing>` bezeichnet. Wenn ein eingehender oder ein ausgehender Anruf beim Feature Server eintrifft, so wird die Liste der Features des Nutzers durch Angerufene bzw. Anrufer nach aktiven Features durchsucht. Wird ein aktives Feature gefunden, so wird die korrespondierende Finite State Machine (FSM) erstellt bzw. im Speicher ausgeführt und interpretiert.

Jeder Zustand der FSM besitzt eine Liste von Operatoren die in der Reihenfolge, wie sie in der Liste eingetragen wurden, ausgeführt werden, bis der erste Operator einen Nachfolgezustand oder einen Nachfolgeoperator als Ergebnis zurückgibt. Jeder Operator gibt also entweder einen Zustand zurück oder er kann wiederum eine Liste mit Operatoren als Nachfolger beinhalten, welche ebenfalls der Reihenfolge nach abgearbeitet werden. Um die Operationen zu terminieren, muss am Ende jeder Liste von Operatoren ein Operator “myEndOp” enthalten sein, der auf den Endzustand der FSM verweist und ausgeführt wird, falls kein anderer Operator mehr zur Ausführung bereit steht.

9.5.2.3 Neue CPL-Befehle

Der CPL-Engine wurden eine Reihe neuer Befehle und Elemente hinzugefügt. Die wichtigsten Elemente sind nachfolgend erläutert.

Context Lookup

Für die Verbindung zur Datenbank wurde die MySQL C++ Library verwendet, welche in MySQL enthalten ist. Das Objekt „Context“ wurde erstellt, um die Funktionalität zur Abfrage des ContextServer bereitzustellen. Aktuell wird hierzu die MySQL Library eingebunden (#include <sqlplus.hh>) und eine MySQL Abfrage durchgeführt. Alternativ wird die Abfrage über SOAP vorgeschlagen.

Um den ContextServer abzufragen, wird ein Objekt vom Typ „Context“ erstellt. Die Abfrage der SQL Datenbank findet in der Methode `get(..)` statt. Die notwendigen Parameter (Adresse des MySQL Server, MySQL User-Name, Passwort, ..) werden bei der Erstellung des CPL-Skriptes eingetragen. Der Name des Nutzers, dessen Kontext abgefragt werden soll, wird aus dem `To:`-Headerfeld der eingehenden SIP-Nachricht ausgelesen. Mit diesen Informationen wird anschließend ein SQL-Statement mit einer Suchanfrage erzeugt.

Um die Datenbank abzufragen, sind hauptsächlich folgende Befehle notwendig::

```
Connection con(use_exceptions);
//Verbindung aufbauen
con.connect(dbpwd.c_str(),dbhost.c_str(),dbuser.c_str());
//Datenbank auswaehlen
con.select_db(db);
//Query Objekt
Query query = con.query();
//Anfrage aufbauen
query << "select␣*␣from␣" << table << "␣where␣user␣=␣\""
             << user << "\"␣AND␣domain␣=\"" << host "\"";
//Anfrage absenden
Result res = query.store();
Row row;
Result::iterator i;
//Ergebnis auslesen
for (i = res.begin(); i != res.end(); i++) row = *i;
state = row["state"];
return state;
```

Das Ergebnis der Datenbankabfrage, d.h. der Kontextbezeichner, wird als Rückgabewert (*return value*) der Methode zurückgegeben. Der Bezeichner liegt anschließend als „string“ vor. Danach wird das Ergebnis ausgelesen und lokal gespeichert, um es anderen Komponenten zur Verfügung zu stellen.

Context Switch

Das Prinzip des *Context Switch* wurde in Unterabschnitt 8.2.1 dargelegt. Der Befehl gehört von seiner Struktur her zu den *CompoundOp*, den verzweigenden Operatoren. Die eigentliche Funktionalität des Operators befindet sich im Objekt „CPLOpContext“.

Der Kontext wird in der *Context Lookup*-Komponente abgefragt. Der Kontextbezeichner, der dem angerufenen Nutzer zugeordnet ist, wird ausgelesen und bewertet. Anhand des Inhaltes wird zwischen verschiedenen möglichen Nachfolgeoperatoren verzweigt, wobei die Zustände als Integer-Werte interpretiert werden:

```
switch(atoi(state)){
case 0: return myFailure->process(event); break;
case 1: return myBusy->process(event); break;
case 2: return myMeeting->process(event); break;
(..)
}
```

Context Notify

Ein neuer CPL Befehl „notify“ soll in der Lage sein, eine Nachricht an den User Agent zu verschicken, um diese dort dem Anrufer anzuzeigen. Der Inhalt der Nachricht soll beliebig über die Parameter des CPL-Befehls definiert werden können. Zudem soll es möglich sein, in einer solchen Nachricht auch Auskunft über den Kontext des Angerufenen zu geben.

Der neue Befehl mit der geforderten Funktionalität kann in der Form

```
<notify />
```

im CPL Skript verwendet werden. Die Nachricht, die an den User Agent des Anrufers geschickt wird, soll vom Typ NOTIFY sein. Um eine Nachricht diesen Typs zu versenden, wird eine neue Nachricht erstellt und das Ziel, sowie der Absender entsprechend verändert („myFrom“ wurde aus den Daten der Zieladresse und „myTo“ aus denen des Absenders generiert):

```
Sptr < NotifyMsg >  myNotify = new NotifyMsg();
myNotify->setCallId(inviteMsg->getCallId());
myNotify->setFrom(myFrom);
myNotify->setTo(myTo);
myNotify->setNotifyDetails(myUrl);
```

Der Kontext wird als Inhalt der NOTIFY-Nachricht verpackt und versendet:

```
Data myNotifyTxt(myMessage);
myNotify->setNotifyMsg(myNotifyTxt);
Sptr < SipEvent > sipEvent;
sipEvent.dynamicCast( event );
sipEvent->getSipStack()->sendAsync( myNotify );
```

Answer Switch

Durch Rückfragen an den Anrufer soll die Möglichkeit geschaffen werden, dem Anrufer selbst entscheiden zu lassen, wie mit dem offenen Rufaufbau fortgefahren werden soll. Anschließend soll die Eingabe des Benutzers wieder zurück an den Server geschickt und dort weiterverarbeitet werden. Je nach Antwort verzweigt das Skript in die Pfade unter `<yes>` oder `<no>`. Um die Nachrichten leichter identifizieren zu können, wurde das Headerfeld `request_answer` für die INVITE Nachricht definiert. Ist das Feld nicht enthalten, so wird zu `<normal-call>` verwiesen.

Die Funktionalität dieser Operatoren befindet sich in den Objekten „CPLOpRequest“, “CPLOpRequestAck“ und „CPLOpRequestAnswer“. In „CPLOpRequest“ wird die Rückfrage in Form einer 183 Session in progress-Statusnachricht behandelt. Endgeräte, die den Inhalt nicht interpretieren können, behandeln die Nachricht als normale Status-Information. In der Nachricht wurde das SIP-Headerfeld `request` eingefügt, damit die Nachricht vom empfangenden UA richtig interpretiert werden kann.

```
StatusMsg statusMsg( *sipCommand, 183 );

SipAnswerRequest answerRequest;
Data myAnswerRequest(status);
answerRequest.set(myAnswerRequest);
statusMsg.setAnswerRequest(answerRequest);

sendReply( event, statusMsg );
```

Eine Besonderheit beim *Answer Switch* ist das Warten auf eine Bestätigung des Endgerätes des Anrufers. Nachdem die Nachricht aus „CPLOpRequest“-Objekt gesendet wurde, wird anschließend in einen Wartezustand verzweigt. Sobald eine Antwort eingeht oder ein Timeout erfolgt, wird der Operator „CPLOpRequestAck“ durch einen „Timeout-Event“ aktiviert. Im Falle eines Timeout-Events wird mit dem internen Operator `<request_failure>` fortgefahren. Wenn eine Antwort eingeht, wird anhand der Antwort weiterverzweigt.

9.5.2.4 Erweiterte Komponenten

Die Reihe von Komponenten wurde erweitert. Eine Übersicht der modifizierten Elemente ist in Abbildung 9.14 zu sehen. Neu hinzugefügte Komponenten sind mit einer hellen Box gekennzeichnet. Erweiterte Teile sind dunkel hinterlegt.

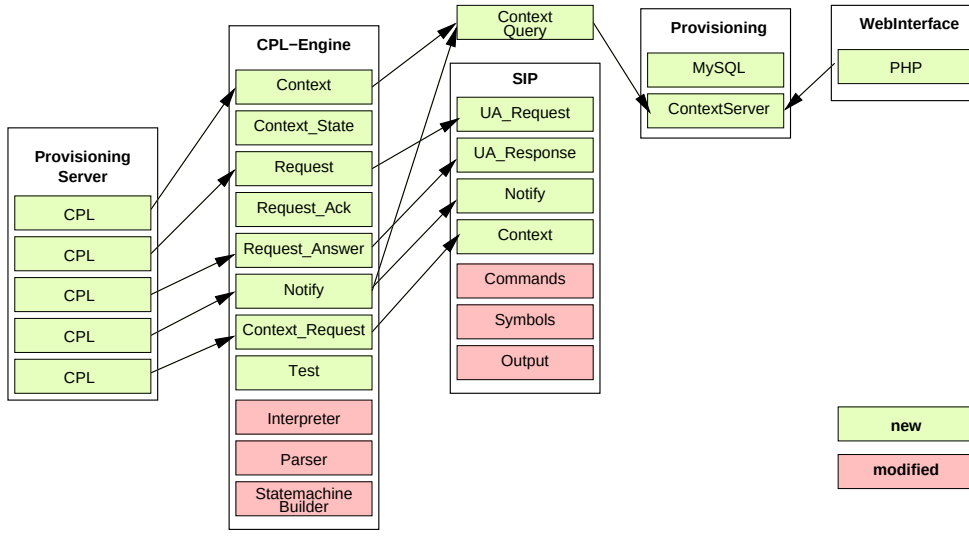


Abbildung 9.14: Neue und modifizierte Komponenten der CPL-Engine

CPL Engine

Die meisten Erweiterungen betreffen die CPL-Engine, die um eine Reihe neuer CPL-Befehle erweitert wurde. Hierzu zählen insbesondere die in Unterabschnitt 8.2.1 vor-

gestellten Operatoren *Context Lookup*, *Context Switch*, *Context Notifier* und *Answer Switch*. Ein Objekt namens *ContextStates* wurde hinzugefügt, um die Kontexte der Nutzer abzuspeichern. Die Befehle *Request-Ack* und *Request-Answer* wurden zusätzlich zum Befehl *Request* benötigt, um das Acknowledgement bzw. die Antwort auf die Rückfrage beim *Answer Switch* zu verarbeiten.

Der Parser (CPLFeatureBuilder) wurde bearbeitet, damit dieser ein CPL-Skript mit den neuen CPL-Befehlen einlesen und interpretieren kann. Im Interpreter und in den Objekten, welche die Statemaschine aufbauen, wurde die Vorgehensweise zur Integration der neuen Befehle in die Statemaschine definiert.

Context Query

Das Objekt *Context* wurde als Schnittstelle zwischen dem ContextServer und der CPL-Engine definiert. Alle neuen Operatoren können dieses Objekt nutzen, um Kontext-Operationen durchführen zu können. Mit diesem Objekt wurde die Interaktion mit dem ContextServer über SQL-Statements realisiert. Eine MySQL-Bibliothek wurde für die interne Handhabung der SQL-Ausdrücke und der Kommunikation mit der Datenbank eingebunden.

Provisioning Server

Der Provisioning Server wurde zum Testen der neuen CPL-Befehle erweitert. Für jeden neuen Befehl wurde ein CPL-Skript erstellt. Zu jedem dieser CPL-Skripte wurde ein Teilnehmer registriert und das entsprechende Feature wurde aktiviert. Dadurch konnten alle neuen Befehle isoliert aufgerufen und getestet werden.

9.5.2.5 Bewertung

Mittlerweile sind die Erweiterungen auch auf den SER-Server [www24] übertragen worden. Der SER-Server besitzt die Möglichkeit, seine Funktionalität durch Zuladen von Modulen zu erweitern. Während die VOCAL-Suite von VOVIDA eine erste umfassende Open Source Implementierung des Session Initiation Protocols und ihrer Komponenten in hoher Code-Qualität darstellt, hat sich der SER-Server zu einem de-facto Standard etabliert. Eine Reihe von kommerziellen IP-Telefonie-Anbietern setzen diesen Server im Wirkbetrieb ein.

Ein erweitertes CPL-Modul wurde entwickelt und umgesetzt, welches die hier beschriebenen Funktionalitäten dem SER-Server zur Verfügung stellt. Es hat sich dabei gezeigt, dass das gewonnene konzeptionelle Verständnis über die Erweiterung der VOCAL CPL-Engine mit vertretbarem Implementierungsaufwand auf verschiedene andere Systeme übertragen werden kann.

9.6 Zusammenfassung

Das Gesamtsystem besteht aus einer Reihe von Einzelkomponenten. Diese zeichnen sich durch ihrer sehr heterogenen Charakteristika aus. So besitzen die Komponenten unterschiedliche Komplexitäten und Anforderungen an die Plattform, auf der sie ausgeführt werden. Auch wurden die Softwarekomponenten in unterschiedlichen Programmiersprachen erstellt, da oftmals eine spezielle Funktionsbibliothek oder die Ausgangsbasis nur in einer bestimmten Sprache zur Verfügung stand.

Das somit verteilte und heterogene System wird durch die Kommunikationsmiddleware zusammengehalten. Die Verteilung der Komponenten sowie die Kommunikation über definierte Protokolle und Schnittstellen bietet auch Vorteile. So können die jeweils geeignetsten Lösungen für ein Teilgebiet mit einbezogen werden, ohne das Gesamtsystem zu modifizieren. Es hat sich gezeigt, dass ein solche komplexes System in einem vertretbaren Aufwand realisiert werden kann. Dies wurde im Wesentlichen durch die Wiederverwendung von existierenden Teillösungen und deren Erweiterungen erreicht.

Kapitel 10

Zusammenfassung und Ausblick

Acti iucundi labores.
(*Arbeiten sind angenehm, wenn sie getan sind.*)

De finibus 5, 46
Marcus Tullius Cicero

Kommunikation ist eine essentielle Tätigkeit des privaten und öffentlichen Lebens. Das Kommunikationsaufkommen insbesondere über technische Geräte stieg in den letzten Jahren sprunghaft an. Weniger die verbreiterte Erreichbarkeit sondern vielmehr wie Kommunikation zielgerichtet erfolgen kann und wie sie kanalisiert werden kann, wird zunehmend an Bedeutung gewinnen. Eine wichtige Beobachtung hierbei ist, dass bei vis-à-vis Kommunikation die Erfassung und Berücksichtigung von Kontexten eine entscheidende Rolle spielt.
In der vorliegenden Arbeit wurde die Einbeziehung von Kontexten in den Ablauf von Kommunikationsbeziehungen zur Steigerung der Effizienz vorgeschlagen. Eine Definition von Effizienz sowie eine Diskussion, wie diese in diesem Umfeld gesteigert werden kann, wurde geführt. Daraus wurden konkrete Methoden abgeleitet, analysiert und bewertet. Eine eigene Kontext-Modellierung wurde entwickelt, aus der ein Kontext-Spiral-Model zur systematischen Handhabung von Kontexten entworfen und eingeführt wurde.
Aus den Systemanforderungen zur Bereitstellung von Methoden und Informationen, die eine Erstellung von Kontext-bewussten Kommunikationsdiensten ermöglichen und vereinfachen, wurde ein Framework unter Verwendung von Software-Engineering Methoden entworfen und prototypisch umgesetzt. Aufbauend auf dem Framework wurde eine erweiterte Call Processing Language (CPL) als Skriptsprache entwickelt, mit der ein Nutzer graphisch Dienste erstellen kann. Zusätzlich wurde ein System mit Kommunikationsbrokern entworfen, welches dem Nutzer den Verbindungsaufbau abnimmt und für beide Seiten günstige Kommunikationszeitpunkte aushandelt und präsentiert.

10.1 Ergebnisse dieser Arbeit

Die erzielten Ergebnisse dieser Arbeit werden in diesem Abschnitt zusammengefasst. Aus einer allgemeinen Betrachtung über Kommunikation mit technischen Mitteln wurde das Themenfeld für eine weitergehende Problemanalyse identifiziert. Aus der allgemeinen Problemstellung wurden Verfahren für eine konkrete Ausprägung abgeleitet. Für diese wurden Anforderungen und offene Fragen identifiziert und ein umfassender systemorientierter Lösungsvorschlag vorgestellt und umgesetzt. Die erarbeiteten Konzepte, Methoden und Verfahren wurden anschließend generalisiert und auf eine allgemeinere Problemstellung übertragen . Sie stellen somit einen Beitrag auch in dem übergeordneten Gebiet der Kontext-bewussten Dienste und Anwendungen dar.

Die technische Möglichkeit, an fast allen Orten zu kommunizieren, führt zu einem stetig anwachsenden Aufkommen an Kommunikation. Dem hohen Grad an Mobilität der Nutzer wird durch eine Vielzahl an mobilen Endgeräten Rechnung getragen. Die reine technische Verfügbarkeit führt jedoch nicht zu einer effizienten Kommunikation. Nutzer haben ein legitimes Interesse, die Kontrolle über eingehende Kommunikationsanfragen zu besitzen und diese möglichst einfach und zielführend zu behandeln. Die Unkenntnis des aktuellen Kontexts des Angerufenen ist ein häufiger Grund dafür, dass die Kommunikation nicht zu einem zufrieden stellenden Ergebnis führt. Die Beachtung des Kontexts spielt bei einer direkten Kommunikation jedoch eine entscheidende Rolle. Aus dieser Beobachtung wurden Verfahren abgeleitet, wie der Kontext für eine Kommunikation über Entfernung und mit technischen Mitteln einbezogen werden kann, so dass die Kommunikation *effizienter* für beide Teilnehmer – sowohl für Anrufer als auch für Angerufene – wird.

In dieser Arbeit wurden neuartige Verfahren zur Steigerung der Effizienz beim Aufbau und der Kontrolle eingehender Kommunikation entworfen. Mittels einer Effizienzbetrachtung wurden die Faktoren identifiziert, welche die Effizienz beeinflussen können. Aus diesen wurden diejenigen Parameter bestimmt, die sich mit den beschriebenen Verfahren beeinflussen lassen. Dabei wird zwischen dem Aufwand des Systems für einen erfolgreichen Verbindungsaufbau und dem Aufwand, den der Nutzer dabei hat, unterschieden. Eine Verbesserung des Kommunikationsaufbaus kann durch Reduktion der nicht zum Erfolg führenden Versuche erreicht werden. Dazu wurden drei konkrete Verfahren vorgeschlagen: Zum einen die Einbeziehung des Nutzerkontexts in die Filterung eingehender Kommunikationsanfragen. Zum anderen der Austausch des Kontexts an den Anrufer, der die weitere Entscheidung, ob die Kommunikation aufgebaut werden soll, auf Basis dieser Information treffen kann. Ein drittes Verfahren erlaubt eine weitgehende Verlagerung des Aufwands vom Nutzer zum System, was durch die Verwendung von Kommunikationsbrokern zur Aushandlung von Kommunikationszeitpunkten erreicht wird.

Bei den vorgeschlagenen Verfahren wurde die Einbeziehung von *Kontexten* als eine Schlüsselkomponente zur Verbesserung der Effizienz der Kommunikation identifiziert. Daraus wurde die neuartige Klasse der *Kontext-bewussten* Kommunikationsdienste definiert. Um Kontexte sinnvoll in einem System verwenden zu können, müssen diese

geeignet modelliert werden. Eine neuartige Modellierung wurde erarbeitet, in der Kontexte als Objekte aufgefasst werden, die aus einer sehr großen Zahl an Kontextmerkmalen bestehen können. Kontexte bzw. ihre Merkmale sind ab ihrer Erfassung zeit- und ortsabhängig. Es wurde die Annahme getroffen, dass Kontexte aus einer Untermenge aller Merkmale mit einer charakteristischen Vorschrift approximiert werden können. Mit dem *Spiral-Modell* wurden Prozessphasen identifiziert und zusammengefasst, die die Erfassung, Gewinnung, Verteilung und schließlich Nutzung von Kontexten beschreiben.

Im Gegensatz zu anderen Arbeiten wurde ein *systemorientierter* Ansatz verfolgt und ein komplettes Framework entworfen und umgesetzt. Das Framework unterstützt alle Phasen des Kontext-Spiral-Modells und berücksichtigt dabei die Charakteristika der Echtzeit-Kommunikationsdienste. Es wurde gezeigt, dass das in dieser Arbeit entwickelte Design-Prinzip der Trennung von Kontextproduzenten, -verteilern und -Nutzern ein sehr geeignetes Fundament für die Unterstützung von Kontext-bewussten Kommunikationsdiensten ist.

Die daraus abgeleitete dreigeteilte Architektur besteht aus einem Kontextaggregationsnetzwerk, dem ContextServer als Verteilkomponente sowie den Kontextnutzern in Form von Kontext-bewussten Diensten. Im Gegensatz zu anderen Ansätzen werden Kontexte durch eine standardisierte Schnittstellenbeschreibung bereitgestellt. Der entwickelte ContextServer bietet Funktionen zur persistenten Speicherung und Verteilung über eine Web Service-Schnittstelle an, über welche die Anwendungen auf die bereitgestellten Methoden zugreifen können.

In der vorliegenden Arbeit wurden unterschiedliche Kommunikationsverfahren, die eine verteilte Architektur ermöglichen, untersucht und bewertet. Eine Kombination aus ereignisbasierter Kommunikation und direkter Peer-to-Peer In-band-Signalisierung unter Verwendung des Session Initiation Protocols (SIP) wurde entworfen und realisiert. Für den Austausch der Daten wurde mit Presence Information Data Format – Context Enhanced ein eigenes Format entwickelt.

Für jede der vier Phasen des Spiral-Modells wurden eigene Ansätze erarbeitet, verwandte Arbeiten identifiziert und betrachtet. Ein Großteil der funktionalen Komponenten und Protokolle wurden prototypisch umgesetzt. Ein auf der Entfernungsabhängigkeit der Signalstärken in einer Wireless-LAN-Infrastruktur beruhendes Innenraumlokationsverfahren wurde als Kontextmerkmalsquelle realisiert. Mit dem Context Aggregation Network (CAN) wurde ein Netzwerk entworfen, dessen Operator-Knoten die notwendigen Schritte zur Komposition von Sensordaten zu Kontexten ausführen kann. Für die Erstellung der Topologie des Kontextaggregationsnetzwerks wurde eine eigene Sprache CAL (Context Aggregation Level Language) entwickelt.

Für die Synthetisierung der Kontextmerkmale zu Kontexten wurden mehrere Lösungsansätze untersucht. In dieser Arbeit wurde die Verwendung eines Fuzzy Logik-basierten Regelsystems vorgeschlagen. Im Unterschied zu anderen untersuchten Ansätzen erlaubt es die Fuzzy Logik, die Komposition von Kontexten natürlich sprachlich aber dennoch mathematisch formal zu beschreiben. Dies erlaubt eine wesentliche Vereinfachung für

die Spezifikation der Kontexte durch den Nutzer. Die Leistungsfähigkeit der Fuzzy Logik Regel-Engine wurde durch eine Performanzmessung mit simulativen Sensoren und einer Plausibilitätsüberprüfung getestet.

Die Nutzung von Kontexten wurde für unterschiedliche Diensttypen und Endgeräte konzipiert und prototypisch umgesetzt. IP-Telefonie Endgeräte (SIP-UA) wurden derart erweitert, dass sie eine Kontext-bewusste Anrufweiterleitung und damit eine Filterung der eingehenden Anrufe ausführen können. Zusätzlich wurde eine Erweiterung der Protokollsemantik des Session Initiation Protocols sowie der internen Verarbeitungslogik des UAs erarbeitet und implementiert. Diese Erweiterung erlaubt die In-band Anfrage von Kontexten und deren Verteilung an andere User Agents.

Einen neuartigen Ansatz leistet diese Arbeit in der Erweiterung der Call Processing Language (CPL) als Skriptsprache zur nutzerzentrierten Erstellung von Kontext-bewussten Diensten. Die Sprache erlaubt einer breiten Nutzergruppe die Realisierung einer effizienten Handhabung eingehender Kommunikation. Dazu wurde sowohl die Sprachsyntax als auch die sie interpretierende CPL-Engine entsprechend erweitert. Ein graphischer Editor erlaubt die einfache Erstellung von CPL-basierten Kommunikationsdiensten, was eine weitere große Vereinfachung des Prozesses darstellt und die Fehlermöglichkeiten reduziert.

In der Arbeit wurde ein Algorithmus entwickelt, der eine (fast) vollständige Verlagerung der Aufwände für den Kommunikationsaufbau auf einen Kommunikationsbroker ermöglicht, was zu einer sehr hohen Effizienz für den Nutzer führt. Der entworfene Digital Call Assistant stellt eine Ausprägung eines Kommunikationsbrokers dar. Der Nutzer spezifiziert seine Kommunikationsvorhaben mittels Regeln und der Digital Call Assistant handelt mit dem jeweilig anderen Assistenten für beide Teilnehmer erfolgversprechende Zeitpunkte aus. Zur Planung werden elektronische Kalender, Kontexte, Nutzerpräferenzen sowie verfügbare Kommunikationstypen berücksichtigt. Die Zeitpunkte werden dem Nutzer präsentiert und bei Bedarf wird die Kommunikation aufgebaut.

Die vorgestellte Arbeit stellt einen systemorientierten Ansatz zur Bereitstellung von Konzepten, Algorithmen und Werkzeugen für Kontext-bewusste Echtzeitkommunikationsdienste dar. Die Arbeit leistet einen wesentlichen Beitrag in der Übertragung der Methoden in den Kontext der SIP-basierten IP-Telefoniedienste. Diese stellen sehr anspruchsvolle Echtzeitkommunikationsdienste dar, welchen ein großes Potenzial eingeräumt wird. Die Arbeit zeigt, wie die vorgestellten Verfahren zur Effizienzsteigerung von Kommunikationsbeziehungen als Proof-of-Concept in Form einer Open-Source-Implementierung zu realisieren sind. Die Methodiken sowie die Erfahrungen bei der Umsetzung können auf das allgemeinere Feld der Kontext-bewussten Dienste und Anwendungen übertragen werden. Damit leistet die Arbeit einen wichtigen Beitrag auf dem Gebiet des Kontext-bewussten Rechnens.

10.2 Zukünftige Anwendungsgebiete und Erweiterungen

Das in dieser Arbeit entwickelte konzeptuelle Framework besteht aus einer Architektur und Kommunikationsverteilmechanismen zur Bereitstellung von Kontexten. Ein weiterer Beitrag stellt eine Skriptsprache zur graphisch unterstützten Erstellung von Kontext-bewussten Kommunikationsdiensten dar. Aufbauend auf den Ergebnissen der Arbeit wurden Anknüpfungspunkte für Erweiterungen der vorgestellten Methoden identifiziert und beschrieben. Diese können die erarbeiteten Konzepte und Methoden nutzen und auf dem erarbeiteten Fundament errichtet werden.

Die Gewinnung von Kontexten benötigt eine große Zahl an Sensoren, die die einzelnen Kontextmerkmale erfassen können. Das dieser Arbeit zugrunde liegende Szenario sieht eine Infrastruktur vor, welche eine große Anzahl an Sensoren enthält. Die Infrastruktur stellt die Energieversorgung der Sensoren sicher und ermöglicht den Zugriff auf die Sensoren durch Anwendungen. Darüber hinaus stehen die Infrastruktur sowie die sie nutzenden Kontext-bewussten Dienste und Teilnehmer unter derselben administrativen Verwaltung. Die Anzahl der Sensoren und die Sensortypen können daher festgelegt werden. Eine geeignete Topologie, sowie die Operatoren für eine Kontextaggregation können durch die Sprache CALL spezifiziert und administriert werden. Diese Voraussetzung erleichtert die Handhabung der Kontextgewinnung. Einige interessante Forschungsthemen ergeben sich durch Aufweitung der statischen Konfiguration der Infrastruktur zu dynamischen Umgebungen.

Die Administration der Sensoren und ihre Auswahl und Zusammenstellung zu einem Netzwerk ist eine mühsame und immer wiederkehrende Aufgabe. Die in dieser Arbeit vorgeschlagenen *Selbstbeschreibungsfähigkeiten* der Sensoren stellen hier einen ersten Ansatz zur Automatisierung der notwendigen Vorgänge dar. Jeder Sensor besitzt dabei einen Satz an Attributen, welche ihn eindeutig identifizieren und spezifizieren. Über den Typ des Sensors bzw. die bereitgestellte Funktionalität des Aggregationsoperators können Anwendungen selbstständig die passenden Knoten finden und zu einem Netz zusammenstellen. Diese Eigenschaften sind für den Einsatz in mobilen Szenarien eine Voraussetzung, damit ein Client in einer fremden Umgebung Datenquellen finden und nutzen kann.

Die Existenz einer Infrastruktur ist in dem dieser Arbeit zu Grunde liegenden Szenario gerechtfertigt. Jedoch kann eine Ausweitung auf bestimmte mobile Szenarien oder auf andere Anwendungen diese Vorgabe hinfällig machen. In einer infrastrukturlosen Umgebung, wie dies häufig bei Sensornetzen anzutreffen ist, müssen andere Randbedingungen beachtet werden. Hierzu zählen Energiehaushalt der Sensoren und Knoten, die eingeschränkten Ressourcen sowie das Routing der Daten. In diesen *ad-hoc Szenarien* muss darüber hinaus durch Verfahren die Verlässlichkeit des Netzes sichergestellt werden, was möglicherweise insbesondere durch nicht-protokollkonforme Knoten erschwert wird [Hol04]. Ein weiteres Forschungsgebiet, welches sich in diesem Zusammenhang eröffnet, ist der Einsatz eines *selbstorganisierenden* Netzes zum Aufbau einer Topologie für ein Kontextaggregationsnetzwerk. Selbstbeschreibungsfähigkeiten der Knoten sowie ad hoc Routingverfahren stellen die grundlegenden Techniken für ein selbstorganisie-

rendes Netz dar, in dem die Anwendungen nach Kontexten anfragen können und diese geliefert bekommen.

Neben der Vereinfachung der Administration und des Betriebs von Netzwerken zur Verteilung von Kontextmerkmalen kann eine weitere manuelle Tätigkeit automatisiert werden. Der Prozess der Regelerstellung – sowohl für den Syntheseprozess, wie er mit Fuzzy Logik in dieser Arbeit vorgeschlagen wurde, als auch für die Erstellung der Regeln zur Spezifikation der Kommunikationshandhabung für beispielsweise den Digital Call Assistant – kann durch *selbstlernende* Verfahren vereinfacht werden. Dabei lernt das System während einer Trainingsphase, welche Kombinationen an Sensoren zu dem gewünschten Kontext kombiniert werden können. Aus diesen Beobachtungen werden beispielsweise mittels Entscheidungsbäumen Vorschriften abgeleitet. Ebenso kann das Verhalten des Nutzers in unterschiedlichen Kontexten in Bezug auf sein Kommunikationsverhalten betrachtet und analysiert werden. In einer Überprüfungsphase kann der Nutzer die Entscheidungen des Systems überwachen und beurteilen, bis schließlich das System weitgehend automatisiert die Entscheidungen trifft.

Die während der Erfassung von Kontextmerkmalen anfallenden Daten, wie sie zur Bestimmung des Kontexts des Nutzers benötigt werden, sind bereits sensible und oftmals auch personenbezogene Informationen. Der Kontext ist in der Regel noch aussagekräftiger und daher als noch sensibler einzustufen. Daher haben Nutzer ein berechtigtes Interesse, diese Daten vor der Einsicht von Fremden zu schützen. Datenschutzrechlichte Fragen, wie z. B. wer, wann, welche Daten wo speichern kann, müssen geklärt sein, damit die vorgestellten Verfahren Akzeptanz finden. Die Bereitstellung von Mechanismen zur Gewährleistung der *Sicherheit* der Daten sowie dem Schutz der *Privatsphäre* in einem dynamischen Umfeld mit vielen (auch unbekannten) Parteien stellen eine wissenschaftliche Herausforderung und unabdingbare Vorbedingung für den Einsatz in Unternehmen und in einer breiten Nutzerschicht dar. Die vorgeschlagenen Detail-Level für die ausgetauschten Kontexte bieten zusammen mit der optionalen Authentisierung über den für SIP spezifizierten Mechanismus einen ersten Ansatz. Die für SIP vorgeschlagenen Verfahren zur Verschlüsselung und zur Integritätssicherung bieten ebenfalls einen Ausgangspunkt für eine leistungsfähigere Lösung.

10.3 Fazit

Die Arbeit leistet einen wichtigen Beitrag zur Erzeugung und Bereitstellung von nutzerzentrierten Echtzeitkommunikationsdiensten. Hierbei stellt die Einbeziehung von Kontexten in die Verarbeitung von Kommunikationsbeziehungen eine Neuerung im Bereich der Kommunikationsdienste dar. Ferner sind Kommunikationsdienste im Forschungsbereich des Kontext-bewussten Rechnens bisher nicht vollständig betrachtet worden. Auch hier liefern die gewonnenen Erkenntnisse ein zukunftsträchtiges Fundament für weitere Arbeiten in diesem Bereich.

Die erarbeiteten methodischen Konzepte für die Steigerung der vom Nutzer wahrnehmbaren Effizienz beim Aufbau und der Steuerung von Kommunikationsbeziehungen wurden prototypisch zum Nachweis der Realisierbarkeit umgesetzt. Dabei wurden durch ein

entwickeltes Framework eine Infrastruktur sowie dem Nutzer Methoden bereitgestellt, um Kontext-bewusste Echtzeitkommunikationsdienste zu realisieren. Die erzielten Ergebnisse und Erfahrungen können auf das allgemeinere Gebiet der Kontext-bewussten Dienste und Anwendungen übertragen werden, da das Framework über generische Konzepte auch diese Dienste unterstützen kann.

Ein wichtiges Ergebnis der vorliegenden Arbeit ist, dass es möglich ist, durch Einbeziehung von Kontexten effizientere Echtzeitkommunikationsdienste zu ermöglichen. Dabei werden in der menschlichen Kommunikation beobachtete Techniken von dem entworfenen System nachgeahmt und Diensten zur Verfügung gestellt. Es hat sich gezeigt, dass die Verwendung von Kontexten sehr intuitiv zu erfassen ist und bestehende Dienste nur geringfügig angepasst werden müssen. Das Verhältnis von Aufwand zu Nutzen ist folglich als sehr günstig zu bewerten. Eine breite Nutzermasse kann durch die Bereitstellung einer graphisch unterstützten Skriptsprache zur Erstellung individueller Dienste angesprochen werden. Zusammen mit der regelbasierten Kommunikation über Broker verkleinert dies die Lücke zu einem digitalen Äquivalents eines menschlichen Assistenten zur Handhabung von Kommunikationsbeziehungen. Eine grundlegende Erkenntnis der Arbeit ist, dass das System und die notwendige Infrastruktur zur Erfassung und Bereitstellung von Kontexten sehr komplex ist. Durch die Trennung der Dienste vom System kann diese Komplexität durch definierte Schnittstellen verborgen werden, was ein entscheidendes Merkmal von pervasiven und ubiquitären Systemen ist.

Literaturverzeichnis

[AAB+98] Aksoy, Demet, Mehmet Altinel, Rahul Bose, Ugur Cetintemel, Michael J. Franklin, Jane Wang und Stanley B. Zdonik: *Research in Data Broadcast and Dissemination.* In: *Advanced Multimedia Content Processing*, Seiten 194–207, 1998.

[AAFSZ95] Acharya, A., S. Alonso, M. Franklin und S. S. Zdonik: *Broadcast Disks: Data Management for Asymmetric Communication Environments.* In: *ACM SIGMOD*, Seiten 59–68, San Jose, CA USA, Juni 1995. ACM Press.

[AAH+97] Abowd, Gregory D., Christopher G. Atkeson, Jason Hong, Sue Long, Rob Kooper und Mike Pinkerton: *Cyberguide: a Mobile Context-aware Tour Guide.* Wireless Networks, 3(5):421–433, 1997.

[ABPRT03] Austin, Daniel, Abbie Barbir, Ed Peters und Steve Ross-Talbot: *Web Services Choreography Requirements 1.0.* W3c working draft WD-ws-chor-reqs-20030812, World Wide Web Consortium, August 2003.

[ACD+03] Andrews, Tony, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, SAP Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana Trickovic und Sanjiva Weerawarana: *Specification: Business Process Execution Language for Web Services Version 1.1.* Technischer Bericht, Mai 2003.

[ACK94] Asthana, Abhaya, Mark Cravatts und Paul Krzyzanowski: *An Indoor Wireless System for Personalized Shopping Assistance.* In: *IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, US, 1994.

[Ack03] Ackermann, Ralf: *Gateways and Components for Supplementary IP Telephony Services in Heterogeneous Environments.* Dissertation, Technische Universität Darmstadt, KOM, Darmstadt, Mai 2003.

[ACL+04] Arkko, J., E. Carrara, F. Lindholm, M. Naslund und K. Norrman: *MIKEY: Multimedia Internet KEYing.* Request for Comments 3830, Internet Engineering Task Force, August 2004.

[AF03] Andreasen, F. und B. Foster: *Media Gateway Control Protocol (MGCP) Version 1.0.* Request for Comments 3435, Internet Engineering Task Force, Januar 2003.

[AGKS01] Ackermann, Ralf, Manuel Görtz, Martin Karsten und Ralf Steinmetz: *Prototyping a PDA based Communication Appliance.* In: *Proceedings of Softcom 2001, Split*, Oktober 2001.

[AGS+93] Adams, Norman, Rich Gold, Bill N. Schilit, Michael Tso und Roy Want: *An Infrared Network for Mobile Computers.* In: *Proceedings of the USENIX Symposium on Mobile and Location-independent Computing*, Seiten 41–52, Cambridge, MA, USA, August 1993.

[AKGM00] Amer, M., A. Karmouch, T. Gray und S. Mankovskii: *Feature Interaction Resolution using Fuzzy Policies.* In: Calder, M. H. und E. H. Magill (Herausgeber): *Proc. 6th Feature Interactions in Telecommunications and Software Systems*, Seiten 94–112, Amsterdam, Niederlande, Mai 2000. IOS Press.

[AKW88] Aho, Alfred V., Brian W. Kernighan und Peter J. Weinberger: *The AWK Programming Language.* Addison-Wesley, 1988.

[AL03] Amyot, Daniel und Luigi Logrippo (Herausgeber): Amsterdam, The Netherlands, Juni 2003. IOS Press.

[Amy01] Amyot, Daniel: *Specification and Validation of Telecommunications Systems with Use Case Maps and Lotos.* Dissertation, University of Ottawa, November 2001.

[ANS89] ANSA: *The Advanced Network System Architecture (ANSA).* Reference Manual, Architecture Project Management, 1989.

[ANS98] ANSI/IEEE: *Local and metropolitan area networks – specific requirements – part 5: Token ring access method and physical layer specifications.* Technischer Bericht Std 802.5-1998E, LAN MAN Standards Committee of the IEEE Computer Society, 1998.

[ANS00] ANSI/IEEE: *Local and metropolitan area networks – specific requirements – part 3: Carrier sense multiple access with collision detection (csma/cd) access method and physical layer specifications.* Technischer Bericht Std 802.3-2002, LAN MAN Standards Committee of the IEEE Computer Society, 2000.

[Aub05] *Voice Browser Call Control: CCXML Version 1.0.* W3c working draft 11 WD-ccxml-20050111, World Wide Web Consortium, Januar 2005.

[Aue86] Auer, Peter: *Kontextualisierung.* Studium Linguistik, Seiten 22–47, 1986.

[Bae98] Baets, W.R.J.: *Organizational Learning and Knowledge Technologies in a Dynamic Environment.* Kluwer Academic Publishers, 1998.

[Bar03] Barnes, M.: *A Mechanism to Secure SIP Information inserted by Intermediaries.* Internet Draft draft-ietf-sipping-sec-inserted-info-01, Internet Engineering Task Force, Oktober 2003. Work in progress.

[Bas00] Bass, T.: *Intrusion detection systems and multisensor data fusion: Creating cyberspace situational awareness.* Communications of the ACM, 43(4):99–105, Mai 2000.

[Bay63] Bayes, T.: *Essay towards solving a problem in the doctrine of chances.* Philosophical Transactions of the Royal Society of London, 53:370–418, 1763. Reprinted in Biometrika, 45:293-315, 1958.

[BBC97] Brown, Peter J., John D. Bovey und Xian Chen: *Context-aware Applications: From the Laboratory to the Marketplace.* IEEE Personal Communications, 5(4):58–64, Oktober 1997.

[BBC+04a] Bajaj, Siddharth, Don Box, Dave Chappell, Francisco Curbera, Glen Daniels, Phillip Hallam-Baker, Maryann Hondo, Chris Kaler, Dave Langworthy, Ashok Malhotra, Anthony Nadalin, Nataraj Nagaratnam, Mark Nottingham, Hemma Prafullchandra, Claus von Riegen, Jeffrey Schlimmer, Chris Sharp und John Shewchuk: *Web Services Policy Framework (WS-Policy).* Technischer Bericht, September 2004.

[BBC+04b] Berglund, Anders, Scott Boag, Don Chamberlin, Mary F. Fernàndez, Michael Kay, Jonathan Robie und Jèrôme Simèon: *Xml path language (xpath) 2.0.* W3c working draft WD-xpath20-20041029, World Wide Web Consortium, Oktober 2004.

[BBD00] Beddus, S., G. Bruce und S. Davis: *Opening up networks with jain parlay.* IEEE Communications Magazine, 38(4), 2000.

[BBHM95] Bacon, Jean, John Bates, Richard Hayton und Ken Moody: *Using Events to build Distributed Applications.* In: *IEEE SDNE Services in Distributed and Networked Environments*, Seiten 148–155, Whistler, British Columbia, USA, 1995.

[BBHS03] Bauer, M., C. Becker, J. Hähner und G. Schiele: *ContextCube - providing context information ubiquitously.* In: *Proceedings of the 23rd International Conference on Distributed Computing Systems Workshops (ICDCS 2003)*, Seiten 308 – 313, Los Alamitos, CA, USA, Mai 2003. IEEE Computer Society Press.

[BBL+00] Brown, Peter, Winslow Burleston, Mik Lamming, Odd-Wiking Rahlff, Guy Romano, Jean Scholtz und Dave Snowdon: *Context-awareness: some Compelling Applications.* In: *CHI 2000 Workshop at Handheld and Ubiquitous Computing 2000 (HUC2k)*, Bristol, UK, August 2000.

[BBM95] Buvač, Saša, Vanja Buvač und Ian A. Mason: *Metamathematics of Contexts.* Fundamenta Informaticae, 23(3), 1995.

[BBM+01] Ballinger, Keith, Peter Brittenham, Ashok Malhotra, William A. Nagy und Stefan Pharies: *Specification: Web Services Inspection Language (WS-Inspection) 1.0.* Technischer Bericht, November 2001.

[BBMS98] Bates, John, Jean Bacon, Ken Moody und Mark Spiteri: *Using events for the scalable federation of heterogeneous components.* In: Guedes, Paulo und Jean Bacon (Herausgeber): *Proceedings of the 8th ACM SIGOPS European Workshop: Support for Composing Distributed Applications*, Sintra, Portugal, 1998.

[BBP00] Bahl, P., A. Balachandran und V. Padmanabhan: *Enhancements to the radar user location and tracking system.* Technischer Bericht MSR-TR-2000-12, Microsoft Research, Februar 2000.

[BBR02] Bauer, Martin, Christian Becker und Kurt Rothermel: *Location models from the perspective of context-aware applications and mobile ad hoc networks.* Personal Ubiquitous Computing, 6(5-6), 2002.

[BCF+04] Boag, Scott, Don Chamberlin, Mary F. Fernàndez, Daniela Florescu, Jonathan Robie und Jèrôme Simèon: *XQuery 1.0: An XML Query Language.* W3c working draft WD-xquery-20041029, World Wide Web Consortium, Oktober 2004.

[BDC+89] Bowen, T.F, F.S Dworack, C.H. Chow, N. Griffeth, G.E. Herman und Y-J Lin: *The feature interaction problem in telecommunications system.* Software Engineering for Telecommunication Switiching Systems, Seiten 59–62, Juli 1989.

[Bec04] *RDF/XML Syntax Specification.* W3c recommendation REC-rdf-syntax-grammar-20040210, World Wide Web Consortium, Februar 2004.

[Bel90] Bellcore: *Advanced intelligent network. release 1 baseline architecture.* Special report 1, Bellcore, März 1990.

[Ber96] Bernstein, Philip A.: *Middleware: A Model for Distributed System Services.* Communications of the ACM, 39(2):86–98, 1996.

[BFKL04] Buschmann, C., S. Fischer, J. Koberstein und N. Luttenberger: *Ein Ansatz zur effizienten Softwareentwicklung für drahtlose Sensornetze auf Basis von Shared Information Spaces.* In: *GI/ITG Fachgespräche 'Systemsoftware für Pervasive Computing'.* GI/ITG, Oktober 2004.

[BHH+01] Burkhardt, J. (Ed.), H. Henn, S. Hepper, K. Rindtorff und T. Schaeck: *Technology and Architecture of Mobile Internet Applications.* November 2001.

[BI98] Brooks, Richard R. und Sundararaja Iyengar: *Multi-Sensor Fusion: Fundamentals and Applications.* Prentice Hall, New Jersey, 1998.

[Bib01] Bibliographisches Institut & F. A. Brockhaus AG und Langenscheidt KG: *Duden Fremdwörter neu*. Dudenverlag, Mannheim/Wien/Zürich, 2001.

[BJ02] Bakker, J.-L. und R. Jain: *Next Generation Service Creation Using XML Scripting Languages*. In: *Proceedings of ICC 2002*, New York, NY (USA), Mai 2002.

[BKK96] Berchtold, S., D. A. Keim und H. P. Kriegel: *The x-tree: An index structure for high-dimensional data*. In: *VLDB*, 1996.

[Bla37] Black, Max: *Vagueness: An exercise in logical analysis*. Philosophy of Science, 4:427 – 455, 1937.

[Bla98] Blackman, S. S.: *Introduction to Sensor Systems*, Kapitel Multiple Sensor Tracking and Data Fusion. Artech House, Norwood, Massachusetts, 1998.

[BMB+00] Bacon, Jean, Ken Moody, John Bates, Richard Hayton, Chaoying Ma, Andrew Mc-Neil, Oliver Seidel und Mark Spiteri: *Generic Support for Distributed Applications*. IEEE Computer, 33(3):68–76, 2000.

[BMC+03] Baugher, McGrew, Carrara, Naslund und Norrman: *The Secure Real-time Transport Protocol*. Internet Draft draft-ietf-avt-srtp-09, Internet Engineering Task Force, Juli 2003. Work in progress.

[BMR+96] Buschmann, Frank, Regine Meunier, Hans Rohnert, Peter Sommerlad und Michael Stal: *Pattern-Oriented Software Architecture, A System of Patterns*. John Wiley & Sons Ltd, Chichester, England, 1996.

[BMR04] Barth, Adam, John C. Mitchell und Justin Rosenstein: *Conflict and Combination in Privacy Policy Languages*. In: *WPES '04: Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, Seiten 45–46, Washington DC, USA, 2004. ACM Press.

[BO99] Bedworth, M.D. und J. O'Brien: *The Omnibus Model: A New Architecture for Data Fusion?* In: *Proceedings of the 2nd International Conference on Information Fusion (FUSION'99)*, Helsinki, Finnland, Juli 1999.

[Bog87] Bogler, P. L.: *Shafer-Dempster Reasoning with Applications to Multisensor Target Identification Systems*. IEEE Transactions on Systems, Man and Cybernetics, 17(6):968–977, November 1987.

[Boo04] *Web Services Architecture*. W3c working group note NOTE-ws-arch-20040211, World Wide Web Consortium, Februar 2004.

[Boy87] Boyd, J. R.: *A discourse on winning and losing*. Unpublished set of briefing slides, Mai 1987.

[BP00] Bahl, Paramvir und Venkata N. Padmanabhan: *RADAR: An in-building RF-based user location and tracking system.* In: *IEEE INFOCOM*, Seiten 775–784, Tel-Aviv, Israel, März 2000. IEEE Computer Society Press.

[BP02] Blasch, Erik. P. und Susan Plano: *Jdl level 5 fusion model: user refinement issues and applications in group tracking.* In: *SPIE Aerosense*, Band 4729, Seiten 270–279, 2002.

[BPSM00] Bray, Tim, J. Paoli und C. M. Sperberg-McQueen: *Extensible Markup Language (XML) 1.0 (second edition).* W3c recommendation REC xml-20001006, World Wide Web Consortium, Oktober 2000.

[BPSM+04] Bray, Tim, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler und Fran cois Yergeau: *Extensible markup language (xml) 1.0 (third edition).* W3C Recommendation, Februar 2004.

[BPW02] Bailey, James, Alexandra Poulovassilis und Peter T. Wood: *Analysis and optimisation of event-condition-action rules on xml.* Computer Networks Journal – Special Issue on Web Dynamics, 39:239–259, 2002.

[BR99] B. Ramsdell, Editor: *S/MIME Version 3 Message Specification.* Request for Comments 2633, Internet Engineering Task Force, Juni 1999.

[Bra89] Braden, R. T.: *Requirements for Internet hosts — communication layers.* Request for Comments 1122, Internet Engineering Task Force, Oktober 1989.

[Bré99] Brézillon, Patrick: *A Survey of the literature of: Context in Artificial Intelligence.* Fundamenta Informaticae, 18(4):321–340, 1999.

[BRG96] Bosse, E., J. Roy und D. Grenier: *Data fusion concepts applied to a suite of dissimilar sensors.* In: *Canadian Conference on Electrical and Computer Engineering 1996*, Seiten 692–695, Mai 1996.

[Bri00] Brieskorn, Jürgen: *Experiences with an H.323 Standard based ENTERPRISE IP Phone.* In: *Proceedings of the 1st IP Telephony Workshop (IPtel 2000)*, Berlin, Germany, April 2000.

[Bri04] *RDF Vocabulary Description Language 1.0: RDF Schema.* W3c recommendation REC-rdf-schema-20040210, World Wide Web Consortium, Februar 2004.

[Bro96] Brown, Peter J.: *The Stick-e Document: A Framework For Creating Context-aware Applications.* In: *In the Proceedings of the Electronic Publishing*, Seiten 259–272, Laxenburg, Austria, September 1996. IFIP.

[BSGT03] Balfanz, Dirk, Jrgen Schirmer, Matthias Grimm und Mohammad-Reza Tazari: *Mobile Situation-awareness within the Project Map.* Computers & Graphics, 27:893 – 898, 2003.

[BTW01] Boley, Harold, Said Tabet und Gerd Wagner: *Design Rationale of RuleML: A Markup Language for Semantic Web Rules.* In: *Proceedings of the SWWS'01*, Stanford, CA, USA, 2001.

[BZJK01] Bakken, D. E., Z. Zhan, C. C. Jones und D. A. Karr: *Middleware support for voting and data fusion.* In: *Proceedings of the International Conference on Dependable Systems and Networks*, Seiten 453–462, Gothenburg, Sweden, 2001.

[Cam01] Campbell, B.: *Framework for SIP Call Control Extensions.* Internet Draft draft-campbell-sip-cc-framework-02.txt, Mai 2001. Work in progress.

[CCF+04] Cabrera, Luis Felipe, George Copeland, Max Feingold, Tom Freund, Jim Johnson, Chris Kaler, Johannes Klein, David Langworthy, Anthony Nadalin, David Orchard, Ian Robinson, John Shewchuk und Tony Storey: *Web Services Coordination (WS-Coordination).* Technischer Bericht, November 2004.

[CD00] Caswell, D. und P. Debaty: *Creating Web Representations for Places.* In: Gellersen, Hans-W und P. Thomas (Herausgeber): *HUC2000, Second In,ternational Symposium on Handheld and Ubiquitous Computing*, Nummer 1927 in *Lecture Notes in Computer Science*, Seiten 114–126, Bristol, UK, September 2000.

[CDM+00] Cheverst, Keith, Nigel Davies, Keith Mitchell, Adrian Friday und Christos Efstratiou: *Developing a Context-aware Electronic Tourist Guide: Some Issues and Experiences.* In: *Proceedings of CHI 2000*, Seiten 17 – 24, April 2000.

[CDMF00] Cheverst, Keith, Nigel Davies, Keith Mitchell und Adrian Friday: *Experiences of Developing and Deploying a Context-aware Tourist Guide: The GUIDE Project.* In: *6th annual International Conference on Mobile Computing and Networking*, Seiten 20 – 31, Boston, Massachusetts, USA, 2000.

[CG89] Carriero, Nicholas und David Gelernter: *Linda in Context.* Communications of the ACM, 32(4):444–458, April 1989.

[CGR+00] Cuervo, F., N. Greene, A. Rayhan, C. Huitema, B. Rosen und J. Segers: *Megaco Protocol Version 1.0.* Request for Comments 3015, Internet Engineering Task Force, November 2000.

[Che04] Chen, Guanling: *Solar: Building a context fusion network for pervasive computing.* Dissertation, Department of Computer Science, Dartmouth College, August 2004.

[Chi04] *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language.* W3c working draft WD-wsdl20-20040803, World Wide Web Consortium, August 2004.

[Cho02] Chown, P.: *Advanced Encryption Standard (AES) Chipersuites for Transport Layer Security (TLS).* RFC 3268, Juni 2002.

[CHvRR04] Clement, Luc, Andrew Hately, Claus von Riegen und Tony Rogers: *UDDI Version 3.0.2. UDDI Spec Technical Committee Draft.* Tc 220041019, OASIS, Oktober 2004.

[Cil02] Cilia, M.: *An Active Functionality Service for Open Distributed Heterogeneous Environments.* Dissertation, Technische Universität Darmstadt, Darmstadt, Darmstadt, Germany, August 2002.

[CK00] Chen, Guanling und David Kotz: *A Survey of Context-Aware Mobile Computing Research.* Technischer Bericht TR2000-381, Dept. of Computer Science, Dartmouth College, November 2000.

[CK02a] Chen, Guanling und David Kotz: *Solar: An open platform for context-aware mobile applications.* In: *Proceedings of the First International Conference on Pervasive Computing (Short paper)*, Seiten 41–47, Juni 2002.

[CK02b] Chen, Yongguang und Hisashi Kobayashi: *Signal strength based indoor geolocation.* In: *International Conference on Communications.* IEEE, April 2002.

[CKM+03] Calder, Muffy, Mario Kolberg, Evan Magill, David Marples und Stephan. Reiff-Marganiec: *Hybrid Solutions to the Feature Interaction Problem.* In: Amyot, Daniel und Luigi Logrippo [AL03], Seiten 187–205.

[CLN00] Chomicki, Jan, Jorge Lobo und Shamin Naqvi: *A Logic Programming Approach to Conflict Resolution in Policy Management.* In: Cohn, Anthony G., Fausto Giunchiglia und Bart Selman (Herausgeber): *KR2000: Principles of Knowledge Representation and Reasoning", publisher = "Morgan Kaufmann*, Seiten 121–132, San Francisco, 2000.

[CLR90] Cormen, T. H., C. E. Leiserson und R. L. Rivest: *Introduction to Algorithms.* MIT Press, 1990.

[CM93] Chakravarthy, Sharma und Deepak Mishra: *Snoop: An expressive event specification language for active databases.* Technical report UF-CIS-TR-92-041, University of Florida, 1993.

[CM00] Castro, Paul und Richard Muntz: *Managing context data for smart spaces.* IEEE Personal Communications, 7(5), Oktober 2000.

[CMKRM03] Calder, Muffy, Evan Magill, Mario Kolberg und Stephan Reiff-Marganiec: *Feature interaction: A critical review and considered forecast.* Computer Networks, 41(1):115–141, Januar 2003.

[CO97] Crocker, D. und P. Overell: *Augmented BNF for Syntax Specifications: ABNF.* Request for Comments 2234, Internet Engineering Task Force, November 1997.

[CR99] Coar, Ken A. L. und D.R.T. Robinson: *The WWW Common Gateway Interface Version 1.1.* Internet Draft draft-coar-cgi-v11-03, Internet Engineering Task Force, Juni 1999.

[Cro82] Crocker, D.: *RFC 822: Standard for the format of ARPA Internet text messages.* Request for Comments 822, Internet Engineering Task Force, August 1982.

[CRS+02a] Campbell, B., J. Rosenberg, H. Schulzrinne, C. Huitema und D. Gurle: *Session Initiation Protocol (SIP) Extension for Instant Messaging.* Request for Comments 3428, Internet Engineering Task Force, Dezember 2002.

[CRS+02b] Campbell, B., J. Rosenberg, H. Schulzrinne, C. Huitema und D. Gurle: *Session Initiation Protocol (SIP) Extension for Instant Messaging.* Request for Comments 3428, Internet Engineering Task Force, Dezember 2002.

[CRW00] Carzaniga, Antonio, David S. Rosenblum und Alexander L. Wolf: *Achieving scalability and expressiveness in an internet-scale event notification service.* In: *Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing (PODC-00)*, Seiten 219–228, New York, NY, USA, Juli 2000. ACM Press.

[CRW01] Carzaniga, Antonio, David S. Rosenblum und Alexander L. Wolf: *Design and evaluation of a wide-area event notification service.* ACM Transactions on Computer Systems, 19(3):332–383, März 2001.

[CS96] Cheeseman, P. und J. Stutz: *Advances in Knowledge Discovery and Data Mining*, Kapitel Bayesian Classification (AutoClass): Theory and Results. AAAI Press/MIT Press, 1996.

[DA99] Dieks, T. und C. Allen: *The TLS Protocol – Version 1.0.* Request for Comments 2246, Internet Engineering Task Force, 1999.

[DA00] Dey, Anind und Gregory Abowd: *The context toolkit: Aiding the development of context-aware applications.* In: *Proceedings of Workshop on Software Engineering for Wearable and Pervasive Computing*, 2000.

[Dam02] Damianou, Nicodemos C.: *A Policy Framework for Management of Distributed Systems.* Dissertation, Imperial College of Science, Technology and Medicine, University of London, Februar 2002.

[DAMV00] Day, M., S. Aggarwal, G. Mohr und J. Vincent: *Instant Messaging / Presence Protocol Requirements.* Request for Comments 2779, Internet Engineering Task Force, Februar 2000.

[Das97] Dasarathy, B. V.: *Sensor Fusion Potential Exploitation-Innovative Architectures and Illustrative Applications.* In: *Proceedings of the IEEE*, Band 85, Seiten 24–38, Januar 1997.

[Das01] Dasarathy, B. V.: *Information Fusion — what, where, why, when, and who.* Information Fusion, 2(2):75–76, 2001. Editorial.

[DCG+04] Dini, P., A. Clemm, T. Gray, F. J. Lin, L. Logrippo und S. Reiff-Marganiec: *Policy-enabled Mechanisms for Feature Interactions: Reality, Expectations, Challenges.* Computer Networks, 45(5):585–603, August 2004.

[DCME01] Davies, Nigel, Keith Cheverst, Keith Mitchell und Alon Efrat: *Using and Determining Location in a Context-sensitive Tour Guide.* IEEE Computer, 34(8), 2001.

[Dem67] Dempster, A. P.: *Upper and Lower Propabilities Induced by a Multi-Valued Mapping.* Annual Mathematical Statistics, 38:325–339, 1967.

[Deu88] Deutsches Institut für Normung: *DIN 44300 – Deutsche Norm – Informationsverarbeitung Begriffe Teile 1 - 9.* Beuth Verlag GmbH, Berlin, November 1988.

[Deu00] Deutsches Institut für Normung: *DIN EN ISO 9000:2000 – Qualitätsmanagementsysteme - Grundlagen und Begriffe.* Beuth Verlag GmbH, Berlin, Dezember 2000.

[Dey98] Dey, Anind K.: *Context-aware Computing: The CyberDesk Project.* In: *AAAI 1998 Spring Symposium on Intelligent Environments*, Seiten 51–54, Palo Alto, CA, März 1998. AAAI Press.

[Dey00] Dey, Anind K.: *Providing Architectural Support for Building Context-Aware Applications.* Dissertation, College of Computing, Georgia Tech, Dezember 2000.

[Dey02] Dey, Anind K.: *Building Context-Aware Applications Evaluation of Ubiquitous Computing Systems: Exercise in Frustration or a Research Opportunity? Augmenting Humans via Objects and the Environment.* Technischer Bericht, Summer School on Ubiquitous and Pervasive Computing, Dagstuhl, Germany, August 2002.

[Dij68] Dijkstra, Edsger W.: *Go To Statement Considered Harmful.* Communications of the ACM, 11(3):147–148, März 1968.

[DJG+02] DeRose, Steven, Ron Daniel Jr., Paul Grosso, Eve Maler, Jonathan Marsh und Norman Walsh: *XML Pointer Language (XPointer).* W3c working draft WD-xptr-20020816, World Wide Web Consortium, August 2002.

[DR03] Dürr, Frank und Kurt Rothermel: *On a Location Model for Fine-grained Geocast.* In: Dey, Anind K., Albrecht Schmidt und Joseph F. McCarthy (Herausgeber): *UbiComp 2003: Ubiquitous Computing, 5th International Conference*, Nummer 2864 in *Lecture Notes in Computer Science*, Seattle, WA, USA, Oktober 2003. Springer Verlag.

[DRS00] Day, M., J. Rosenberg und H. Sugano: *A Model for Presence and Instant Messaging*. Request for Comments 2778, Internet Engineering Task Force, Februar 2000.

[DS98] Dawson, F. und D. Stenerson: *Internet Calendaring and Scheduling Core Object Specification (iCalendar)*. RFC 2445, November 1998.

[DSA99] Dey, Anind K., Daniel Salber und Gregory D. Abowd: *A context-based infrastructure for smart environments*. In: *In the Proceedings of the 1st International Workshop on Managing Interactions in Smart Environments (MANSE '99)*, Seiten 114–128, Dublin, Ireland, Dezember 1999. Springer Verlag.

[DSA01] Dey, Anind K., Daniel Salber und Gregory Abowd: *A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-aware Applications*. Human-Computer Interaction, 16, 2001.

[DW88] Durrant-Whyte, H. F.: *Sensor models and multisensor integration*. International Journal of Robotics Research, 7(6):97–113, Dezember 1988.

[DW90] Durrant-Whyte, H. F.: *Toward a fully decentralized architecture for multi-sensor data fusion*. In: *IEEE International Conference on Robotics and Automation*, Band 2, Seiten 1331–1336, Cincinnati, OH, USA, 1990.

[Elm02] Elmenreich, Wilfried: *Sensor Fusion in Time-Triggered Systems*. Dissertation, Technische Universität Wien, November 2002.

[Eti95] Etique, P.-A.: *Service Specification, Verification and Validation for the Intelligent Network*. Dissertation, Swiss Federal Institute of Technology, Lausanne, 1995.

[Euc03] Euchner, M.: *HMAC-authenticated Diffie-Hellman for MIKEY*. Internet Draft draft-ietf-msec-mikey-dhhmac-02, Internet Engineering Task Force, Juni 2003. Work in progress.

[Faa04] Faatz, Andreas: *Ein Verfahren zur Anreicherung fachgebiesspezifischer Ontologien durch Begriffsvorschläge*. Dissertation, Technische Universität Darmstadt, KOM, Darmstadt, November 2004.

[Fal00] *XML Schema Part 0: Primer*. W3c candidate recommendation CR-xmlschema-0-20001024, World Wide Web Consortium, Oktober 2000.

[FB96] Freed, N. und N. Borenstein: *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*. Request for Comments 2046, Internet Engineering Task Force, November 1996.

[FD02] F. Dietrich, J. P. Hubaux: *Formal methods for communication services: meeting the industry expectations*. Computer Networks, Seiten 99–120, Januar 2002.

[FDWB98] Friday, A., N. Davies, S. Wade und G. Blair: *L2imbo: A Distributed Systems Platform for Mobile Computing.* ACM Mobile Networks and Applications (MONET), Special Issue on Protocols and Software Paradigms of Mobile Networks, 3(2):143–156, August 1998.

[FF98] Franklin, David und Joshua Flaschbart: *All Gadget and No Representation Makes Jack a Dull Environment.* In: *AAAI 1998 Spring Symposium on Intelligent Environments*, Seiten 155–160, Palo Alto, CA, USA, März 1998. AAAI Press.

[FGM+99] Fielding, R., J. Gettys, J. Mogul, , H. Frystyk, L. Masinter, p. Leach und T. Berners-Lee: *Hypertext Transfer Protocol — HTTP/1.1.* Request for Comments 2616, Internet Engineering Task Force, Juni 1999.

[FH95] Foote, K. E. und D. J. Huebner: *Error, accuracy, and precision.* The geographer's craft project, Department of Geography, University of Texas at Austin, 1995.

[FHBH+99] Franks, J., P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen und L. Stewart: *HTTP Authentication: Basic and Digest Access Authentication.* Request for Comments 2617, Internet Engineering Task Force, Juni 1999.

[For04] Fortes, Victor: *Design and Implementation of a Fuzzy Logik System for Context Aggregation.* Diplomarbeit, Technische Universität Darmstadt, Darmstadt, September 2004.

[FZ98] Franklin, M. und S. Zdonik: *Data in Your Face : Push Technology in Perspective.* In: *ACM SIGMOD Intl. Conference on Management of Data (SIGMOD 98)*, Seattle, WA, USA, Juni 1998.

[GAMS03] Görtz, Manuel, Ralf Ackermann, Andreas Mauthe und Ralf Steinmetz: *Using Context Information to Avoid Service Interactions in IP Telephony.* In: Ventre, Giorgio und Roberto Canonico [VC03], Seiten 340–351, ISBN 3-540-20534-9. http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/GAMS03-1.html.

[GAS+03] Görtz, Manuel, Ralf Ackermann, Ralf Steinmetz, Harald Müller und Thomas Lederer: *Kombination von Lokationsinformationen und Kalendereinträgen, um eine Vorhersage eines zukünftigen Kontextes bzw. der Erreichbarkeit von Kommuikationspartner zu treffen.* Erfindungsmeldung, Dezember 2003.

[GAS04a] Görtz, Manuel, Ralf Ackermann und Ralf Steinmetz: *Enhanced SIP Communication Services by Context Sharing.* In: *30th EUROMICRO Conference 2004*, September 2004.

[GAS04b] Görtz, Manuel, Ralf Ackermann und Ralf Steinmetz: *The Digital Call Assistant: Determine Optimal Time Slots for Calls.* In: Roca, Vincent und Franck Rousseau (Herausgeber): *MIPS*, Band 3311 der Reihe *Lecture Notes in Computer Science*, Seiten 230–241. Springer, November 2004, ISBN 3-540-23928-6. http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/GAS04-2.html.

[GHM+03] Gudgin, Martin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau und Henrik Frystyk Nielsen: *SOAP Version 1.2 Part 1: Messaging Framework*. W3C Recommendation REC-soap12-part1-20030624, World Wide Web Consortium, Juni 2003.

[GJS92] Gehani, N. H., H. V. Jagadish und O. Shmueli: *Composite event specification in active databases: Model & implementation.* In: *Proceedings of the 18th International Conference on Very Large Databases*, 1992.

[GJS94] Gehani, N., H. Jagadish und O. Shmueli: *Advanced Database Concepts and Research Issues*, Kapitel Compose: A System for Composite Event Specification and Detection. Lecture Notes in Computer Science. Springer Verlag, 1994.

[GMCF95] Galvin, J., S. Murphy, S. Crocker und N. Freed: *Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted*. Request for Comments 1847, Internet Engineering Task Force, Oktober 1995.

[Gol97] Goldstein, E. Bruce: *Wahrnehmungspsychologie. Eine Einführung.* Spektrum Akademischer Verlag, Heidelberg, 1997.

[GPA+03] Görtz, Manuel, Alejandro Perez, Ralf Ackermann, Andreas Mauthe und RALF Steinmetz: *Location Sensing using RADAR.* Technischer Bericht TR-KOM-2003-09, Multimedia Communications Lab, Darmstadt University of Technology, September 2003.

[Gra98] Grand, Mark: *Patterns in Java Volume 1: A Catalog of Reusable Design Patterns Illustrated with UML.* John Wiley & Sons Ltd., Chichester, England, 1998.

[Gro98] Grossmann, P.: *Multisensor data fusion.* The GEC journal of Technology, 15:27–37, 1998.

[Guh91] Guha, R.V.: *Contexts: A Formalization and Some Applications.* Dissertation, Stanford, 1991.

[Gün79] Günther, Gotthard: *Life as poly-contexturality.* Beiträge zur Grundlegung einer operationsfähigen Dialektik, 1979.

[Gut84] Guttmann, A.: *r-trees: A dynamic index structure for spatial searching.* ACM, 1984.

[Hal92] Hall, David Lee: *Mathematical Techniques in Multi-Sensor Data Fusion.* Artech House, Norwood, Massachusetts, 1992.

[Han92] Hanson, Eric N.: *Rule condition testing and action execution in Ariel.* Seiten 49–58, 1992.

[HB01] Hightower, Jeffrey und Gaetano Borriello: *Location systems for ubiquitous computing.* Computer, 34(8):57–66, August 2001.

[HG94] Halgamuge, S. K. und M. Glesner: *Neural networks in designing fuzzy systems for real world applications.* Fuzzy Sets and Systems, 65, 1994.

[HJ98] Handley, M. und V. Jacobson: *SDP: Session Description Protocol.* Request for Comments 2327, Internet Engineering Task Force, April 1998.

[HKL+99] Hohl, Fritz, Uwe Kubach, Alexander Leonhardi, Kurt Rothermel und Markus Schwehm: *Next Century Challenges: Nexus an Open Global Infrastructure for Spatialaware Applications.* In: *The Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM 99)*, Seiten 249–255. ACM Press, 1999.

[HL01] Hall, David L. und James Llinas: *Handbook of Multisensor Data Fusion.* CRC Press, 2001.

[HNBR97] Hull, Richard, Philip Neaves und James Bedford-Robert: *Towards situated computing.* In: *Proceedings of the 1st International Symposium on Wearable Computers*, Seiten 146–153, Oktober 1997.

[Hol04] Hollick, Matthias: *Dependable Routing for Cellular and Ad hoc Networks.* Dissertation, Department of Electrical Engineering and Information Technology, Darmstadt University of Technology, Germany, Dezember 2004.

[Hou99] Housley, R.: *Cryptographic Message Syntax.* Request for Comments 2630, Internet Engineering Task Force, Juni 1999.

[HSSR99] Handley, M., Henning Schulzrinne, E. Schooler und Jonathan Rosenberg: *SIP: Session Initiation Protocol.* Request for Comments 2543, Internet Engineering Task Force, März 1999.

[HW03] Halpern, J. Y. und V. Weissman: *Using first-order logic to reason about policies.* In: *Proceedings of the Computer Security Foundations Workshop (CSFW'03)*, 2003.

[II95] II, R. B. Robrock: *The Intelligent Network – Changing the Face of Telecommunications.* Proceedings of the IEEE, 79(1), Januar 1995.

[Int] International Telecommunication Union: *Intelligent network.* Recommendation Q.12*nn*, Telecommunication Standardization Sector of ITU, Geneva, Switzerland.

[Int86] International Organization for Standardization: *Standard Generalized Markup Language (SGML).* Iso 8879:1986(E), Information processing – Text and Office Systems, Geneva, Oktober 1986.

[Int88] International Organization for Standardization: *LOTOS – A formal description technique based on temporal ordering of observation behavior.* Iso/tc97/sc21 98807:1988(IS), Information technology, Geneva, 1988.

[Int92a] International Organization for Standardization: *Basic Reference Model of Open Distrubuted Processing, Part 1: Overview and guide to use.* ISO/IEC JTC1/SC212/WG7 CD 10746-1, IEC, 1992.

[Int92b] International Organization for Standardization: *Z – Base Standard version 1.0.* Iso/iec jtc1/sc22, IEC, Geneva, November 1992.

[Int92c] International Telecommunication Union: *Intelligent Network — Global Functional Plane.* Recommendation Q.1203, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Oktober 1992.

[Int92d] International Telecommunication Union: *Principles of the Intelligent Network Architecture.* Recommendation Q.1201, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Oktober 1992.

[Int93a] International Telecommunication Union: *Global Functional Plane for Intelligent Network CS-1.* Recommendation Q.1213, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, März 1993.

[Int93b] International Telecommunication Union: *Introduction to Intelligent Network Capability Set 1.* Recommendation Q.1211, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, März 1993.

[Int96a] International Organization for Standardization: *Extended BNF.* Iso/iec 14977:1996(E), Information technology – Syntactic metalanguage, Geneva, 1996.

[Int96b] International Telecommunication Union: *Series H: Audiovisual and Multimedia Sytems – Visual Telephone Systems and Equipment for Local Area Networks which provide a non-guaranteed Quality of Service.* Recommendation H.323, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Mai 1996.

[Int97a] International Telecommunication Union: *Introduction to Intelligent Network Capability Set 2.* Recommendation Q.1221, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, September 1997.

[Int97b] International Telecommunication Union: *Series e: Overall network operation, telephone service, service operation and human factors – e.164: The international public telecommunication numbering plan.* Recommendation E.164, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Mai 1997.

[Int97c] International Telecommunication Union: *Series X: Data networks and Open System Communications – X.680: Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation.* Recommendation X.680, Standardization Sector of ITU, Geneva, Switzerland, Dezember 1997.

[Int98a] International Telecommunication Union: *Series H: Audiovisual and Multimedia Sytems – h.450.1: Generic functional protocol for the support of supplementary services in H.323.* Recommendation H.450.1, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Februar 1998.

[Int98b] International Telecommunication Union: *Series H: Audiovisual and Multimedia Sytems – h.450.3: Call diversion supplementary service for H.323.* Recommendation H.450.3, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Februar 1998.

[Int98c] International Telecommunication Union: *Series Q: Switching and Signalling – isdn user-network interface layer 3 specification for basic call control.* Recommendation Q.931, Standardization Sector of ITU, Geneva, Switzerland, Geneva, Switzerland, Mai 1998.

[Int99a] International Organization for Standardization: *Estelle – A formal description technique based on extended state transition model.* Iso/tc97/sc21 9074:1999(IS), Information Technology, Geneva, 1999.

[Int99b] International Telecommunication Union: *Series Z: Languages and General Software Aspects for Telecommunication Systems, Formal description techniques (FDT) – Message Sequence Chart.* Recommendation Z.120, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, März 1999.

[Int99c] International Telecommunication Union: *Series Z: Languages and General Software Aspects for Telecommunication Systems, Formal description techniques (FDT) – Specification and Description Language SDL.* Recommendation Z.100, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, November 1999.

[Int00a] International Telecommunication Union: *Series H: Audiovisual and Multimedia Sytems – H.245: Control Protocol for Multimedia Communication.* Recommendation, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Juni 2000.

[Int00b] International Telecommunication Union: *Series H: Audiovisual and Multimedia Sytems – H.248: Gateway Control Protocol.* Recommendation, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Juni 2000.

[Int03a] International Telecommunication Union: *Series H: Audiovisual and Multimedia Sytems – H.225.0: Call signalling protocols and media stream packetization for packet-based multimedia communication systems.* Recommendation, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Mai 2003.

[Int03b] International Telecommunication Union: *Series H: Audiovisual and Multimedia Sytems – H.323: Packet based Multimedia Communication Systems.* Recommendation, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Mai 2003.

[Jac03] Jacobsen, H.-Arno (Herausgeber): *2nd Intl. Workshop on Distributed Event-Based Systems (DEBS 03).* ACM Press, Juni 2003.

[JAMS00] Jain, R., F. M. Anjum, P. Missier und S. Shastry: *Java C all Control, Coordination, and Transactions.* IEEE Communications Magazine, 38, Januar 2000.

[Jav02] Java Community Process: *SIP Servlet API.* Java specification requests JSR 116, Java Community Process, Mai 2002.

[JDGK95] Jospeh, A., A. DeLespinasse, D. Gifford und M. F. Kaashoek: *Rover: A Toolkit for Mobile Information Access.* In: *Proceedings of the 15th ACM Symposium on Operating System Principles (SOSP)*, Band 29, Seiten 156–171, Copper Mountain Resort, CO, USA, Dezember 1995. ACM Press.

[JMM99] Jagadish, H. V., Alberto O. Mendelzon und Inderpal Singh Mumick: *Managing Conflicts Between Rules.* Journal of Computer and System Sciences, 58(1):13–28, 1999.

[JZ98] Jackson, Michael und Pamela Zave: *Distributed Feature Composition: A Virtual Architecture for Telecommunications Services.* Software Engineering, 24(10):831–847, 1998.

[Kag04] Kagal, Lalana: *A Policy-Based Approach to Governing Autonomous Behavior in Distributed Environments.* Dissertation, University of Maryland Baltimore County, September 2004.

[Kal60] Kalman, R. E.: *A New Approach to Linear Filtering and Prediction Problems.* Transaction of the ASME, Series D, Journal of Basic Engineering, 82:35–45, 1960.

[KB61] Kalman, R. E. und R. S. Bucy: *New Results in Linear Filtering and Prediction Theory.* Transaction of the ASME, Series D, Journal of Basic Engineering, 83:95–108, 1961.

[KBK00] Kristensen, Anders, Anders Byttner und Roman Kurmanowytsch: *Programming SIP Services.* In: *Proceedings of the 1st IP Telephony Workshop (IPtel 2000)*, Berlin, Germany, April 2000.

[KBM+00] Kindberg, T., J. Barton, J. Morgan, G. Becker, I. Bedner, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, C. Pering, J. Schettino, B. Serra und M. Spasojevic: *People, Places, Things: Web Presence for the Real World.* In: *WMCSA*, Monterey, California, USA, Dezember 2000.

[Kec00] Keck, Dirk Oliver: *Erkennung von Wechselwirkungen zwischen Mehrwertdiensten durch Analyse ihrer Konfigurationen und Protokollabläufe.* 79. Bericht über verkehrstheoretische Arbeiten, Universität Stuttgart, 2000.

[Kel02] Kellerer, Wolfgang: *Serverarchitektur zur netzunabhängigen Diensteuerung in heterogenen Kommunikationsnetzen.* Dissertation, Technische Universität München, Fakultät für Elektrotechnik und Informationstechnik, München, Januar 2002.

[Kes92] Kessler, O.: *Functional description of the data fusion process.* Technischer Bericht, Warminister. Office of Naval Technology, Naval Air Development Center, 1992.

[KFJ03] Kagal, Lalana, Tim Finin und Anupam Joshi: *A policy language for a pervasive computing environment.* In: *POLICY '03: Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, Seite 63, Washington, DC, USA, 2003. IEEE Computer Society.

[KK98] Keck, Dirk O. und Paul J. Kühn: *The Feature and Service Interaction Problem in Telecommunications Systems: A Survey.* IEEE Transactions on Software Engineering, 24(10):779–796, Oktober 1998.

[KKG+90] Kopetz, H., H. Kantz, G. Grünsteidl, P. Puschner und J. Reisinger: *Tolerating transient faults in mars.* In: *Proccedings of the 20th. Symposium on Fault Tolerant Computing*, Newcastle upon Tyne, UK, Juni 1990.

[KL03] Keller, Alexander und Heiko Ludwig: *The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services.* Source Journal of Network and Systems Management, 11(1):57 – 81, März 2003.

[Kle99] Klein, Lawrence A.: *Sensor and Data Fusion Concepts and Applications.* SPIE Optical Engineering Press, 2 Auflage, 1999,.

[Kli99] Klir, George J.: *Uncertainty and Information Measures for Imprecise Probability: An Overview.* In: *Proceedings of the first International Probabilities and Their Applications*, Gent, Belgium, Juni 1999.

[KM00] Kristol, D. und L. Montulli: *Http state management mechanism.* Request for Comments 2965, Internet Engineering Task Force, Oktober 2000.

[KMB96] Kock, N.F., R.J. McQueen und M. Baker: *Learning and Process Improvement in Knowledge Organisations: A Critical Analysis of Four Contemporary Myths.* The Learning Organization, 3, 1996.

[Koh89] Kohonen, Teuvo: *Self-Organization and Associative Memory.* Springer Verlag, Berlin, Germany, 3 Auflage, 1989.

[KZK97] Kam, M., X. Zhu und P. Kalata: *Sensor Fusion for Mobile Robot Navigation.* In: *Proceedings of the IEEE*, Band 85, Seiten 108–119, Januar 1997.

[Lae99] Laerhoven, K. Van: *Online adaptive context awareness, starting with lowlevel sensors.* Licentiaat in de informatica, Free University of Brussels (VUB), Brussels, Mai 1999.

[LC00] Laerhoven, K. Van und O. Cakmakci: *What shall we teach our pants?* In: *Proc. of the Fourth Internation Symposium on Wearable Computers, ISWC 2000*, Atlanta, GA, USA, 2000. IEEE Computer Society Press.

[LCB99] Liebig, C., M. Cilia und A. Buchmann: *Event Composition in Time-dependent Distributed Systems.* In: *Proceedings 4th IFCIS Conference on Cooperative Information Systems (CoopIS 99)*, Seiten 70 – 78, Edinburgh, Scotland, September 1999. IEEE Computer Society Press.

[Len04] Lennox, Jonathan Michael: *Services for Internet Telephony.* Dissertation, Columbia University, New York, USA, Dezember 2004.

[Leo98] Leonhardt, U.: *Supporting Location-Awareness in Open Distributed Systems.* Dissertation, Department of Computing, Imperial College of Science, University of London, 1998.

[LH98] Llinas, James und David L. Hall: *An introduction to multi-sensor data fusion.* In: *Proceedings of the 1998 IEEE International Symposium on Circuits and Systems*, Band 6, Seiten 537–540, Mai-Jun 1998.

[Lib97] Libes, D.: *Writing a Tcl Extension in Only ... 7 Years.* In: *Proceedings of the Fifth Annual Tcl/Tk Workshop*, Seiten 108–119, Juli 1997.

[Lil00] Lilley, J.: *Scalability in an Intentional Naming System.* Diplomarbeit, Massachusetts Institute of Technology, Juni 2000.

[LK89] Luo, R. C. und M. Kay: *Multisensor integration and fusion in intelligent systems.* IEEE Transactions on Systems, Man, and Cybernetics, 19(5):901–930, Sep.-Okt. 1989.

[LKRF99] Leonhardi, A., U. Kubach, K. Rothermel und A. Fritz: *Virtual Information Towers - A Metaphor for Intuitive, Location-Aware Information Access in a Mobile Environment.* Technischer Bericht, 1999.

[LMD96] Leonhardt, U., J. Magee und P. Dias: *Location service in mobile computing environments.* In: *Computer & Graphics. Special Issue on Mobile Computin*, Band 20, Sep/Oct 1996.

[LRS01] Lennox, Jonathan, Jonathan Rosenberg und Henning Schulzrinne: *Common Gateway Interface for SIP.* Request for Comments 3050, Januar 2001.

[LS99] Lupu, Emil C. und Morris Sloman: *Conflicts in Policy-Based Distributed Systems Management.* IEEE Trans. Softw. Eng., 25(6):852–869, 1999.

[LS00a] Lennox, Jonathan und Henning Schulzrinne: *Call Processing Language Framework and Requirements.* Request for Comments 2824, Mai 2000.

[LS00b] Lennox, Jonathan und Henning Schulzrinne: *Feature interaction in internet telephony.* In: *Feature Interaction in Telecommunications and Software Systems VI*, Glasgow, UK, Mai 2000.

[Łuk13] Łukasiewicz, Jan: *Die logischen Grundlagen der Wahrscheinlichkeitsrechnung.* Krakow, 1913.

[Łuk20] Łukasiewicz, Jan: *O logice trójwartościowej (Über die dreiwertige Logik).* Ruch Filozofiezny, 5:170 – 171, 1920.

[LWS04] Lennox, Jonathan, X. Wu und Henning Schulzrinne: *Call Processing Language (CPL): A Language for User Control of Internet Telephony Services.* Request for Comments 3880, Oktober 2004.

[MA75] Mamdani, E. H. und S. Assilian: *An experiment in linguistic synthesis with a fuzzy logic controller.* Int. Journal of Man-Maschines Studies, 7:1 – 13, 1975.

[Mah03] Mahy, R.: *Connection Reuse in the Session Initiation Protocol (SIP).* Internet Draft draft-ietf-sip-connect-reuse-00, Internet Engineering Task Force, August 2003. Work in progress.

[May04] Mayrhofer, Rene: *An Architecture for Context Prediction.* Dissertation, Johannes Kepler University of Linz, Austria, October 2004.

[MB97] McCarthy, John und Saša Buvač: *Formalizing Context (Expanded Notes).* Computing Natural Language, 1997.

[MBC+04] McGlashan, Scott, Daniel C. Burnett, Jerry Carter, Peter Danielsen, Jim Ferrans, Andrew Hunt, Bruce Lucas,

Brad Porter, Ken Rehor und Steph Tryphonas: *Voice Extensible Markup Language (VoiceXML) Version 2.0.* W3c recommendation REC-voicexml20-20040316, World Wide Web Consortium, März 2004.

[MGA+04] Martinovic, Ivan, Manuel Görtz, Ralf Ackermann, Andreas Mauthe und Ralf Steinmetz: *Trust and Context: Two Complementary Concepts for Creating Spontaneous Collaborative Networks and Intelligent Applications.* In: *Proceedings of SoftCOM'04, International Conference on Software, Telecommunications and Computer Networks, Croatia/Italy*, Oktober 2004. http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/MGA+04-1.html.

[Mit98] Mitchell, Keith: *Developing a Context Sensitive Tourist Guide.* In: *Proceedings of the Fifth Cabernet Radicals Workshop*, Valadares, Porto, Portugal, Juli 1998.

[Mit02] Mitchell, K.: *Supporting the Development of Mobile Context-Aware Computing.* Dissertation, Department of Computing, Lancaster University, Januar 2002.

[Mos02] Moschgath, Marie-Luise.: *Kontextabhängige Zugriffskontrolle für Anwendungen im Ubiquitous Computing.* Dissertation, Technische Universität Darmstadt, Darmstadt, 2002.

[MP93] Matos, Gilberto und James Purtilo: *Reconfiguration of Hierarchical Tuple-Spaces: Experiments with Linda-Polylith.* Technical report, University of Maryland Institute for Advanced Computer Studies, 1993.

[MP95] Motschnig-Pitrik, Renate: *An Integrated View on the Viewing Abstraction: Contexts and Perspectives in Software Development, AI, and Databases.* Journal of Systems Integration, 5(1):23–60, 1995.

[MRF03] Mayrhofer, R., H. Radi und A. Ferscha: *Recognizing and predicting context by learning from user behavior.* In: G. Kotsis, A. Ferscha, W. Schreiner und K. Ibrahim (Herausgeber): *Proceedings of the International Conference On Advances in Mobile Multimedia (MoMM2003)*, Band 171, Seiten 25–35, Jakarta, Indonesia, September 2003. Austrian Computer Society (OCG), ISBN 3-85403-171-8.

[Müh02] Mühl, Gero: *Large-Scale Content-Based Publish/Subscribe Systems.* Dissertation, Technische Universität Darmstadt, Darmstadt, Darmstadt, Germany, 2002.

[Mül96] Müller, Harald: *Flexible Signalisierungsarchitekturen für Multimediadienste mit heterogenen Endgeräten.* Dissertation, Technische Universität München, Fakultät für Elektrotechnik und Informationstechnik, München, 1996.

[Mur96] Murphy, R. R.: *Biological and cognitive foundations of intelligent sensor fusion.* IEEE Transactions on Systems, Man and Cybernetics, 21(1):42–51, 1996.

[Nei76] Neisser, Ulric: *Cognition and reality.* San Francisco, 1976. Morgan Kaufmann.

[Nel98a] Nelson, Giles: *Context-Aware and Location System.* Dissertation, University of Cambridge, Januar 1998.

[Nel98b] Nelson, Giles: *Context-Aware and Location System.* Dissertation, University of Cambridge, Januar 1998.

[Nel99] Nelles, Oliver: *Nonlinear System Identification with Local Linear Neuro-Fuzzy Models.* Dissertation, Regelungstechnik und Prozessautomatisierung,Technische Universität Darmstadt, Darmstadt, 1999.

[NKK97] Nauck, D., F. Klawonn und R. Kruse: *Foundations on Neuro-Fuzzy Systems.* Wiley, 1997.

[NLiMK03] Nakamura, Masahide, Pattara Leelaprute, Ken ichi Matsumoto und Tohru Kikuno: *Semantic Warnings and Feature Interaction in Call Processing Language on Internet Telephony.* In: *Symposium on Applications and the Internet*, Orlando, FL, USA, Januar 2003.

[NLiMK04] Nakamura, Masahide, Pattara Leelaprute, Ken ichi Matsumoto und Tohru Kikuno: *On Detecting Feature Interactions in Programmable Service Environment of Internet Telephony.* Journal of Computer Networks, 45(5):605–624, 2004.

[NM01] Nicklas, Daniela und Bernhard Mitschang: *The Nexus Augmented World Model: An Extensible Approach for Mobile, Spatially-aware Applications.* In: Wang, Yingxu, Shushma Patel und Ronald Johnston (Herausgeber): *Proceedings of the 7th International Conference on Object-Oriented Information Systems : OOIS 01*, Seiten 392–401, Calgary, Canada, August 2001.

[Nor98] Norman, A. D.: *The invisible computer*, 1998.

[OPSS93] Oki, Brian, Manfred Pfluegl, Alex Siegel und Dale Skeen: *The Information Bus – An Architecture for Extensible Distributed Systems.* In: Liskov, Barbara (Herausgeber): *Proceedings of the Fourteenth ACM Symposium on Operating System Principles*, Seiten 58–68, Asheville, NC, USA, Dezember 1993. ACM Press.

[OT03] Ono, K. und S. Tachimoto: *Requirements for End-to-middle Security for the Session Initiation Protocol (SIP).* Internet Draft draft-ietf-sipping-end2middle-security-reqs-00, Internet Engineering Task Force, Oktober 2003. Work in progress.

[Ous94] Ousterhout, John: *Tcl and the Tk Toolkit.* Addison-Wesley, 1994.

[Ous98] Ousterhout, John K.: *Scripting: Higher-Level Programming for the 21st Century.* Computer, 31(3):23–30, 1998.

[Pas98] Pascoe, Jason: *Adding Generic Contextual Capabilities to Wearable Computers.* In: *In the Proceedings of the 2nd IEEE International Symposium on Wearable Computers (ISWC'98)*, Seiten 92–99, Pittsburgh, PA, Oktober 1998. IEEE.

[Pas01] Pascoe, Jason: *Context-Aware Software.* Dissertation, Computing Laboratory, University of Kent at Canterbury, August 2001.

[Pat99] Paton, Norman W.: *Active Rules in Database Systems.* Springer-Verlag, Heidelberg, 1999.

[PB02] Pietzuch, Peter R. und Jean Bacon: *Hermes: A Distributed Event-based Middleware Architecture.* In: Bacon, Jean, Ludger Fiege, Rachid Guerraoui, H.-Arno Jacobsen und Gero Mühl (Herausgeber): *1st Intl. Workshop on Distributed Event-Based Systems (DEBS 02)*, Vienna, Austria, Juni 2002. IEEE Computer Society Press.

[PBT+04] Petzold, Jan, Faruk Bagci, Wolfgang Trumler, Theo Ungerer und Lucian Vintan: *Global State Context Prediction Techniques Applied to a Smart Office Building.* In: *In Proceedings for The Communication Networks and Distributed Systems Modeling and Simulation Conference*, San Diego, CA, USA, Januar 2004.

[PBTU03] Petzold, Jan, Faruk Bagci, Wolfgang Trumler und Theo Ungerer: *The State Predictor Method for Context Prediction.* In: *In Adjunct Proceedings Fifth International Conference on Ubiquitous Computing*, Seattle, WA, USA, Oktober 2003.

[Pie04] Pietzuch, Peter Robert: *A Scalable Event-Based Middleware.* Dissertation, Department of Computer Science, Queens College, University of Cambridge, Februar 2004.

[Pro92] Provan, G. M.: *The validity of dempster-shafer belief functions.* International Journal of Approximate Reasoning, 6:389–399, 1992.

[PSB03] Pietzuch, Peter R., Brian Shand und Jean Bacon: *A Framework for Event Composition in Distributed Systems.* In: *Proceedings of the 2003 International Middleware Conference*, Seiten 62–82, Rio de Janeiro, Brazil, Juni 2003. Springer Verlag.

[Rao98] Rao, N. S. V.: *A fusion method that performs better than best sensor.* Proceedings of the First International Conference on Multisource-Multisensor Information Fusion, 1:19–26, 1998.

[RCDD98] Rodden, Tom, Keith Cheverst, Nigel Davies und Alan Dix: *Exploiting context in HCI design for mobile systems.* In: *Workshop on Human Computer Interaction with Mobile Devices*, Mai 1998.

[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, Germany, November 2001. Springer Verlag.

[Ree95] Reed, R.: *Technolgies for Service Engineering.* In: A. Clarke, *et al.* (Herausgeber): *Bringing Telecommunication Services to People (IS&N'95)*, Nummer 998 in *Lecture Notes in Computer Science*, Seiten 290–291, Berlin, Germany, 1995. Springer Verlag.

[RKCD01] Rowstron, Antony, Anne-Marie Kermarrec, Miguel Castro und Peter Druschel: *Scribe: The design of a large-scale event notification infrastructure.* In: *Third International Workshop on Networked Group Communication (NGC 2001)*, Seiten 30–43, London, UK, November 2001. Springer Verlag.

[RM02] Reiff-Marganiec, Stephan: *Runtime Resolution of Features Interactions in Evolving Telecommunications Systems.* Dissertation, Department of Computing Science, University of Glasgow, Glasgow (UK), Mai 2002.

[RMT04] Reiff-Marganiec, S. und K. J. Turner: *Feature Interaction in Policies.* Computer Networks, 45(5):569–584, August 2004.

[Roa02] Roach, A. B.: *Session Initiation Protocol (SIP)-Specific Event Notification.* Request for Comments 3265, Internet Engineering Task Force, Juni 2002.

[Roc97] Rocha, Luis Mateus: *Relative Uncertainty and Evidence Sets: A Constructivist Framework.* International Journal of General Systems, 26(1-2):35–61, 1997.

[Roe02] Roedig, Utz: *Firewall-Architekturen für Multimedia-Applikationen.* Dissertation, Technische Universität Darmstadt, KOM, Darmstadt, 2002.

[Ros02] Rosenberg, Jonathan: *The Session Initiation Protocol (SIP) UPDATE Method.* Request for Comments 3311, Internet Engineering Task Force, September 2002.

[RPM98] Ryan, Nick, Jason Pascoe und David Morse: *Enhanced reality fieldwork: the context-aware archaeological assistant.* In: Gaffney, V., M. van Leusen und S. Exxon (Herausgeber): *Computer Applications and Quantitative Methods in Archaeology.* Oxford, 1998.

[RS02] Rosenberg, Jonnathan und Henning Schulzrinne: *An offer/answer model with the session description protocol (sdp).* Request for Comments 3264, Internet Engineering Task Force, Juni 2002.

[RSC+02] Rosenberg, Jonathan, Henning Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley und E. Schooler: *SIP: Session Initiation Protocol*. Request for Comments 3261, Internet Engineering Task Force, Juni 2002.

[Rus23] Russel, B.: *Vagueness*. The Australasian Journal of Psychology and Philosophy, 1:84 – 92, 1923.

[SA04] Saint-Andre, P.: *Extensible Messaging and Presence Protocol (XMPP): Core*. Request for Comments 3920, Internet Engineering Task Force, Oktober 2004.

[SA98] Salber, Daniel und Gregory Abowd: *The Design and Use of a Generic Context Server*. In: *Proceedings of the Perceptual User Interfaces Workshop*, Seiten 63–66, November 98.

[SAG+93] Schilit, Bill N., Norman Adams, Rich Gold, Michael Tso und Roy Want: *The PARCTAB Mobile Computing System*. In: *Proceedings of the Fourth Workshop on Workstation Operating Systems (WWOS-IV)*, Seiten 34–39, Napa, CA, USA, Oktober 1993. IEEE Computer Society.

[Sal00] Salber, Daniel: *Context-Awareness and Multimodality*. In: *Colloque sur la multimodalité*. IMAG, Grenoble, Mai 2000.

[SAT+99] Schmidt, A., K. A. Aidoo, A. Takaluoma, U. Tuomela, K. Van Laerhoven und W. Van de Velde: *Advanced Interaction in Context*. Lecture Notes in Computer Science, 1707, 1999.

[SAW94] Schilit, Bill, Norman Adams und Roy Want: *Context-aware computing applications*. In: *IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, US, 1994.

[SBG99] Schmidt, Albrecht, Michael Beigl und Hans-W. Gellersen: *There is more to context than location*. Computers and Graphics, 23(6):893–901, 1999.

[SBW99] Steinberg, A. N., C. L. Bowman und F. E. White: *Revisions to the jdl data fusion model*. In: *Proceedings of the 1999 IRIS Unclassified National Sensor and Data Fusion Conference (NSSDF)*, Mai 1999.

[SCFJ96] Schulzrinne, Henning, S. Casner, R. Frederick und V. Jacobson: *RTP: A Transport Protocol for Real-Time Applications*. Request for Comments 1889, Internet Engineering Task Force, Januar 1996.

[Sch95] Schilit, William Noah: *A System Architecture for Context-Aware Mobile Computing*. Dissertation, Columbia University, 1995.

[Sch99] Schwartz, E.: *Design and implementation of intentional names*. Diplomarbeit, Massachusetts Institute of Technology, Mai 1999.

[Sch01] Schmitt, Jens Burkhard: *Heterogeneous Network Quality of Service Systems*. Kluwer Academic Publishers, Juni 2001. ISBN 0-793-7410-X.

[Sch02a] *Web Services Description Requirements.* W3c working draft WD-ws-desc-reqs-20021028, World Wide Web Consortium, Oktober 2002.

[Sch02b] Schmidt, Albrecht: *Ubiquitous Computing — Computing in Context.* Dissertation, Lancaster University, UK, November 2002.

[SDA99] Salber, Daniel, Anind K. Dey und Gregory Abowd: *The Context Toolkit: Aiding the development of context-enabled applications.* In: *ACM Conference on Human Factors in Computer Systems (CHI '99)*, Seiten 434–441, Pittsburgh, PA, USA, Mai 1999. ACM Press.

[SE01] Stillman, Scott und Irfan Essa: *Toward reliable mulitmodal sensing in aware environments.* In: *Perceptual User Interfaces (PUI 2001) Workshop (held in conjunction with ACM UIST 2001 Conference)*, Orlando, Florida„ November 2001.

[SF02] Sentz, Kari und Scott Ferson: *Combination of Evidence in Dempster-Shafer Theory.* Technischer Bericht, Los Alamos National Laboratory, Los Alamos, NM, April 2002.

[SFK+04] Sugano, H., S. Fujimoto, G. Klyne, A. Bateman, W. Carr und J. Peterson: *Presence Information Data Format (PIDF).* Request for Comments 3863, Internet Engineering Task Force, August 2004.

[SG01] Schmidt, Albrecht und Hans-Werner Gellersen: *Modell, Architektur und Plattform für Informationssysteme mit Kontextbezug.* Infromatik Forschung und Entwicklung, 16(4):213 – 224, 2001.

[SGKR04] Schulzrinne, H., V. Gurbani, P. Kyzivat und J. Rosenberg: *RPID: Rich Presence Extensions to the Presence Information Data Format (PIDF).* Internet Draft draft-ietf-simple-rpid-04, Internet Engineering Task Force, Oktober 2004. Work in progress.

[Sha76] Shafer, Glenn: *A Mathematical Theory of Evidence.* Princeton University Press, Princeton, 1976.

[Sho01] Showalter, T.: *Sieve: A mail filtering language.* RFC 3028, Januar 2001.

[SK98] Seidl, T. und H. P. Kriegel: *Optimal multi-step k-nearest neighbor search.* In: *ACM SIGMOD*, 1998.

[SMDH98] Silverberg, S., S. Mansour, F. Dawson und R. Hopson: *iCalendar Transport-Independent Interoperability Protocol(iTIP) Scheduling Events, BusyTime, To-dos and Journal Entries.* Request for Comments 2446, Internet Engineering Task Force, November 1998.

[SN99] Schneider, Jean-Guy und Oscar Nierstrasz: *Components, Scripts and Glue.* In: Barroca, Leonor, Jon Hall und Patrick Hall (Herausgeber): *Software Architectures – Advances and Applications*, Seiten 13–25. Springer-Verlag, 1999, ISBN 1-85233-636-6.

[SOC02] Schulzrinne, H., D. Oran und G. Camarillo: *The Reason Header Field for the Session Initiation Protocol (SIP)*. Request for Comments 3326, Internet Engineering Task Force, Dezember 2002.

[Spa03] Sparks, R.: *The Session Initiation Protocol (SIP) Refer Method*. Request for Comments 3515, Internet Engineering Task Force, April 2003.

[SRC84] Salzer, J. H., D. P. Reed und D. Clarke: *End-to-End Arguments in System Design*. ACM Transactions on Computer Systems, 2(4):277 – 288, 1984.

[Sri95] Srinivasan, R.: *RPC: Remote Procedure Call Protocol Specification Version 2*. Request for Comments 1831, Internet Engineering Task Force, August 1995.

[SSO83] Swinehart, Daniel C., L.C. Stewart und S.M. Ornstein: *Adding Voice to an Office Computer Network*. In: *IEEE GlobeCom*, November 1983.

[ST85] Sugeno, M. und T. Takagi: *Fuzzy identification of systems and its applications to modeling and control*. IEEE Trans. on Systems Man & Cybernetics, 15:116 – 132, 1985.

[Ste98] Stevens, W. Richard: *UNIX Network Programming, Volume 1: Networking APIs: Sockets and XTI*. Prentice Hall PTR, Upper Saddle River, NJ, 2 Auflage, 1998.

[Ste99] Steinmetz, Ralf: *Multimedia Technologie*. Springer Verlag, 3 Auflage, 1999.

[STM00] Schmidt, Albrecht, Antii Takaluoma und Jani Mntyjrvi: *Context-aware telephony over wap*. In: *4. Handheld and Ubiquitous Computing (HUC2k)*. Springer-Verlag, London, Ltd., Personal Technologies, September 2000. Short Paper.

[Sug85] Sugeno, M.: *An Introductory Survey of Fuzzy Control*. Information Science, 36:59 – 83, 1985.

[Sun87] Sun Microsystems, Inc.: *XDR: External Data Representation standard*. Request for Comments 1014, Internet Engineering Task Force, Juni 1987.

[SW04] Steinmetz, Ralf und Klaus Wehrle: *Peer-to-Peer-Networking and -Computing*. Informatik Spektrum, Aktuelles Schlagwort, 27(1):51–54, 2004.

[TBF+03] Terpstra, Wesley W., Stefan Behnel, Ludger Fiege, Andreas Zeidler und Alejandro P. Buchmann: *A Peer-to-Peer Approach to Content-based Publish/Subscribe*. In: Jacobsen, H.-Arno [Jac03].

[TDM97] Troost, R., S. Dorner und K. Moore: *Communication Presentation Information in Internet Messages: The Content-Disposition Header Field*. Request for Comments 2183, Internet Engineering Task Force, August 1997.

[TIN97] TINA-Consortium: *TINA-C Glossary of Terms*, 1997. `http://www.tinac.com`.

[Tur02] Turner, Kenneth J.: *Modelling SIP Services*. In: *Proc. Formal Techniques for Networked and Distributed Systems (FORTE XV)*, Band 2529 der Reihe *Lecture Notes in Computer Science*, Seiten 162–177, Berlin, Germany, November 2002. Springer Verlag.

[Tur03] Turner, Kenneth J.: *Representing New Voice Services and Their Features*. In: Koenig, Harmut, Monica Heiner und Adam Wolisz (Herausgeber): *Proc. Formal Techniques for Networked and Distributed Systems (FORTE XVI)*, Lecture Notes in Computer Science, Seiten 123–140, Berlin, Germany, September 2003. Springer Verlag.

[VC03] Ventre, Giorgio und Roberto Canonico (Herausgeber): *Interactive Multimedia on Next Generation Networks, First International Workshop on Multimedia Interactive Protocols and Systems, MIPS 2003, Napoli, Italy, November 18-21, 2003, Proceedings*, Band 2899 der Reihe *Lecture Notes in Computer Science*. Springer, November 2003, ISBN 3-540-20534-9.

[vdM73] Malsburg, Christoph von der: *Self-organization of Orientation Sensitive Cells in the Striate Cortex*. Kybernetic, 14:85–100, 1973.

[vN56] Neumann, J. von: *Automata Studies*, Kapitel Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components, Seiten 43–98. Princeton University Press, 1956.

[Vuo95] Vuorimaa, Petri: *Fuzzy Self-Organizing Map, and Its Applications*. Dissertation, Signal Processing Laboratory,Tampere University of Technology, 1995.

[Wal98] Wald, L.: *A european proposal for terms of reference in data fusion*. International Archives of Photogrammetry and Remote Sensing, XXXII(Part 7):651–654, 1998.

[WB98] Weiser, M. und J. S. Brown: *The Coming Age of Calm Technology*. In: *In P. J. Denning & R. M. Metcalfe (Eds.), Beyond calculation: The next fifty years of computing*, Seiten 75–85, New York, NY, 998. http://www.ubiq.com/hypertext/weiser/acmfuture2endnote.htm.

[WC95] Widom, Jennifer und Stefano Ceri (Herausgeber): *Active Database Systems: Triggers and Rules For Advanced Database Processing*. Morgan-Kaufmann, San Mateo, California, USA, 1995.

[Wei91] Weiser, M.: *The computer for the 21st century.* Scientific American, 265(3):94–104, September 1991.

[Wej00] Wejchert, J.: *The disappearing computer, information document.* In: *IST Call for proposals, European Commission, Future and Emerging Technologies*, Februar 2000. http://www.disappearing-computer.net/mission.html.

[Wil92] Wille, Rudolf: *Concept Lattices and Conceptual Knowledge Systems.* Computers and Mathematics with Applications, 23, 1992.

[WJH97] Ward, Andy, Alan Jones und Andy Hopper: *A new location technique for the active office.* IEEE Personal Communications, 4(5):42–47, Oktober 1997.

[WL90] Waltz, E. und J. Llinas: *Multisensor Data Fusion.* Artech House, Norwood, Massachusetts, 1990.

[WS03] Wu, Xiaotao und Henning Schulzrinne: *Programmable End System Services Using SIP.* In: *IEEE International Conference on Communications (ICC)*, Mai 2003.

[WS05] Wu, Xiaotao und Henning Schulzrinne: *LESS: Language for End System Services in Internet Telephony.* Internet Draft draft-wu-iptel-less-00, Internet Engineering Task Force, Februar 2005. Work in progress.

[WSSY02] Wu, Huadong, Mel Siegel, Rainer Stiefelhagen und Jie Yang: *Sensor Fusion Using Dempster-Shafer Theory.* In: *IEEE International Measurement Technology Conference (IMTC)*, Anchorage AK, USA, 2002.

[Wu03] Wu, Huadong: *Sensor Data Fusion for Context-Aware Computing Using Dempster-Shafer Theory.* Dissertation, The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, Dezember 2003.

[YAS03] Youssef, Moustafa, Ashok Agrawala und Udaya Shankar: *WLAN Location Determination via Clustering and Probability Distributions.* In: *IEEE PerCom 2003*, März 2003.

[YEO96] Yamada, M., S. Esaki und T. Omiya: *A Study on IN Basic Call State Model for Packet Switched Network.* In: *IEEE Intelligent Network Workshop IN'96*, Seiten 418–422, April 1996.

[YYDW99] Yang, Jie, Weiyi Yang, Matthias Denecke und Alex Waibel: *Smart Sight: A Tourist Assistant System.* In: *ISWC*, Seiten 73–78, 1999.

[Zad65] Zadeh, Lotfi A.: *Fuzzy Sets.* Information Control, 8:338–353, 1965.

[Zad94] Zadeh, Lotfi A.: *Soft Computing and Fuzzy Logic.* IEEE Software, 6(11):48–56, 1994.

[Zav93] Zave, Pamela: *Feature Interactions and Formal Specifications in Telecommunications.* IEEE Computer, 26(8):20–31, August 1993.

[Zei04] Zeidler, Andreas: *A Distributed Publish/Subscribe Notification Service for Pervasive Environments.* Dissertation, Technische Universität Darmstadt, Darmstadt, Darmstadt, Germany, August 2004.

[ZJ99a] Zave, Pamela und Michael Jackson: *DFC modifications I (Version 2): Routing extensions.* Technical report, AT&T Laboratories, November 1999.

[ZJ99b] Zave, Pamela und Michael Jackson: *DFC modifications II: Protocol extensions.* Technical report, AT&T Laboratories, November 1999.

[ZJ00] Zave, Pamela und Michael Jackson: *New feature interactions in mobile and multimedia telecommunication services.* In: Calder, Muffy und Evan Magill (Herausgeber): *Feature Interactions in Telecommunications and Software Systems VI*, Seiten 51–66, Amsterdam, The Netherlands, Mai 2000. IOS Press.

Online Referenzen

[www1] *Apache jakarta tomcat.* http://jakarta.apache.org/tomcat/.

[www2] *Apache xindice.* http://xml.apache.org/xindice/.

[www3] *Brigsoft – Active work tracker.* http://brigsoft.com/bsactivity/.

[www4] *Com: Component object model technologies.* http://www.microsoft.com/com.

[www5] *Distributed computing environment (dce).* http://www.opengroup.org/dce/.

[www6] *Doxygen.* http://www.doxygen.org/.

[www7] *The expect home page.* http://expect.nist.gov/.

[www8] *Fundacion grupo Eroski – Iluminar bien para ver mejor.* http://ideasana.fundaciongrupoeroski.es/web/es/04/iluminar/.

[www9] *gsoap: C/c++ web services and clients.* http://gsoap2.sourceforge.net/.

[www10] *Instant messaging and presence protocol (impp)).* http://www.ietf.org/html.charters/OLD/impp-charter.html.

[www11] *Java remote method invocation (java rmi).* http://java.sun.com/products/jdk/rmi.

[www12] *Java web services developer pack (java wsdp).* http://java.sun.com/webservices/jwsdp/index.jsp.

[www13] *Microchip.* http://www.microchip.com.

[www14] *Mysql.* http://www.mysql.com/.

[www15] *Oasis.* http://www.oasis-open.org.

[www16] *Object management group.* http://www.omg.org/.

[www17] *Open geospatial consortium, inc. (ogc).* http://www.opengeospatial.org.

[www18] *The parlay group.* http://www.parlay.org, http://www.parlay.org.

[www19] *Phyton qt binding.* http://www.riverbankcomputing.co.uk/pyqt.

[www20] *Power Per Unit Area of Surface.* http://hyperphysics.phy-astr.gsu.edu/hbase/vision/areance.html.

[www21] *Projekt x-ing.* http://www.informatik.hu-berlin.de/~xing/.

[www22] *Python programming language.* http://www.python.org.

[www23] *Session initiation protocol (sip).* http://www.ietf.org/html.charters/sip-charter.html.

[www24] *SIP Express Router.* http://www.iptel.org/ser.

[www25] *Sip for instant messaging and presence leveraging extensions (simple).* http://www.ietf.org/html.charters/simple-charter.html.

[www26] *Sun microsystems, jain apis for integrated networks.* http://java.sun.com/products/jain.

[www27] *The sunbird project - standalone calendar.* http://www.mozilla.org/projects/calendar/sunbird.html.

[www28] *Tamino xml database.* http://www1.softwareag.com/corporate/products/tamino/default.as%p.

[www29] *Tcl developer xchange!* http://www.tcl.tk/.

[www30] *Trolltech – Cross-platform C++ GUI Development.* http://www.trolltech.com/.

[www31] *Using castor xml.* http://www.castor.org/xml-framework.html.

[www32] *Webservices - axis.* http://ws.apache.org/axis/.

[www33] *Xilinx: Programmable logic devices.* http://www.xilinx.com/.

[www34] *Your Source for Open Source Communication – VOCAL.* http://www.vovida.org.

[www35] Tourrilhes, Jean: *Wireless Tools for Linux.* http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html%.

[www36] Youssef, Moustafa A.: *Mwvlan: A New GPL Driver for the Wavelan IEEE/Orinoco.* http://www.cs.umd.edu/~moustafa/mwvlan/mwvlan.html.

Publikationen des Autors

Konferenzbeiträge

[1] Ackermann, Ralf, Vasilios Darlagiannis, Manuel Görtz, Martin Karsten und Ralf Steinmetz: *An Open Source H.323-SIP Gateway as Basis for Supplementary Service Interworking*. In: *Proceedings of the 2nd IP Telephony Workshop, New York*, Seiten 169–175, April 2001. http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/ADG+01-1.html.

[2] Ackermann, Ralf, Manuel Görtz, Martin Karsten und Ralf Steinmetz: *Prototyping a PDA based Communication Appliance*. In: *Proceedings of Softcom 2001, Split, Croatia*, Oktober 2001. http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/AGKS01-1.html.

[3] Roedig, Utz, Manuel Görtz, Martin Karsten und Ralf Steinmetz: *RSVP as Firewall Signalling Protocol*. In: *Proceedings of the 6th IEEE Symposium on Computers and Communications, Hammamet, Tunisia*, Seiten 57–62. IEEE, Juli 2001. http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/RGKS01-1.html.

[4] Görtz, Manuel, Ralf Ackermann, Martin Karsten und Ralf Steinmetz: *Using a Multi-Layer Approach to tackle the Service Interaction Problem in Telephony Scenarios*. In: *Proceedings of EuroMicro 2002*, September 2002. http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/MGKS02-1.html.

[5] Görtz, Manuel, Ralf Ackermann, Andreas Mauthe und Ralf Steinmetz: *A Prototype Setup for Location-aware Personal Communication Services*. In: *Evolute Workshop 2003*, November 2003. http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/GAMS03-2.html.

[6] Görtz, Manuel, Ralf Ackermann, Andreas Mauthe und Ralf Steinmetz: *Using Context Information to Avoid Service Interactions in IP Telephony*. In: *Multimedia Interactive Protocols AND Systems (MIPS) 2003*, November 2003. http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/GAMS03-1.html.

[7] Görtz, Manuel, Ralf Ackermann und Ralf Steinmetz: *The Digital Call Assistant: Determine Optimal Time Slots for Calls.* In: *Second International Workshop on Multimedia Interactive Protocols AND Systems (MIPS 2004)*, November 2004. http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/GAS04-2.html.

[8] Görtz, Manuel, Ralf Ackermann und Ralf Steinmetz: *Enhanced SIP Communication Services by Context Sharing.* In: *30th EUROMICRO Conference*, September 2004. http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/GAS04-1.html.

[9] Martinovic, Ivan, Manuel Görtz, Ralf Ackermann, Andreas Mauthe und Ralf Steinmetz: *Trust AND Context: Two Complementary Concepts for Creating Spontaneous Collaborative Networks AND Intelligent Applications.* In: *Proceedings of SoftCOM'04, International Conference on Software, Telecommunications AND Computer Networks, Croatia/Italy*, Oktober 2004. http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/MGA+04-1.html.

[10] Görtz, Manuel, Ralf Ackermann, Johannes Schmitt und Ralf Steinmetz: *Context-aware Communication Services: A Framework for Building Enhanced IP Telephony Services.* In: *International Conference on Computer Communications and Networks (ICCCN) 2004*, Seiten 272–279, Oktober 2004. http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/GASS04-1.html.

Buchkapitel

[11] Görtz, Manuel: *Security Pattern*, Kapitel Use Case: IP Telephony. Wiley & Sons, 2005.

[12] Steinmetz, Ralf, Ralf Ackermann, Utz Roedig, Manuel Görtz und Markus Schumacher: *IP-Telefonie: Protokolle, Herausforderungen, Loesungen und kritische Analyse der Sicherheit*, Seiten 57–77. Januar 2002. http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/SAR+02-1.html.

[13] Steinmetz, Ralf, Manuel Görtz und Harald Müller: *Real Time Enterprise in der Praxis*, Kapitel Technische Bausteine. Springer Verlag, Berlin, Juni 2004.

[14] Steinmetz, Ralf, Manuel Görtz und Harald Müller: *The Practical Real Time Enterprise*, Kapitel Technical Building Blocks. Springer Verlag, Berlin, April 2005.

Erfindungsmeldungen

[15] Görtz, Manuel, Ralf Ackermann, Ralf Steinmetz, Harald Müller und Thomas Lederer: *Kombination von Lokationsinformationen und Kalendereinträgen, um eine Vorhersage eines zukünftigen Kontextes bzw. der Erreichbarkeit von Kommuikationspartner zu treffen.* Erfindungsmeldung, Dezember 2003.

[16] Görtz, Manuel, Ralf Ackermann, Ralf Steinmetz, Harald Müller und Thomas Lederer: *Verbessertes Kommunikationsverhalten des Anrufers durch Einbeziehung von Kontextinformationen des Angerufenen, Erfindungsmeldung.* Erfindungsmeldung, Dezember 2003.

Technische Berichte, Projektberichte

[17] Ackermann, Ralf, Manuel Görtz, Martin Karsten und Ralf Steinmetz: *Project Siemens CAMPUS: SHINE – H.450–SIP Interworking.* Technischer Bericht Project Phase 1: Comprehensive Analysis of Supplementary Services in H.450 AND SIP, Multimedia Communications (KOM), Darmstadt University of Technology, August 2001.

[18] Ackermann, Ralf, Manuel Görtz, Martin Karsten und Ralf Steinmetz: *Project Siemens CAMPUS: SHINE – H.450–SIP Interworking.* Technischer Bericht Project Phase 2: Interworking Design, Multimedia Communications (KOM), Darmstadt University of Technology, Februar 2002.

[19] Ackermann, Ralf, Manuel Görtz, Martin Karsten und Ralf Steinmetz: *Project Siemens CAMPUS: SHINE – H.450–SIP Interworking.* Technischer Bericht Project Phase 3: Gateway Implementation AND Test, Multimedia Communications (KOM), Darmstadt University of Technology, Mai 2002.

[20] Ackermann, Ralf, Manuel Görtz und Ralf Steinmetz: *IP Telefonie Feldtest (mit Siemens Komponenten) für den Ersatz der TK-Anlage der Hochschulregion Darmstadt.* Technischer Bericht Projektzusammenfassung und Einsatzempfehlung für die AG "TK Anlage" des Ausschuss 5 der TU Darmstadt, Multimedia Communications (KOM), Darmstadt University of Technology, Februar 2001.

[21] Ackermann, Ralf, Manuel Görtz, Florian Winterstein und Ralf Steinmetz: *Project Siemens CAMPUS: SHINE – H.450–SIP Interworking.* Technischer Bericht Project Phase 4: Integration of Further Supplementary Services, End-System Development AND Final Project Summary, Multimedia Communications (KOM), Darmstadt University of Technology, November 2002.

[22] Görtz, Manuel, Ralf Ackermann, Alejandro Perez und Ralf Steinmetz: *Project Siemens CAMPUS: C^3 – Contex Controlled Communication.* Technischer Bericht Project Phase 2: Design and Implementation of a Location Sensing System, Multimedia Communications (KOM), Darmstadt University of Technology, Oktober 2003.

[23] Görtz, Manuel, Ralf Ackermann, Johannes Schmitt und Ralf Steinmetz: *Project Siemens CAMPUS: C³ – Contex Controlled Communication.* Technischer Bericht Project Phase 4: Intergration of Complete System for Context-aware Communication Services and Final Project Summary, Multimedia Communications (KOM), Darmstadt University of Technology, Oktober 2003.

[24] Görtz, Manuel, Ralf Ackermann und Ralf Steinmetz: *Project Siemens CAMPUS: C³ – Contex Controlled Communication.* Technischer Bericht Project Phase 1: Comprehensive Analysis of Context-aware Computing, März 2003.

[25] Görtz, Manuel, Ralf Ackermann und Ralf Steinmetz: *Project Siemens CAMPUS: C³ – Contex Controlled Communication.* Technischer Bericht Project Phase 3: Design and Implementation of a System for Context-aware Communication Services, Multimedia Communications (KOM), Darmstadt University of Technology, April 2003.

[26] Görtz, Manuel, Alejandro Perez, Ralf Ackermann, Andreas Mauthe und Ralf Steinmetz: *Location Sensing using RADAR.* Technischer Bericht TR-KOM-2003-09, Multimedia Communications Lab, Darmstadt University of Technology, September 2003. `http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/GPA+03-1.html`.

[27] Roedig, Utz, Ralf Ackermann, Manuel Görtz und Ralf Steinmetz: *Sicherheitsarchitekturen für Netzübergänge für Multimedia-Dienste.* Technischer Bericht Project Phase: Abschlussbericht, KIMK GmbH, Januar 2004.

[28] Roedig, Utz, Manuel Görtz, Martin Karsten und Ralf Steinmetz: *RSVP as Firewall Signalling Protocol.* Technischer Bericht 6, KOM, Dezember 2000. `http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/RGKS00-1.html`.

[29] Roedig, Utz, Manuel Görtz und Ralf Steinmetz: *Multimedia Firewalls und H.323.* Technischer Bericht Project Phase: Abschlussbericht, KIMK GmbH, 2003.

[30] Schmitt, Johannes, Manuel Görtz, Ralf Ackermann und Ralf Steinmetz: *Project Siemens CAMPUS: SELECTIONS – SElf-LEarning.* Technischer Bericht Project Phase 1: Comprehensive Analysis of Self-Learning Methods, Multimedia Communications (KOM), Darmstadt University of Technology, April 2005.

[31] Tittel, Stephan, Manuel Görtz, Ralf Ackermann und Ralf Steinmetz: *Unterschung der Machbarkeit von VoIP und DSL.* Technischer Bericht Project Phase: Abschlussbericht, Multimedia Communications (KOM), Darmstadt University of Technology, Januar 2005.

Anhänge

Anhang A

Messergebnisse der WLAN-basierten Lokationsverfahren

Ein Programm zur Messung der Signalstärken und Erstellung der Radio Map wurde entwickelt und eingesetzt [GPA+03]. Um die Signalstärken zu allen sichtbaren Basisstationen messen zu können, wurden die erweiterten Linux-Kernel-Treiber [www36] für die WLAN-Karte verwendet. Für den Zugriff auf die Informationen der Karte wurde die API der *Wireless Extensions* [www35] benutzt. Die Radio Map wurde in einer MySQL-Datenbank abgespeichert. Über einen Web Service, der auf dem ContextServer (siehe Abschnitt 9.4) läuft, können Anwendungen die Ortsinformationen beziehen.

A.1 Programm zur Messung der Signalstärken

Die Umsetzung des Innenraumlokationsverfahrens, welches WLAN-Signalstärken aufbaut, besteht aus den Komponenten: Messclient, Datenbank und Web Services. Als Zielplattform wurde ein PDA mit WLAN Netzwerkkarte und einem Linux Betriebssystem gewählt. Der gewählte PDA ist leistungsfähig genug, um auch als IP-Telefonie Endgerät fungieren zu können [AGKS01].

Ein Programm wurde entwickelt, welches die Erstellung der Radio Map, als auch die aktive Messung des aktuellen Fingerprints unterstützt. Darüberhinaus zeigt das Programm an, in welchem Raum sich der Nutzer gegenwärtig befindet. Die Details zum Messclient sind nachfolgend aufgeführt.

A.1.1 Implementierungsdetails

Als Plattform für den Client wurde ein mobiles Endgerät, in diesem Fall ein PDA gewählt. Der Compaq iPAQ H3870 wurde mit dem Betriebssystem Linux Familiar 0.7.2 und dem Kernel 2.4.19-rmk6-pxa1-hh30 betrieben. Der iPAQ besitzt einen StrongARM-Prozessor für den ein Cross-Compiler existiert, so dass das Programm auf einen Linux Desktop PC mit x86-Architektur entwickelt werden konnte und anschließend auf die PDA-Plattform portiert werden konnten. Aufgrund von Problemen mit der Treiber-API jedoch konnte nicht die vollständige Leistungsfähigkeit der Hardwaretreiber für die Netzwerkkarte abgerufen werden, so dass sich das Programm als nicht lauffähig auf dem PDA erwiesen hat.

Als WLAN-Karte wurde eine Lucent, jetzt ORiNOCO, GoldCard verwendet. Um ein Signalstärken-*Scanning* durchführen zu können, mussten „gepatchte“ Treiber eingesetzt werden. Der Treiber `mwlan_cs` von [www36] wurde verwendet und angepasst. Zusammen mit den `iw-tools` [www35] konnten die notwendigen Daten gewonnen werden.

A.1.2 Programm zur Unterstützung des Nutzers

Das vorher beschriebene Programm erfüllt die technisch notwendigen Funktionen. Die Signalstärken können bestimmt werden und in eine Datei oder eine Datenbank gespeichert werden. Jedoch ist dies nicht nutzerfreundlich. Daher wurde ein Programm entwickelt, das mittels einer graphischen Oberfläche (GUI) die technischen Details vor dem Nutzer verbirgt. Der Nutzer kann auf alle Funktionalitäten über Menüs zugreifen. Darüberhinaus bietet das Programm die graphische Darstellung des Raumes, in welchem sich der Nutzer gegenwärtig befindet.

Das Programm wurde in PyQT [www19] einem Phyton-Binding [www22] für die QT Module von Trolltech [www30] realisiert. Dadurch kann die entwickelte GUI auf unterschiedlichen Plattformen ausgeführt werden. Die Hauptfunktionen des Programms sind:

- Durchführen der Messung der empfangenen Signalstärken von den Access Points
- Berechnung der Fingerprints aus den Signalstärken
- Speichern der Daten in eine Datenbank oder alternativ über einen Web Service
- Abruf von gespeicherten Informationen aus der Datenbank oder von einem Web Service

Das Programmhauptfenster besteht aus vier Reitern, hinter denen sich jeweils eine der vier Hauptfunktionen verbirgt. Es sind dies SSmeasure, Format File, FP Calculation und Retrieve FP. Sie werden nachfolgend vorgestellt.

A.1.2.1 Menü SSmeasure

Das Menü *SSmeasure* ist in Abbildung A.1 dargestellt. In dem Menü lassen sich Parameter für die Signalstärkenmessung festlegen. Die Funktion dient der Erstellung einer Radio Map. Der Knopf *Floorplan* öffnet dazu eine Karte des Stockwerks, wie dies in Abbildung A.5 zusehen ist, und der Nutzer kann den Raum auswählen, in dem er sich befindet.

Den Raum kann er durch die Eingabe einer Positionsnummer genauer spezifizieren. Die Zuordnung der Nummern zu den Positionen ist in Abbildung A.2 dargestellt. Weiterhin kann der Nutzer die Anzahl der Messungen sowie die Zeit zwischen den einzelnen Messungen festlegen. Nach Drücken des *OK*-Knopfes wird die Messung gestartet, wodurch das in Anhang A.1.2 vorgestellte Programm gestartet wird. Die Daten werden in einer Datei, deren Name sich aus dem Raum und der Position bestimmt, gespeichert.

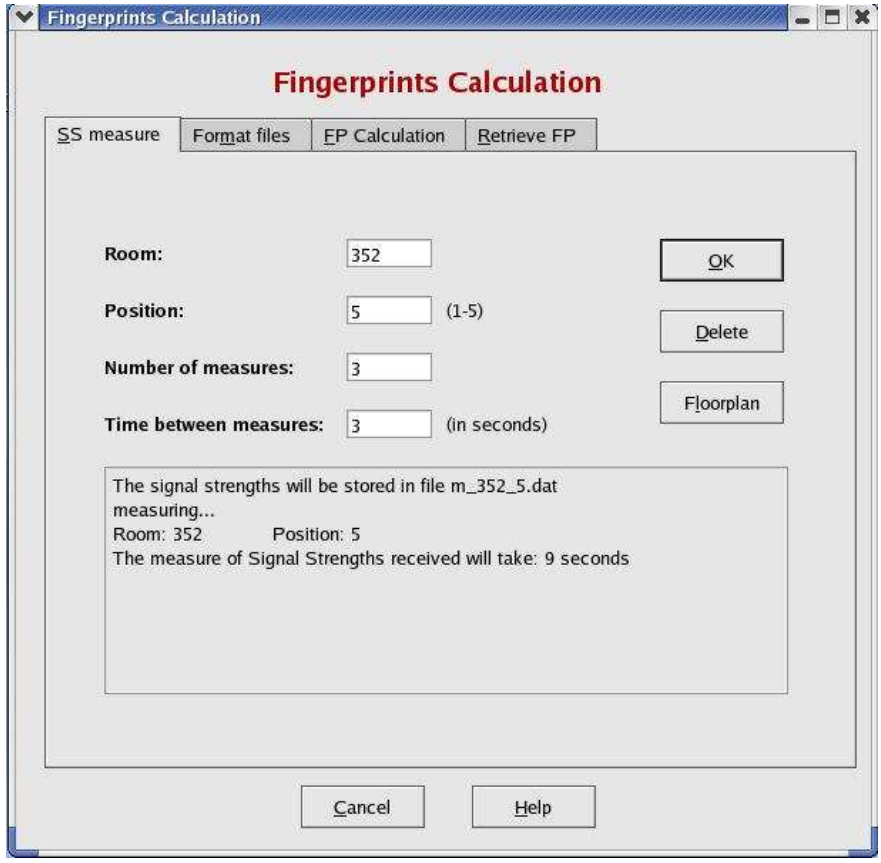


Abbildung A.1: Programmreiter zur Parametrisierung der Messung

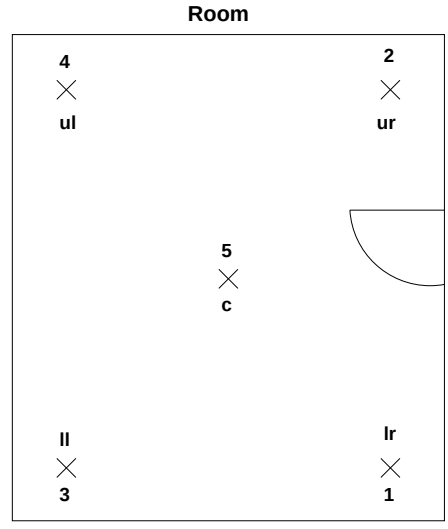


Abbildung A.2: Zuordnung der Positionen in einem Raum zu Nummern

A.1.2.2 Menü Format files

In diesem Menüpunkt können Daten mit Daten geladen werden und durch ein awk-Skript, welches im Hintergrund aufgerufen werden kann, formatiert werden. Dies erleichert die Lesbarkeit der Dateien.

A.1.2.3 Menü FP Calculation

Im Reiter *FP Calculation* können die Daten aus den formatierten Dateien eingeladen werden. Anschließend werden die Fingerprints erstellt und graphisch dargestellt. Dies ist in Abbildung A.3 gezeigt. Das Menü zeigt auch an, wieviele Messungen durchgeführt worden sind. Abschließend können die Daten in eine Datenbank übertragen (*Insert into DB*) oder gelöscht werden.

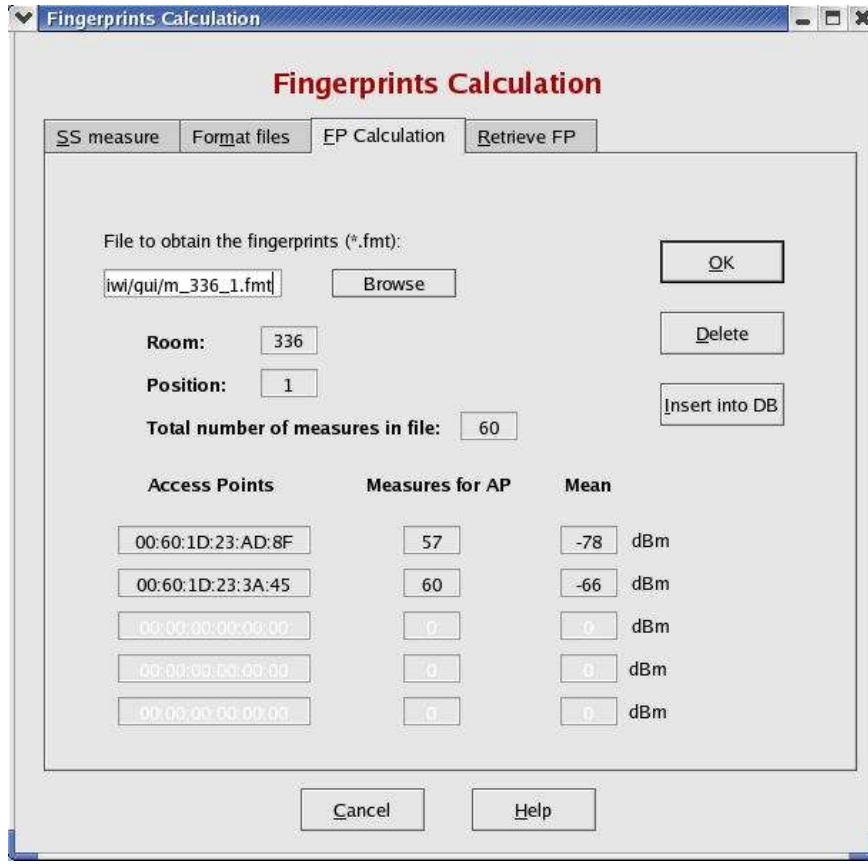


Abbildung A.3: Programmreiter zur Berechnung der Fingerprints

A.1.2.4 Menü Retrieve FP

Im *Retrieve FP* Menü können die Daten zu einem bestimmten Raum und Position aus der Datenbank bezogen werden. Der Nutzer kann den Raum und die Position direkt in den Textfeldern spezifizieren oder wieder aus der Karte (*Floormap*) auswählen. Das Menü ist in Abbildung A.4 abgebildet.

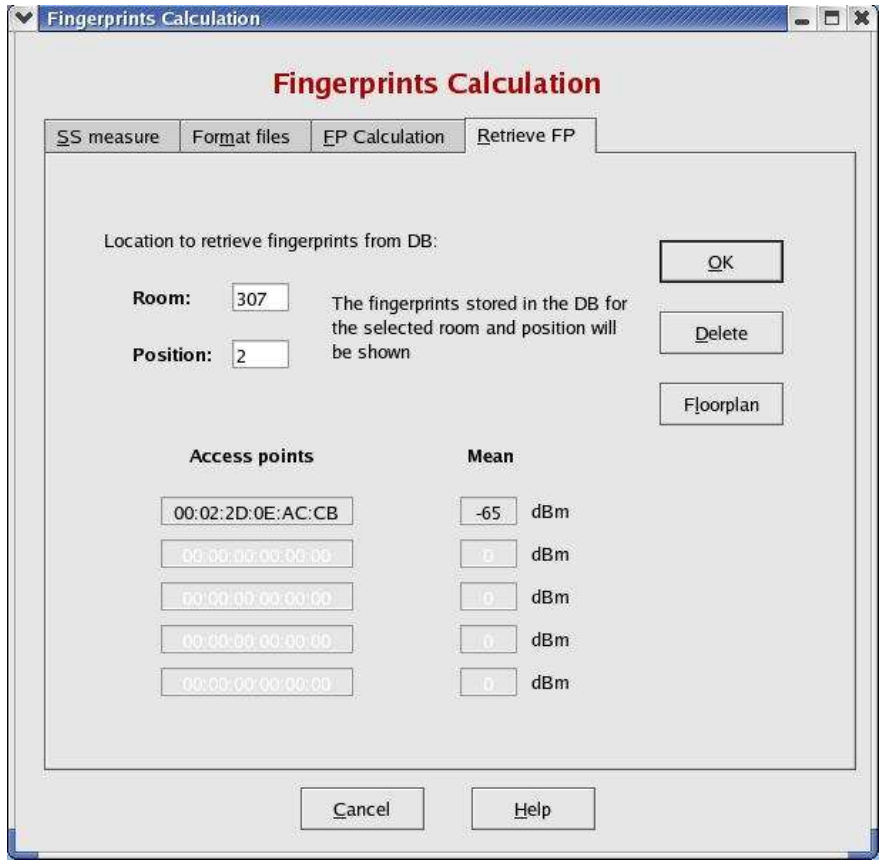


Abbildung A.4: Programmreiter zum Abruf der Fingerprints

A.1.3 Floorplan-Komponente

Eine in verschiedenen Menüpunkten des Programms genutzte Funktion ist die *Floorplan*-Komponente. Diese zeigt eine graphische Abbildung des Grundrissplans der Stockwerke mit den vorhandenen Räumen. Der Nutzer kann mit der Maus über die Räume fahren und die Informationen zu den Räumen werden ihm angezeigt, wie dies in Abbildung A.5 dargestellt ist.

Umgekehrt kann der Nutzer eine Raumnummer eingeben und der dazugehörige Raum wird hervorgehoben. Wird ein Raum ausgewählt so werden die Daten zwischengespei-

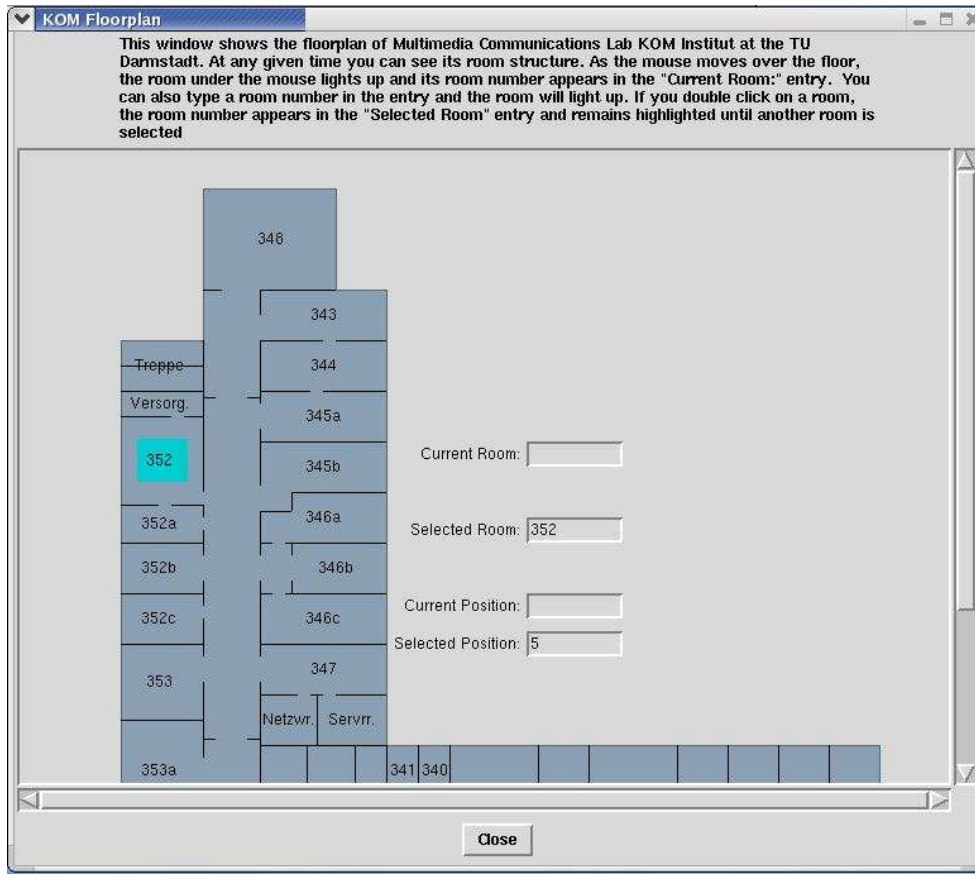


Abbildung A.5: Grundrissplan mit Informationen zu den Räumen

chert und so anderen Programmen zu Verfügung gestellt. Das Programm wurde mit der Sprache tcl/tk [www29] geschrieben und ist daher portabel.

A.1.4 Datenbank

Die Daten, die während der Messungen gesammelt wurden, werden in einer Datenbank gespeichert. Als Datenbank wurde eine MySQL-Datenbank [www14] eingesetzt. Die Anwendungen können direkt in die Datenbank oder über Web Service gespeichert werden. Der Aufbau der Datenbank ist als Entity-Relationship-Model (ER)-Modell in Abbildung A.6 abgebildet.

Die drei Grundtabellen nehmen die Daten für die Access Points, die Clients sowie die Lokationen auf. Jede Tabelle kann dynamisch erweitert werden, um neue Access Points aufnehmen zu können oder neue Charakteristiken speichern zu können. Über SQL-Abfragen können die gewünschten Informationen von den Anwendungen bezogen werden.

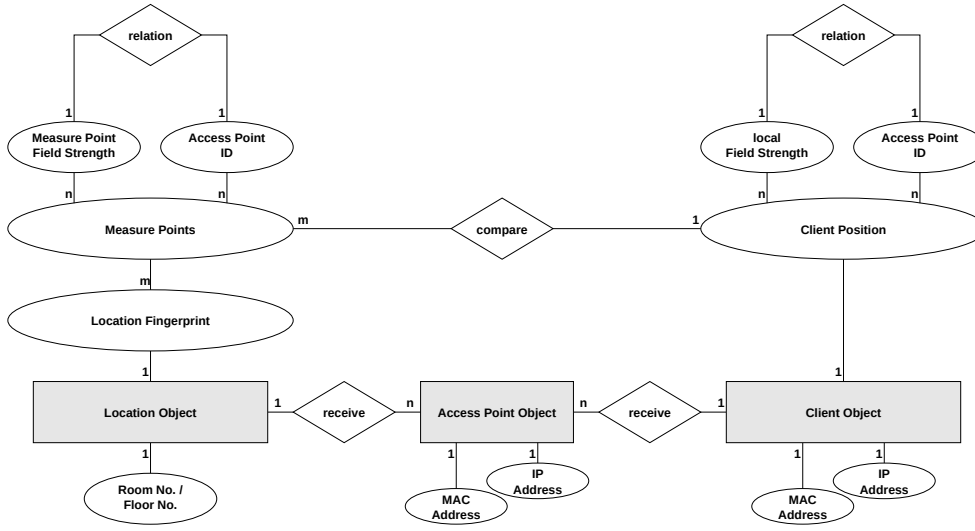


Abbildung A.6: Entity-Relationship Modell der Datenbank mit den Lokationsinformationen

A.2 Einsatzszenario

Die praktische Erprobung des entwickelten Innenraumlokationverfahrens wurde auf dem Stockwerk des Lehrstuhls „Multimedia Kommunikation (KOM)“ durchgeführt. Die Verteilung der Basisstationen[1] wurde nicht speziell für die Lokationsbestimmung verändert[2] und ist in Abbildung A.7 dargestellt. Ebenfalls in der Abbildung ist der Weg, der abgeschritten worden ist, um die Charakteristik der Signalstärken zur Entfernung zu bestimmen. Das Ergebnis ist in Abbildung 9.4 dargestellt.

Die Beschaffenheit des Einsatzgebiets, d. h. die Lage der Räume oder das Material der Decken und Wände beeinflusst sehr stark die Signalcharakteristiken. Die in [BP00] angegebenen Werte zur Anzahl der Access Points sowie die Ergebnisse konnten nicht direkt übernommen bzw. bestätigt werden. Daher müssen für den Einsatz in anderen Umgebungen ebenfalls erst Tests durchgeführt werden. Die Größenordnung der Erkennungsrate deckt sich jedoch mit den experimentellen Ergebnissen aus [BP00].

A.3 Statistische Auswertung der Messungen

A.3.1 Langzeitmessung

Um die Stabilität der gemessenen Signalstärken bestimmen zu können, wurde eine Langzeitmessung über 27 Stunden durchgeführt. Die Bewertung der Messung ist in

[1] Avaya AP 1000

[2] Jedoch fiel ein Access Point aus und zwei Access Points wurden in andere Räume verschoben. Dadurch musste eine Vielzahl ein Messungen erneut durchgeführt werden.

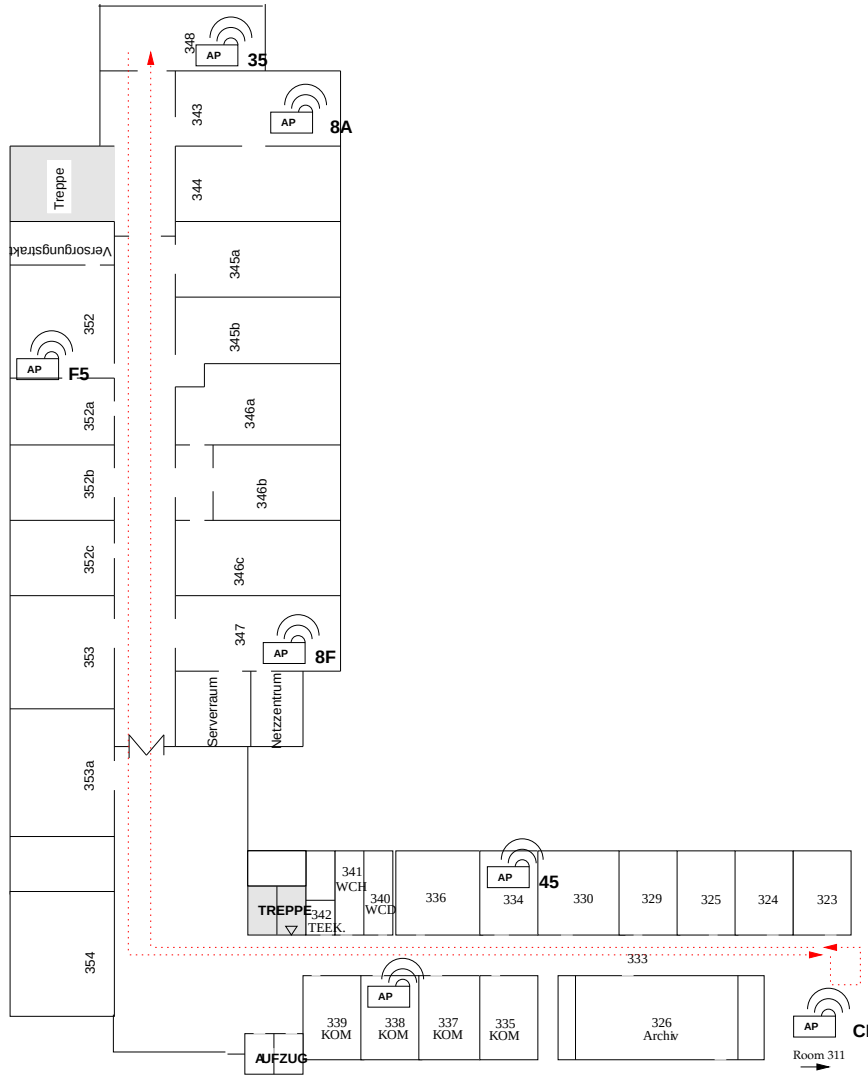


Abbildung A.7: Raumplan des Lehrstuhls KOM

Tabelle A.1: Signalstärken der 27 Stunden Messung

Indizes	**24h**			**Arbeitszeit**			**Nachtzeit**		
AP	$8A$	$8F$	35	$8A$	$8F$	35	$8A$	$8F$	35
Mean	-70	-73	-69	-73	-72	-70	-68	-75	-68
Median	-69	-74	-68	-73	-71	-69	-68	-75	-68
Mode	-68	-75	-68	-73	-71	-69	-68	-75	-68
Std Abw (s)	2,88	2,58	1,86	3,25	2,93	2,28	1,26	1,68	0,95

Tabelle A.2: Fingerprints in unterschiedlichen Positionen in verschiedenen Räumen

Raum	344		345a	
AP	8A	8F	8A	8F
Pos 1 (lr)	-43	-83	-67	-74
Pos 2 (ur)	-54	-79	-67	-74
Pos 3 (ll)	-62	-77	-66	-75
Pos 4 (ul)	-64	-78	-74	-68
Pos 5 (c)	-65	-82	-68	-79

Unterabschnitt 9.2.2.1 beschrieben und die Messkurve in Abbildung 9.5 aufgetragen. Die dazugehörigen statistischen Merkmale sind in Tabelle A.1 angegeben.

A.3.2 Raum/Lage-Abhängigkeit der Messgeräts

Um charakteristische Fingerprints Φ für jeden Raum gewinnen zu können, wurden zwei weitere Tests durchgeführt. Zum einen wurden Unterschiede an verschiedenen Positionen in einem Raum untersucht, zum anderen die Orientierung des Messclients.

A.3.2.1 Position im Raum

Um die Unterschiede in den zu messenden Signalstärken in einem einzelnen Raum aufzuzeigen, wurden in zwei Räume (344 und 345a) Messungen durchgeführt. Dazu wurden fünf Positionen ausgewählt, wie diese in Abbildung A.2 gezeigt sind. Die Messung für Raum 344 ist in Abbildung A.8 gezeigt. Die statistischen Auswertungen der Messungen für beide Räume sind in Tabelle A.2 aufgeführt.

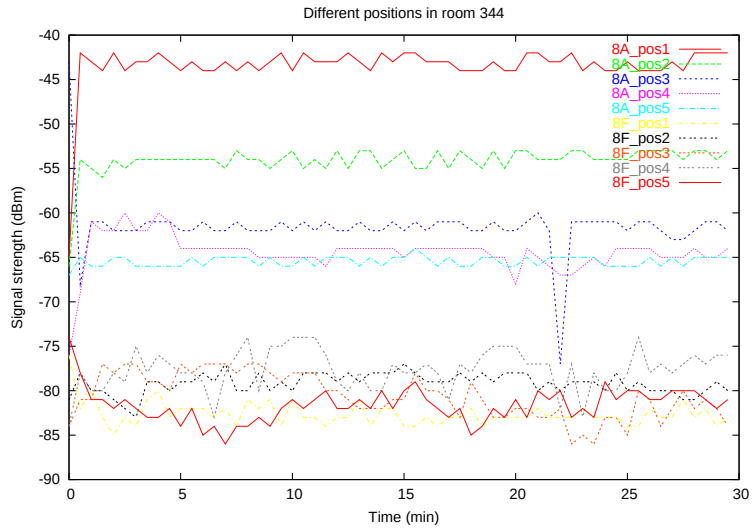


Abbildung A.8: Signalstärke an verschiedenen Positionen im Raum

Tabelle A.3: t-Test über den Einfluss der Orientierung des mobilen Endgeräts

Indices	0°	90°
Mittelwert	-74,275	-78,2
Varianz	4,33	7,39
Messungen	120	120
Hypothesized mean difference	0	
Observed mean difference	3,925	
ν	228,89	
t_{Stat}	12,56	
$P(t \leq t)$ two-tail	0	
t_{Critical} two-tail	1,97	

A.3.2.2 Orientierung des Messgeräts

Es wurde ein t-Test, wie er in Anhang A.3.2.3 beschrieben ist, durchgeführt, um die *Signifikanz* der Unterschiede in den Messungen in Bezug auf der Orientierung des Messgerätes zu testen. In Abbildung A.9 zeigt sich eine Verschiebung des Mittelwerts in einer Orientierung (90°) um 3 dB. Die Ergebnisse des t-Tests mit einem Konfidenzintervall von 95% sind in Tabelle A.3 aufgeführt. Die Hypothese, dass es keinen signifikanten Unterschied zwischen den beiden Orientierungen gibt, wird angenommen, wenn der Wert von t_{Stat} im Intervall von $\pm t_{\text{Critical}}$ liegt.

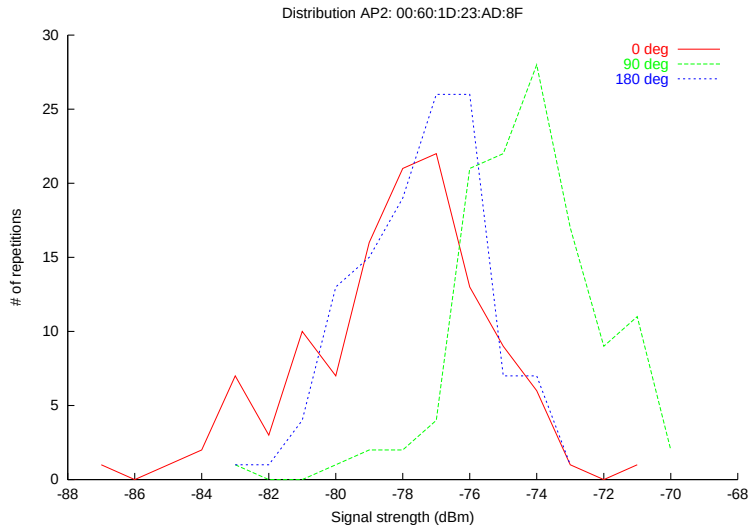


Abbildung A.9: Einfluss der Signalstärke in verschiedenen Orientierungen des Endgeräts

A.3.2.3 t-Test

Der t-Test erlaubt es, zwei unterschiedliche Beobachtungen zu vergleichen und zu entscheiden, ob die Differenzen der Mittelwerte sich signifikant unterscheiden. Hierzu wird ein Konfidenzintervall um die Mittelwerte gebildet. Die Hypothese, welche gegen die Signifikanz spricht, ist, dass der Unterschied Null sein muss. Folgende Schritte werden bei einem t-Test ausgeführt.

1. Berechnung des Mittelwerts der Messung

$$\overline{x}_a = \frac{1}{n_a} \sum_{i=1}^{n_a} x_{i_a} \qquad \overline{x}_b = \frac{1}{n_b} \sum_{i=1}^{n_b} x_{i_b}$$

2. Berechnung der Standardabweichung der Messung

$$s_a = \left\{ \frac{\left(\sum_{i=1}^{n_a} x_{i_a}^2 \right) - n_a \overline{x}_a^2}{n_a - 1} \right\}^{\frac{1}{2}} \qquad s_b = \left\{ \frac{\left(\sum_{i=1}^{n_b} x_{i_b}^2 \right) - n_b \overline{x}_b^2}{n_b - 1} \right\}^{\frac{1}{2}}$$

3. Berechnung der Differenz der Mittelwerte

$$\overline{x}_a - \overline{x}_b$$

4. Berechnung der Standardabweichung der Differenz der Mittelwerte

$$s = \sqrt{\frac{s_a^2}{n_a} + \frac{s_b^2}{n_b}}$$

5. Berechnung der Anzahl der Freiheitsgrade

$$\nu = \frac{\left(\frac{s_a^2}{n_a} + \frac{s_b^2}{n_b}\right)}{\frac{1}{n_a+1}\left(\frac{s_a^2}{n_a}\right)^2 + \frac{1}{n_b+1}\left(\frac{s_b^2}{n_b}\right)^2} - 2$$

6. Berechnung des Kofidenzintervalls für die Differenz der Mittelwerte

$$(\overline{x}_b - \overline{x}_a) \pm t_{[1-\frac{\alpha}{2};\nu]} s$$

   wobei $t_{[1-\frac{\alpha}{2};\nu]}$ die $(1-\frac{\alpha}{2})$-Quantile des t-Merkmals mit dem Freiheitgrad ν ist.

7. Wenn das Konfidenzintervall Nullen enthält, so ist der Unterschied nicht signifikant im Konfidenz-Level von $100(1-\alpha)\%$

A.3.3 Benachbarte Räume

Die Unterschiede der Fingerprints in zwei aneinander grenzenden Räumen ist in der Regel ausreichend, um eine Entscheidung, in welchem Raum sich das mobile Endgerät befindet, treffen zu können. Abbildung A.10 zeigt die gemessenen Signalstärken in zwei benachbarten Räumen. In Abbildung A.11 sind die Signalstärken, die in zwei benachbarten Räumen (345 und 348) gemessen wurden, zu den einzelnen Basisstationen gezeigt. Durch die Kombination der Signalstärken zu mehreren Basisstationen ergeben sich genügend aussagekräftige Messungen, um Räume auch bei Überschneidung einzelner Signalstärken unterscheiden zu können.

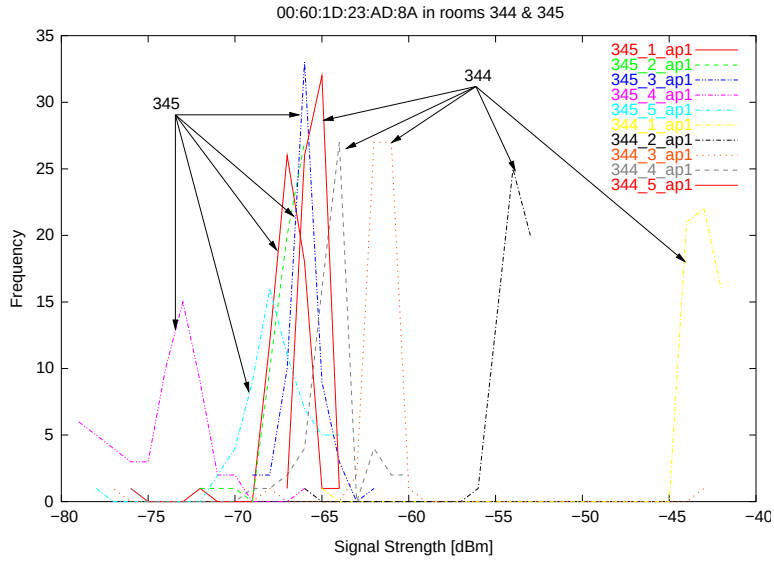


Abbildung A.10: Signalstärken in zwei direkt benachbarten Räumen

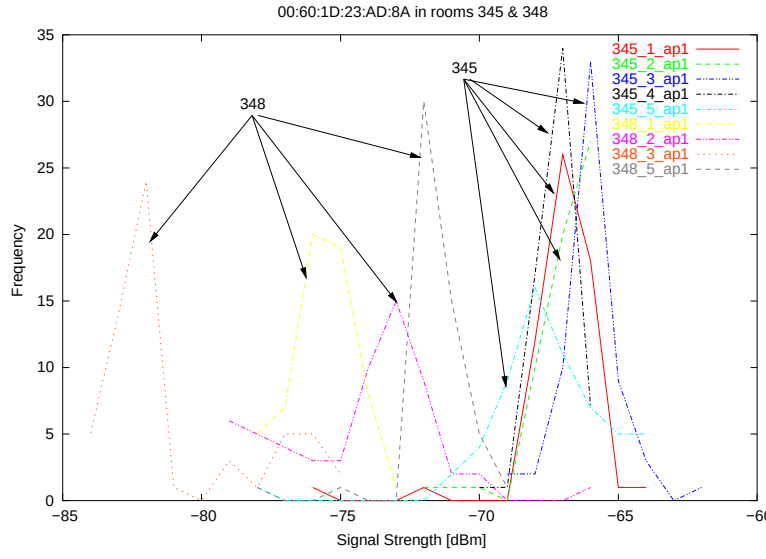


Abbildung A.11: Signalstärken in zwei nahe beieinander liegenden Räumen

Anhang B

Fuzzy Logik

Das in Abschnitt 6.5 vorgeschlagene Verfahren Fuzzy Logik Regeln zur Aggregation von Kontextmerkmalen zu Kontexten zu verwenden wurde umgesetzt. Die einzelnen aufgestellten Variablen werden nachfolgend vorgestellt. Ebenso wurde die Performanz der Fuzzy-Engine gemessen.

B.1 Fuzzy Logik Variablen

Die folgenden Sensordaten sind als Fuzzy Logik Variablen modelliert worden, um sie zu Kontexten zu aggregieren. Sie sind nachfolgend aufgelistet:

- Wochentag (days of the week)
- Zeit (time)
- Helligkeit (brightness)
- Maus- und Tippaktivität (mouse and typing activity)
- Umgebung (presence of other people)
- Bewegungsmuster (movement of the user)
- Lokation (detector of position)

B.1.1 Fuzzy Eingangsvariablen und Mitgliedfunktionen

Eine Reihe von Eingangsvariablen wurden für die Machbarkeitsstudie von Fuzzy Logik zur Kontextaggregation definiert. Diese werden nun erläutert. Die meisten Fuzzy Logik Variablen sind logarithmisch dargestellt. Es handelt sich meist um trapezoide Funktionen, die einen großen Wertebereich umfassen.

B.1.1.1 Wochentag

Die Mitgliedfunktion (*Day*) ist gegeben durch:

```
FuzzyVariable daysOfTheWeek = new FuzzyVariable ("Days",0,6,"Days");
```

Die (Pseudo-) Einheit ist 'Days' und der Wertebereich ist von $0 \ldots 6$. So entspricht 0 dem Montag und 1 dem Dienstag und so weiter. Mit ihnen wurden die Mitgliedsfunktionen (*Weekend*) und (*Weekdays*) definiert, wie es in Abbildung B.1 gezeigt ist.

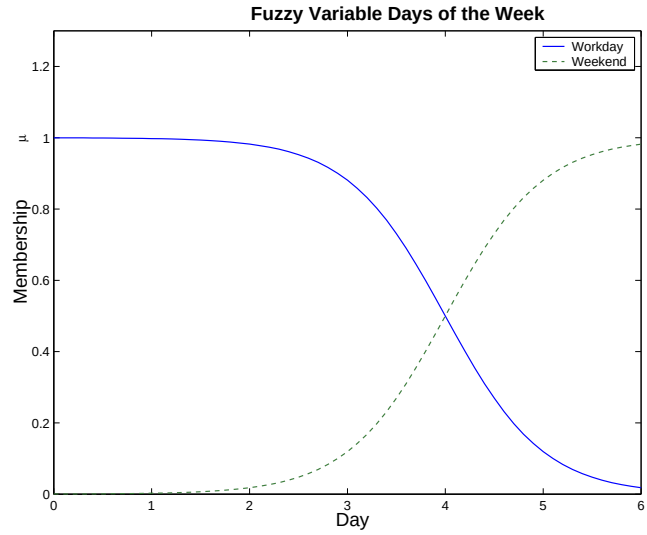


Abbildung B.1: Mitgliedsfunktion für Wochentage

B.1.1.2 Zeit

Der 24-Stunden Tag wurde in vier Bereiche unterteilt. Die Aufteilung ist in Abbildung B.2 abgebildet. Die Variable wurde für das Programm wie folgt definiert: Der Wertebereich ist von $0 \ldots 23$ 'Hours'

```
FuzzyVariable time = new FuzzyVariable ("Time",0,23,"Hour");
```

B.1.1.3 Helligkeit

Die Helligkeit in Räumen wird in dem umgesetzten Sensor in 'lux' gemessen. Die Einheit korrespondiert mit $\mathrm{lm/m^2}$ Für die Festlegung von typischen Werten, wurden aus [www20] und [www8] abgeleitet. Die Zuordnung von Helligkeitswerten und Aktivitäten ist in Tabelle B.1 aufgeführt.

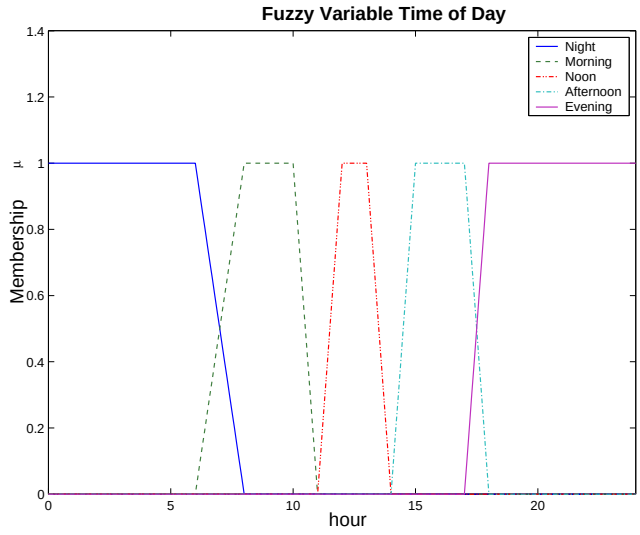


Abbildung B.2: Mitgliedsfunktion für Zeit

Tabelle B.1: Typische Helligkeitswerte und die dazugehörigen Situationen

Situation	**typischer Wert**
Sunlight	102 lux
TV stage	25 lux
Sky light alone	16 lux
Dull day	1 lux
Reading	200-500 lux (recommended)
Room light	50-200 lux (recommended)
Moonlight	0.4 lux

Die Mitgliedsfunktion hat einen Wertebereich von 0 . . . 600 'lux' und ist definiert durch: `FuzzyVariable light = new FuzzyVariable ("Light",0,600,"Lux");` und in Abbildung B.3 gezeigt.

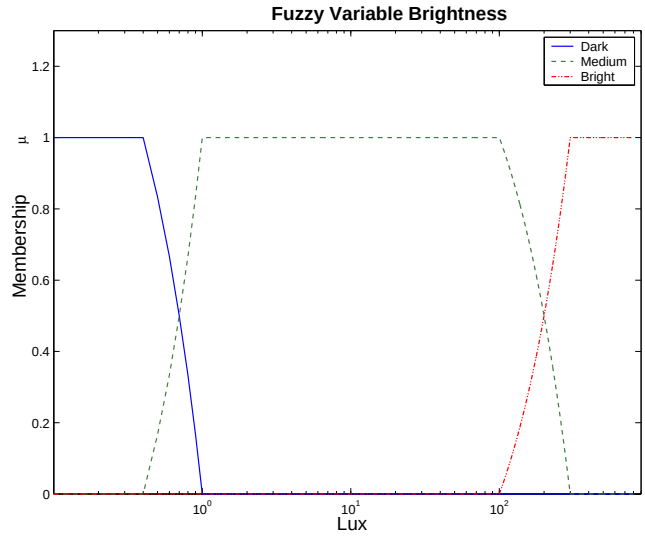


Abbildung B.3: Mitgliedsfunktion für die Helligkeit

B.1.1.4 Mausaktivität

Eine Referenz für die Wertebereiche für Mausaktivität ist in [www3] gegeben. Bei einer konzentrierten Arbeit am PC wird die Aktivität der Maus und der Tastatur höher sein, als wenn der Nutzer nicht am PC arbeitet. Die Mitgliedsfunktion ist in Abbildung B.4 gezeigt.

Als Einheit wurde 'Events/minute' gewählt und der Wertebereich umfasst 0 . . . 600 und wird wie folgt definiert:

```
FuzzyVariable typingMouse =
new FuzzyVariable ("Typing and mouse activity", 0, 600, 'Events/minute");
```

B.1.1.5 Gruppengröße

Die Mitgliedsfunktion (*GroupSize*) für die Zahl der einen Nutzer umgebenden Personen wird wie folgt definiert

```
FuzzyVariable groupSize = new FuzzyVariable ("Group Size",0,30,"People");
```

Die dazugehörige Funktion ist in Abbildung B.5 gezeigt.

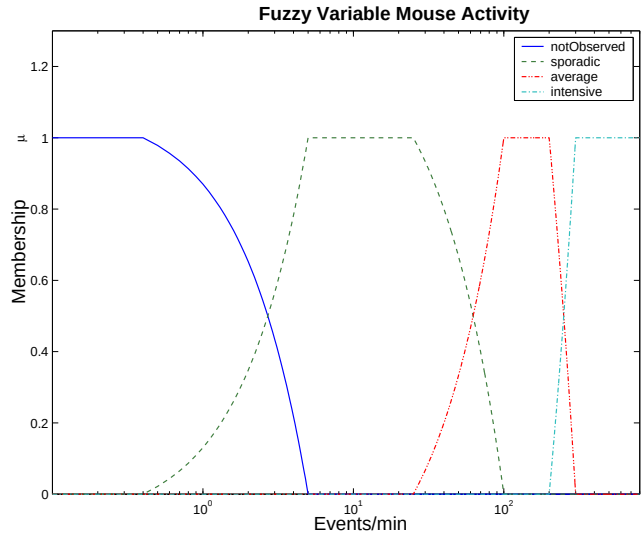


Abbildung B.4: Mitgliedsfunktion für Mausaktivität

Das Bestimmen der Anzahl von Personen in der unmittelbaren Nachbarschaft eines Nutzers kann nützliche Rückschlüsse liefern, die beispielsweise mit der reinen Lokationsangabe nicht möglich wären. So wird bei alleiniger Nutzung des Ortes 'Besprechungsraum' keine *Besprechung* erkannt, bei einer Gruppe allerdings sind die Indizien stärker gegeben.

B.1.1.6 Bewegungsmuster

Die Einbeziehung von Bewegungsmustern des Nutzers gibt einen guten Hinweis auf einen mögliche aktuelle Aktivität. Zusätzlich können die Informationen für die Vorhersage von Kontexten genutzt werden. In der Kombination mit anderen Sensorinformationen kann so aus *static*, *dark*, *night* geschlossen werden, dass der Nutzer ruht.

Die linguistischen Terme sind in Abbildung B.6 abgebildet. Die Mitgliedsfunktion (*Movement*) ist wie folgt definiert

```
FuzzyVariable movement = new FuzzyVariable ("Movement",0,120,"cm/s");
```

B.1.1.7 Lokation

Eine wichtige Informationsquelle für die Bestimmung von Kontexten ist die Lokation. Zum einen sind, wie in Abschnitt 4.2 gezeigt, Kontexte umgebungsabhängig und zum anderen gibt der Ort, in dem sich der Nutzer oder das Gerät befindet bereits einen guten Hinweis auf den Kontext.

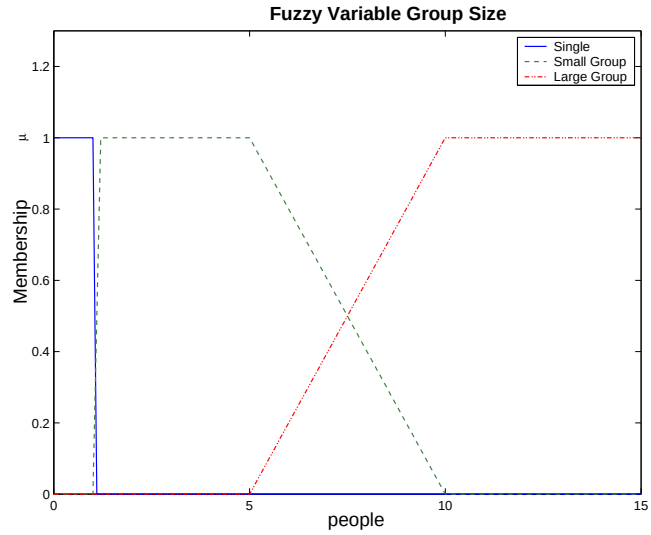


Abbildung B.5: Mitgliedsfunktion für die Gruppengröße um einen Nutzer

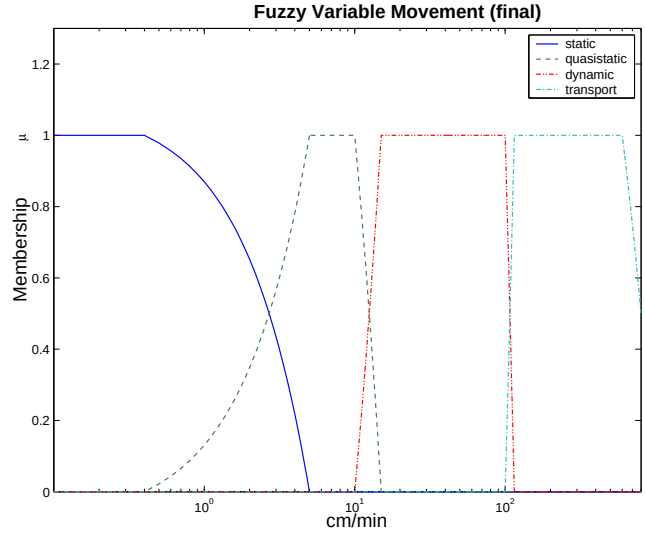


Abbildung B.6: Mitgliedsfunktion für Bewegungsmuster

In Abschnitt 9.2 wurde ein auf der Signalstärke beruhendes Lokationsverfahren vorgestellt. Die Erkennung unterliegt immer einer gewissen Unsicherheit, die sich durch Fuzzy-Variablen modellieren lässt. Für die in Abbildung B.7 gezeigte Mitgliedfunktion wurden jedoch *Singletons* gewählt, die die (*Position*) direkt angeben. Dies ist definiert als:

```
FuzzyVariable position = new FuzzyVariable ("Position", 0,3);
```

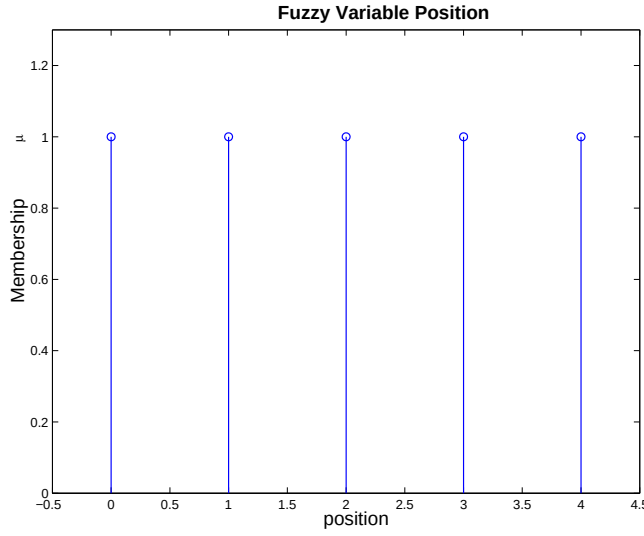


Abbildung B.7: Mitgliedsfunktion für Position

B.1.2 Fuzzy Ausgabevariablen

Die Eingangswerte werden durch die Fuzzy Regeln auf 7 mögliche Kontexte abgebildet. Es wurden die Regeln für die folgenden Bezeichnungen definiert:

- Sleeping
- AtHome
- Driving
- Meeting
- Available
- Busy
- Office

B.2 Fuzzy Regeln

Tabelle B.2: Fuzzy Regeln für die Simulation

Regel	Fuzzy Regeln Bedingung	Schlussfolgerung	Bemerkung
1	Weekend Dark Static	Sleeping	Der Kontext *Sleeping* soll erkannt werden, wenn der Nutzer am Wochenende liegt und es dunkel ist.
2	Night *not* Bright TypingNotObserved *not* BigGroup	Sleeping	Der Nutzer ist alleine oder nur von wenigen Menschen umgeben und es ist Nacht (0:00 bis ca. 6:00).
3	Evening Dark Static	Sleeping	Diesse Regel ist unabhängig vom Wochentag und bezieht sich auf die Variablen Tageszeit, Helligkeit und Bewegung.
4	*very* Weekend NotAtWork	AtHome	Der Kontext AtHome soll erkannt werden, wenn es starke Indizien für ein Wochenende gibt und der Nutzer nicht im Büro ist.
5	Weekday Evening Medium *or* Bright TypingNotObserved *not* BigGroup *not* AtWork	AtHome	An einem normaler Werktag am Abend soll der Kontext AtHome erkannt werden,
6	NotObserved *not* BigGroup *very* Dynamic, *not* AtWork	Driving	Der Nutzer ist nicht in einer großen Gruppe und bewegt sich schnell ohne Aktivitäten an der Tastatur.
7	Weekday BigGroup AtWork	Meeting	Der Nutzer befindet sich in einer größeren Gruppe an einem Werktag. Daraus lässt sich auf ein Meeting schließen.
8	Weekday InMeetingRoom	Meeting	Die Lokation *InMeetingRoom* gibt bereits einen deutlichen Hinweis auf den Kontext *Meeting*.
9	Weekday Bright IntensiveTyping	Working	Es ist ein Wochentag und hell und der Nutzer tippt auf der Tastatur. Der Kontext *Working* soll erkannt werden.

wird auf der nächsten Seite fortgesetzt

Tabelle B.2 – fortgesetzt von vorheriger Seite

Regel	Bedingung	Fuzzy Regeln Schlussfolgerung	Bemerkung
10	Weekday Medium *or* Bright SporadicTyping *or* Average Alone *or* (very SmallGroup) Static *or* Quasi-static AtWork	Available	Bei diesen Fuzzy Variablen soll der Nutzer als verfügbar (*Available*) für Kommunikation ausgezeichnet werden.
11	Workable Morning *or* Afternoon Medium Quasistatic *or* Dynamic NotAtWork	Available	Auch bei dieser Variablenkombination soll der Kontext *Available* erkannt werden.

B.3 Simulation der Engine

Tabelle B.3: Input Simulation

Simulation	Fuzzy Variablen						
	days	time	light	typing	people	move	position
1	6.5	1	0.1	3	0	2	2
				0.5			
		17					
		18					
	4	1					
		19					
2	4	19	10	0	3	0.05	2
	3	17					
		17.5					
3	3	12	100	100	3	0.05	2
							0
	6						0
	3	12			10		
					6		
					3		

wird auf der nächsten Seite fortgesetzt

Tabelle B.3 – fortgesetzt von vorheriger Seite

Simulation	Fuzzy Variablen						
	days	time	light	typing	people	move	position
4	3	12	100	300 250	3	0.05	0
5	3	12	100	0 100	3 10 10 3 	3 20 100 200 115	2
6	 6						1 1
7	5	20	0.6	0	1	12	2
8	3	18	300	100	0	3	2

Tabelle B.4: Input Simulation

Simulation	Production Rules										
	1	2	3	4	5	6	7	8	9	10	11
1	1 0.85 0.85 0.85 0.01 0.01	 1 1 	1 0.05 0.85 0.85	1 1 1 1 0.00039 0.00039	 0.23						
2			0.00039	1	0.047 0.52						0.019 0.019
3	*noDef*					1	0.6		0.99 0.16 0.99	0.019	
4							1	0.5002	0.5		
5						0.0033 1			0.6	0.6 1 1 1 0.0033	
6	*no Def*						1				
7			1	0.01							
8						1					0.00025

Tabelle B.5: Statistische Auswertung der Performanzbetrachtung der Fuzzy Logik Engine

Indizes	Start Engine	Evaluation
Mittelwert	214,13	7,59
Median	221,50	6,45
Std Abw (s)	20,29	5,52

B.4 Performanzbetrachtung der Fuzzy Logik Engine

Um die Leistungsfähigkeit der verwendeten Fuzzy Logik Engine sowie der aufgestellten Regeln zu testen wurde mittels eines einfachen Messaufbaus die Latenzzeiten bestimmt. Dazu wurden Zeitstempel (*time stamps*) genommen, um daraus die Zeitunterschiede bestimmen zu können. Die Messungen wurden auf einem PC mit x86-Architektur (P4 2 GHz, 512 MB Hauptspeicher) unter dem Linux Betriebssystem durchgeführt.

Es wurden 30 Versuchsreihen erstellt. In jeder der Versuchreihen wurde das Programm einmal aufgerufen. Dieser Wert wurde als Start- und Initialisierungszeit t_{start} bezeichnet und separat behandelt. Im Programm wird 1000 mal die Zeit für die Auswertung der Regeln t_{eval} bestimmt. Dazu wurde jeweils der Mittelwert von 10 Auswertungen genommen. Die Regeln wurden bei jedem Aufruf mit Zufallszahlen als Eingangswerte aufgerufen. Insgesamt wurden also 30000 Messungen durchgeführt. Die statistische Auswertung ist in Tabelle B.5 gegeben.

Die Messung der Startzeiten der Engine ist in Abbildung B.9 gezeigt. Der Graph zeigt leichte Schwankungen um den Mittelwert von:

$$\bar{t}_{\text{start}} = 214,13 \text{ ms}$$

Die Schwankungen sind mit der Tatsache, dass die Messungen auf einem Multitasking-System mit anderen aktiven Anwendungen durchgeführt worden sind, erklären. Mit ungefähr 210 ms ist die Zeit für den Start der Engine und der ersten Auswertung auch für das angestrebte Einsatzszenario in einem vertretbaren Rahmen. In der Regel wird die Engine nur einmal zur Initialisierung gestartet und steht danach zur Verfügung.

Die Zeiten, die zur Auswertung der Regeln benötigt werden sind in Abbildung B.9 dargestellt. Hierbei sind immer 10 Auswertungsdurchläufe zu einem Wert zusammengefasst. Es zeigt, sich dass die Zeit um sehr gering um den Mittelwert von

$$\bar{t}_{\text{eval}} = 7,59 \text{ ms}$$

schwankt. Die Einzelauswertung der Regeln beträgt ca. $0,7$ ms, was für die Echtzeitanforderungen des Telefoniesystems als ausreichend zu erachten ist.

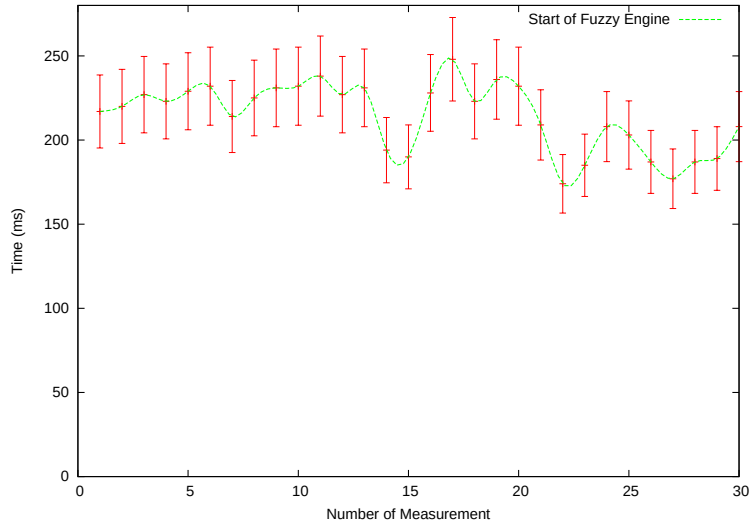


Abbildung B.8: Initialisierungszeit der Fuzzy Logik Engine

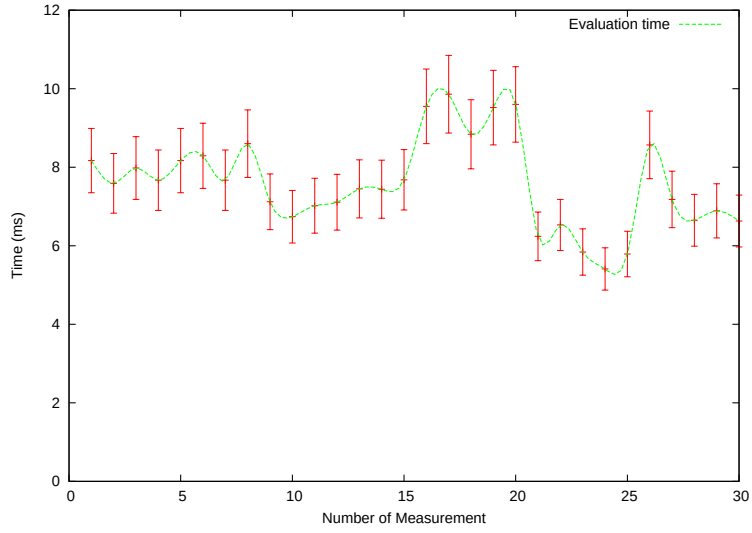


Abbildung B.9: Zeit für die Auswertung der Regeln (10 Durchläufe)

Anhang C

CALL und PIDF-CE

C.1 CALL-Editor zur Generierung der Topologie und Simulation von Aggregationsnetzen

In Unterabschnitt 7.2.2 wurde die Sprache CALL eingeführt. Diese erlaubt die Konfiguration von Aggregationsnetzwerken. In dieser Arbeit wurden ein Großteil der Sensoren simuliert, um das System testen zu können. Die Konfiguration der Topologie eines Context Aggregation Network kann mittels eines Skriptes erfolgen. Dieses wird dann vom Simulator interpretiert.

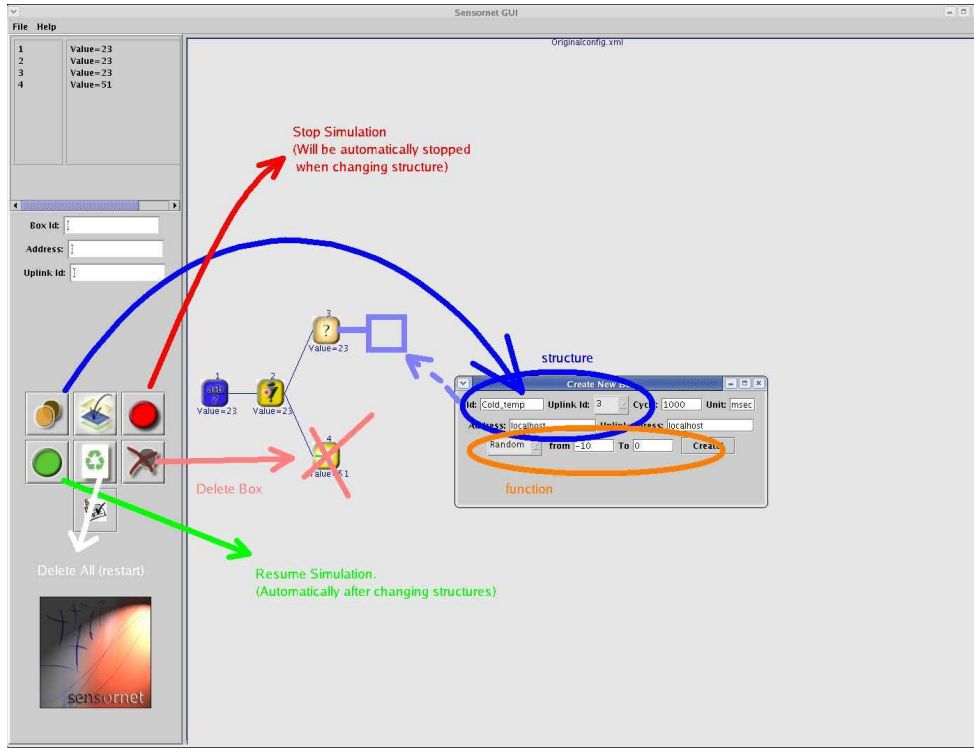


Abbildung C.1: Konfiguration der Operations-Boxen im CALL-Editor

Im Rahmen der Arbeit ist ein graphischer Simulator entstanden, mit dem man die Konfiguration vornehmen kann und anschließend die Toplogie simulieren kann. Der Aufbau des Editors ist in Abbildung C.1 gezeigt. Es zeigt die Konfiguration der Boxen, die die in Abschnitt 7.2 beschriebenen Operatoren realisieren. Die einzelnen Funktionen sind kurz durch Pfeile und Erklärung angegeben.

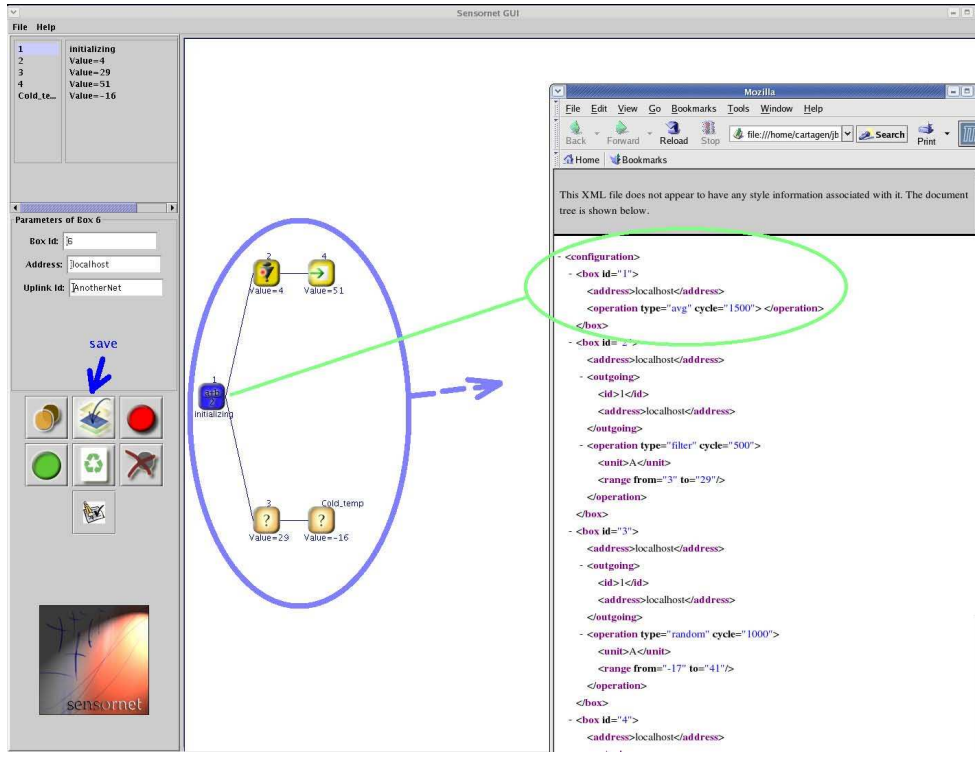


Abbildung C.2: Abspeichern der Konfiguration einer Topologie

Im Simulator des Editors werden die erzeugten CALL-Skripte ausgeführt. Die Werte werden hierbei sowohl in eine Datei für eine spätere Auswertung gespeichert, als auch graphisch angezeigt, wie dies in der Abbildung C.1 oben links und an den Boxen zu sehen ist. Die erstellte Topologie kann abspeichert werden, wie dies in Abbildung C.2 zu sehen ist. Ein Beispiel ist in Listing C.1 dargestellt.

Listing C.1: Beispielkonfigurationskript für CALL

```
<configuration>
    <box id="0" sensor_ID="sensor-0" level="de">
        <address>localhost</address>
        <operation type="store" cycle="1500" />

    </box>

    <box id="1" level="test">
        <address>localhost</address>
        <outgoing>
```

```
            <id>0</id>
            <address>localhost</address>
        </outgoing>
        <operation type="file" cycle="1000">
            <unit>day</unit>
            <file name="data/sensor-output.txt" repeat="true" />
        </operation>
    </box>

    <box id="2" sensor_ID="sensor-2" level="de">
        <address>localhost</address>
        <operation type="store" cycle="1500" />
    </box>

    <box id="3">
        <address>localhost</address>
        <outgoing>
            <id>2</id>
            <address>localhost</address>
        </outgoing>
        <operation type="file" cycle="1100">
            <unit>context</unit>
            <file name="data/day.txt" repeat="true" />
        </operation>
    </box>

    <box id="4" sensor_ID="sensor-4" level0="de">
        <address>localhost</address>
        <operation type="store" cycle="1500" />
    </box>

    <box id="5">
        <address>localhost</address>
        <outgoing>
            <id>4</id>
            <address>localhost</address>
        </outgoing>
        <operation type="file" cycle="1000">
            <unit>context</unit>
            <file name="data/time.txt" repeat="true" />
        </operation>
    </box>
</configuration>
```

C.2 XML Schema für PIDF-CE

Das XML Schema für die Erweiterungen `<direction>` und `<importance-level>` sind nachfolgend aufgeführt:

Listing C.2: XML Schema for <tuple> extensions

```
<?xml version="1.0" encoding="UTF-8"?>
   <xs:schema xmlns="urn:ietf:params:xml:ns:pidf:epidf-tuple"
     xmlns:pidf="urn:ietf:params:xml:ns:pidf"
     xmlns:xs="http://www.w3.org/2001/XMLSchema"
     elementFormDefault="qualified"
     attributeFormDefault="unqualified">

<!- This import brings in the XML language attribute xml:lang ->
     <xs:import namespace="http://www.w3.org/XML/1998/namespace"
       schemaLocation="http://www.w3.org/2001/xml.xsd"/>

     <xs:annotation>
       <xs:documentation xml:lang="en">
            Describes EPIDF tuple extensions for PIDF.
          </xs:documentation>
     </xs:annotation>

     <xs:attributeGroup name="SinceUntil">
    <xs:attribute name="since" type="xs:dateTime"/>
    <xs:attribute name="until" type="xs:dateTime"/>
     </xs:attributeGroup>

     <xs:element name="direction">
       <xs:complexType>
            <xs:simpleContent>
               <xs:restriction base="xs:token">
                 <xs:enumeration value="Tx-Rx"/>
                 <xs:enumeration value="Rx"/>
                 <xs:enumeration value="Tx"/>
               </xs:restriction>
                   <xs:attributeGroup ref="SinceUntil"/>
            </xs:simpleContent>
       </xs:complexType>
     </xs:element>

     <xs:element name="importance-level">
        <xs:complexType>
            <xs:simpleContent>
               <xs:extension base="xs:integer">
                  <xs:attributeGroup ref="SinceUntil"/>
               </xs:extension>
            </xs:simpleContent>
       </xs:complexType>
     </xs:element>
</xs:schema>
```

Das XML Schema der Erweiterungen für das `<status>`-Element sind nachfolgend aufgeführt. Es sind dies die Tags: `<currently-doing>`, `<where>`, `<privacy>`, `<future-status>` und `<past-status>`

Listing C.3: XML Schema for <status> extensions

```
<?xml version="1.0" encoding="UTF-8"?>
   <xs:schema xmlns="urn:kom:epidf"
   xmlns:pidf="urn:ietf:params:xml:ns:epidf-status"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified"
   attributeFormDefault="unqualified">

<!- This import brings in the XML language attribute xml:lang ->
   <xs:import namespace="http://www.w3.org/XML/1998/namespace"
   schemaLocation="http://www.w3.org/2001/xml.xsd"/>

   <xs:annotation>
   <xs:documentation xml:lang="en">
        Describes EPIDF status extensions for PIDF.
   </xs:documentation>
   </xs:annotation>

   <xs:attributeGroup name="SinceUntil">
      <xs:attribute name="since" type="xs:dateTime"/>
      <xs:attribute name="until" type="xs:dateTime"/>
   </xs:attributeGroup>

   <xs:simpleType name="tokenlist">
      <xs:list itemType="xs:token"/>
   </xs:simpleType>

   <xs:element name="doing">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="currently-doing">
               <xs:complexType>
                  <xs:restriction base="xs:token">
                     <xs:enumeration value="holiday"/>
                     <xs:enumeration value="on-the-phone"/>
                     <xs:enumeration value="away"/>
                     <xs:enumeration value="appointment"/>
                     <xs:enumeration value="meeting"/>
                     <xs:enumeration value="in-movement"/>
                     <xs:enumeration value="travel"/>
                     <xs:enumeration value="sleeping"/>
                     <xs:enumeration value="busy"/>
                     <xs:enumeration value="permanent-absence"/>
                  </xs:restriction>
                  <xs:attributeGroup ref="SinceUntil"/>
               </xs:complexType>
            </xs:element>
        <xs:any maxOccurs="unbounded"/>
         </xs:sequence>
     </xs:complexType>
   </xs:element>
```

```
<xs:element name="where">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="place">
          <xs:complexType>
                <xs:restriction base="xs:token">
           <xs:enumeration value="at-home"/>
                   <xs:enumeration value="at-work"/>
                   <xs:enumeration value="working"/>
                   <xs:enumeration value="in-office"/>
                   <xs:enumeration value="quiet"/>
                   <xs:enumeration value="noisy"/>
                   <xs:enumeration value="public"/>
                </xs:restriction>
                <xs:attributeGroup ref="SinceUntil"/>
             </xs:complexType>
         </xs:element>
         <xs:any maxOccurs="unbounded">
      </xs:sequence>
   </xs:complexType>
</xs:element>

<xs:element name="privacy">
   <xs:complexType>
      <xs:simpleContent>
         <xs:restriction base="tokenlist">
             <xs:enumeration value="private"/>
             <xs:enumeration value="public"/>
             <xs:attributeGroup ref="SinceUntil"/>
         </xs:restriction>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>

<xs:element name="future-status" maxOccurs="unbounded">
   <xs:complexType>
      <xs:sequence>
         <xs:extension base="xs:token">
            <xs:attributeGroup ref="SinceUntil"/>
         </xs:extension>
      </xs:sequence>
   </xs:complexType>
</xs:element>

<xs:element name="past-status">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="token">
            <xs:attributeGroup ref="SinceUntil"/>
         </xs:extension>
      </xs:simpleContent>
```

```
      </xs:complexType>
   </xs:element>

</xs:schema>
```

C.2.1 Erweiterungen für Kontextaggregations-Operatoren

Eine Maßgabe des Formates PIDF-CE war die Verwendung bei der Kontextaggregation. Dazu wurde das Format entsprechend erweitert. Nachfolgend ist dies für einige Fusionsoperationen gezeigt.

```
<node name="X">
     <fusion>Dempster-Shafer</fusion>
     <node name="Y">
          <fusion>Neural network</fusion>
          <node name="Z">
               <fusion>Dempster-Shafer</fusion>
          <!- Noise sensor ->
               <sensor id="1">
                    <auth-class>Family & work</auth-class>
                    <time-date>2004-06-15T12:15:33Z</time-date>
                    <decay-function>Linear</decay-function>
                    <value>40</value>
                    <unit>dB</unit>
                    <type>Noise</type>
                    <id>1</id>
                    <owner>Room 305</owner>
                    <dependability>78%</dependability>
                    <frequence>5Hz</frequence>
               </sensor>
          <!-Temperature sensor ->
               <sensor id="2">
                    <auth-class>Work</auth-class>
                    <time-date>2004-06-15T12:15:33Z</time-date>
                    <decay-function>Exponent</decay-function>
                    <value>22</value>
                    <unit>Celsius</unit>
                    <type>Temperature</type>
                    <id>2</id>
                    <owner>Room 305</owner>
                    <dependability>91%</dependability>
                    <frequence>1Hz</frequence>
               </sensor>
          <!- Movement sensor ->
               <sensor id="3">
                    <auth-class>All</auth-class>
                    <time-date>2004-06-15T12:15:33Z</time-date>
                    <decay-function>Linear</decay-function>
                    <value>True</value>
                    <unit>Boolean</unit>
```

```
                <type>Movement</type>
                <id>3</id>
                <owner>Room 305</owner>
                <dependability>86%</dependability>
                <frequence>10Hz</frequence>
           </sensor>
      </node>

<!- Mouse activity ->
      <sensor id="4">
           <auth-class>Work</auth-class>
           <time-date>2004-06-15T12:15:33Z</time-date>
           <decay-function></decay-function>
           <value>True</value>
           <unit>Boolean</unit>
           <type>Mouse</type>
           <id>4</id>
           <owner>Peter</owner>
           <dependability>95%</dependability>
           <frequence>100Hz</frequence>
      </sensor>

<!- Typing activity ->
      <sensor id="5">
           <auth-class>Work</auth-class>
           <time-date>2004-06-15T12:15:33Z</time-date>
           <decay-function>none</decay-function>
           <value>False</value>
           <unit>Boolean</unit>
           <type>Typing</type>
           <id>5</id>
           <owner>Peter</owner>
           <dependability>99%</dependability>
           <frequence>200Hz</frequence>
      </sensor>
</node>

<node name="W">
      <fusion>Fuzzy Aggregation</fusion>

<!- Light sensor ->
      <sensor id="6">
           <auth-class>Work</auth-class>
           <time-date>2004-06-15T12:10:18Z</time-date>
           <decay-function>Linear</decay-function>
           <value>9500</value>
           <unit>Lux</unit>
           <type>Light</type>
           <id>6</id>
           <owner>Room 306</owner>
           <dependability>29%</dependability>
           <frequence>1Hz</frequence>
```

```
          </sensor>

     <!- Light sensor ->
          <sensor id="7">
               <auth-class>Work</auth-class>
               <time-date>2004-06-15T12:14:55Z</time-date>
               <decay-function>none</decay-function>
               <value>8000</value>
               <unit>Lux</unit>
               <type>Light</type>
               <id>7</id>
               <owner>Corridor</owner>
               <dependability>60%</dependability>
               <frequence>4Hz</frequence>
           </sensor>
     </node>
</node>
```

Anhang D

Web Services

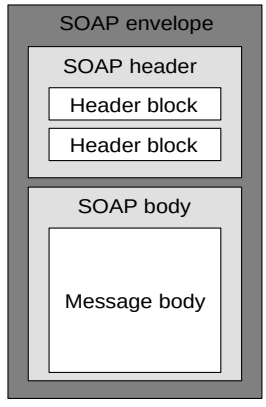
D.1 Simple Object Access Protocol – SOAP

Der Aufbau einer SOAP-Nachricht und ein textuelles Beispiel ist in Abbildung D.1 dargestellt.

D.2 Web Service Description Language – WSDL

Input- und *Output*-Nachrichten beschreiben die Parameter, die an die Operation des Web Services geschickt werden, beziehungsweise den Satz an Rückgabewerten. WSDL-Dokumente können in andere WSDL-Dokumente über das `<import>` Element eingebettet werden, was eine Modularisierung der Dienstbeschreibungen erlaubt.

Eine WSDL-Beschreibung besteht strukturell aus einem Container-Element namens `<definitions>`, welches die Dienstbeschreibung sowie die globalen Definitionen und Namensräume enthält. Die zu verwendenden Datentypen werden innerhalb des `<types>` Elements definiert. Hierbei wird üblicherweise XML Schema Definitions (XSD) verwendet, wobei auch andere XML-Typendefinierungsansätze möglich sind. Die Nachrichten, die zwischen dem Dienst und dem Client ausgetauscht werden können, sind im




```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SOAP-ENV:Envelope
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/
      soap/encoding/"
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
      envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/
      encoding/"
 xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/1999/XMLSchema">
      <SOAP-ENV:Body>
            <ns:getContext
             xmlns:ns1="urn:ContextServer">
                  <user-id xsi:type="xsd:int">
                       190373</user-id>
            </ns:getContext>
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Abbildung D.1: Graphische und textuelle Repräsentation einer SOAP-Nachricht

<messages>-Elements definiert, welche wiederum aus <part> Sub-Elementen bestehen. Die <part> Elemente beschreiben die einzelnen Daten der Nachrichten.

Das <portType>-Element spezifiziert die Schnittstelle des Web Services und beschreibt eine Teilmenge der von dem Dienst unterstützten Operationen. Die dazugehörigen Aktionen sind im <operation>-Element ähnlich zu Java-Methodendefinitionen beschrieben. Die konkret zu verwendenden Protokolle und Datenformate für die abstrakte <portType>-Beschreibung sind im <binding>-Element definiert. Das <service>-Element identifiziert den Web Service durch <port>-Elemente, die jeweils einen Endpunkt für den Dienst darstellen.

Die Anatomie einer Beschreibung in WSDL ist nachfolgend gezeigt. Ein Stern (*) symbolisiert, das ein Elemente mehrfach auftreten kann.

```
<definitions>
   <import>*
   <types>
      <schema></schema>*
   </types>
   <message>*
      <part></part>*
   </message>
   <PortType>*
      <operation>*
         <input></input>
         <output></output>
         <fault></fault>*
      </operation>
   </PortType>
   <binding>*
      <operation>*
         <input></input>
         <output></output>
      </operation>
   </binding>
   <service>*
      <port></port>*
   </service>
</definitions>
```

D.3 Performanzbetrachtung

Ein einfacher Messaufbau zur Bestimmung der Latenzzeiten bei der Verwendung von Web Services wurde erstellt. Die erzielten Messungen sollten einen Eindruck geben, wie lange Aufrufe brauchen und in welchen Komponenten wieviel Zeit verbraucht wird. Hierzu wurden drei Zeitstempel (*time stamps*) genommen und für die Messung verwendet. Der erste Zeitstempel ist die Zeit nach dem Starten des lokalen Clients. Der

zweite Zeitstempel wurde nach der Initialisierung des Aufrufs des Web Services im Client genommen. Nachdem das Ergebnis des Web Services im Client vorliegt, wird der dritte Zeitstempel genommen.

Als Testumgebung fungierten zwei PCs auf x86-Architektur (P4 1.4 GHz, P3 500 MHz) mit Linux Betriebssystem, die über 100 Mbit Ethernet Interfaces an das geswitchte Local Area Network (LAN) angeschlossen waren. Der Web Service und der Client waren in Java unter der Verwendung von Sun's Java WebService Pack [www12] entwickelt worden. Ein Apache Tomcat/Axis Server [www1, www32] stellte den Web Service bereit. Der Client wurde von einer Linux Shell aus gestartet. Das Programm musste sich den Prozessor und den zur Verfügung stehenden Hauptspeicher (786 MB) mit anderen Programmen teilen. Während der Messungen waren keine anderen großen Anwendungen aktiv. Die Messungen wurden überwiegend unter einem Gnome/X11 Desktop durchgeführt. Zusätzlich wurden Vergleichsmessungen mit einem Kommandozeilen-Linux erstellt. Es wurden zwei verschiedene Messszenarien untersucht. Beim ersten Aufbau wurde das Messprogramm gestartet und der Web Service einmal aufgerufen. Anschließend wurde der Web Service noch weitere 99-mal gestartet und die Zeitstempel erfasst.

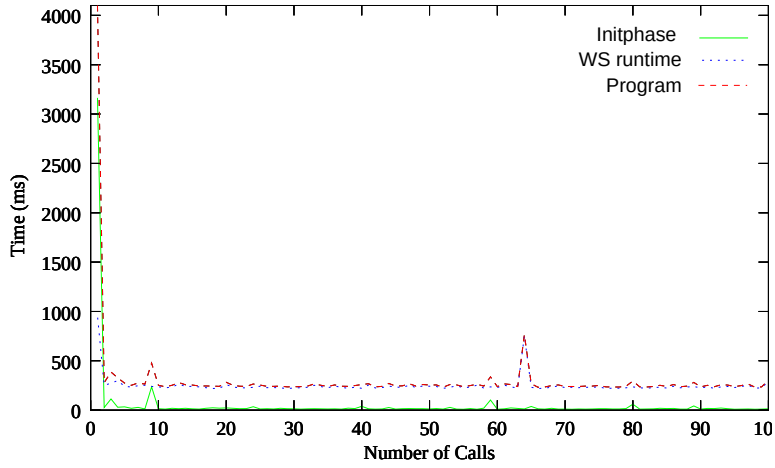


Abbildung D.2: Performanzmessung für den Aufruf eines Funktion eines Web Services

Die Messung ist in Abbildung D.2 gezeigt. Der Graph zeigt deutlich die Unterschiede in der Laufzeit für den ersten Aufruf verglichen mit den übrigen Aufrufen. Der erste Aufruf brauchte ca. 5500 ms, bis das Ergebnis verfügbar war. Nach der Initialisierung wurden jeweils nur noch ca. 400 ms benötigt. Die beiden Phasen wurden anschließend jeweils einzeln untersucht, um eine statistische Auswertung erstellen zu können.

D.3.1 Web Service Einzelaufruf

Der Graph in Abbildung D.3 zeigt das Ergebnis von 262 Programmstarts mit anschließenden Web Service Aufrufen. Die unterste (blau gestrichelte) Linie zeigt die Zeit die für den reinen *Web Service Aufruf* benötigt wird. Die mittlere (grün punktierte) Linie

stellt die Zeit dar, die für die *Initialisierung* verbraucht wird. Die Summe der beiden Zeiten ist in der oberen (rot durchgezogene) Kurve dargestellt.

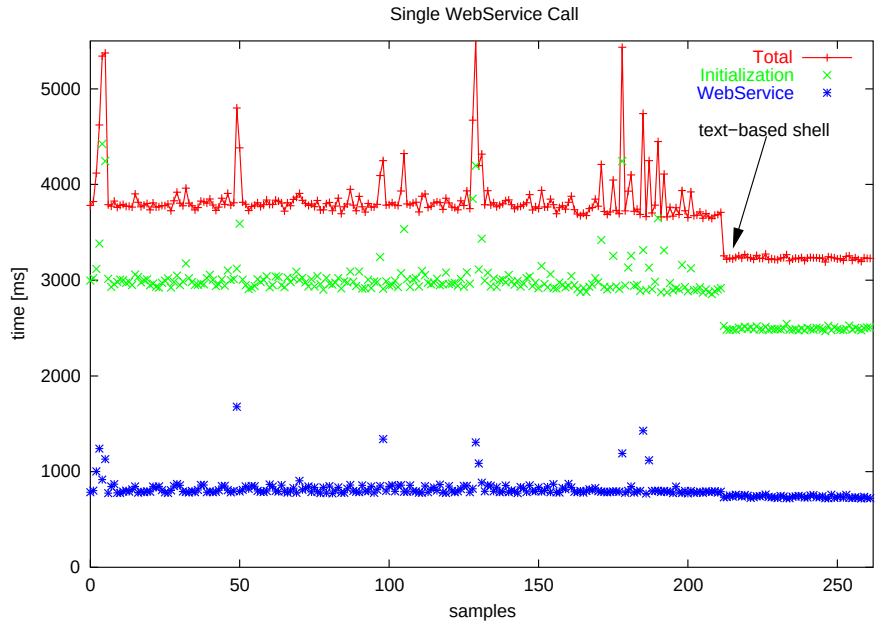


Abbildung D.3: Performanzmessung für den Einzelaufruf des Messprogramms und des Web Services

Der wiederholte Aufruf des Programms führte nicht zu einer Beschleunigung des Aufrufs des Web Services. Es finden also keine Caching-Effekte auf Seiten des Servers statt. Die Zeit für die Initialisierung des Programms liegt immer etwa dreimal höher als die Zeit für die Ausführung des Web Services, von einigen Ausreissern abgesehen. Im letzen Teil der Kurven sind die Werte für die Messungen unter dem Kommandozeilenmodus des Linux-Betriebssystems aufgetragen. Die Messungen sind noch konstanter und liegen bei ca. 80% der Zeiten für die Ausführung unter der graphischen Oberfläche.

In Tabelle D.1 sind die Ergebnisse und die Tendenzen (Mittelwert, Median und Mode) aufgeführt. Zusätzlich wurden noch die Standardabweichung[1], der Minimal- und Maximalwert sowie die Bandbreite der Werte angegeben.

Die Anzahl n der notwendigen Messwerte für die statistischen Aussagen wurde bestimmt, so dass $C = P(\mid \overline{x} - \mu \mid < \varepsilon)$ für ein angegebenes Konfidenzintervall C und maximal erlaubten Fehler ε gültig ist. Mittels (9.2) und $\varepsilon = 100$ wurde die Anzahl der notwendigen Messungen zu $n = 11$ bestimmt. Die Zahl der durchgeführten Messungen ist jedoch weit höher, so dass die Voraussetzungen erfüllt sind.

[1]Die Standardabweichung ist mit s statt mit dem üblichen σ angegeben, da es sich um Messwerte der Gesamtpopulation handelt.

Tabelle D.1: Indizes für die Auswertung der Messung der Startzeit eines Web Service Clients

Indizes	Init	Web Service	Total
Mittelwert	2925,06	812,10	3737,16
Median	2952,50	792,00	3772,00
Mode	2949,00	785,00	3785,00
Std Abw (s)	171,54	51,54	206,15
Min	2468	719	3190
Max	4424	1678	5503
Spanne	1956	959	2313

D.3.2 Web Service Mehrfachaufruf

Um die Zeit für den Aufruf eines Web Services aus einer bereits initialisierten Anwendung zu messen, wurde die entfernte Operation aus einer Schleife heraus aufgerufen. Der jeweils erste Messwert wurde dabei ignoriert. Die Ergebnisse der Messung sind in Abbildung D.4 dargestellt. Die periodischen Peaks sind wahrscheinlich auf den Garbage Collector zurückzuführen. Die statistisch relevanten Indizes sind in Tabelle D.2 zusammengetragen.

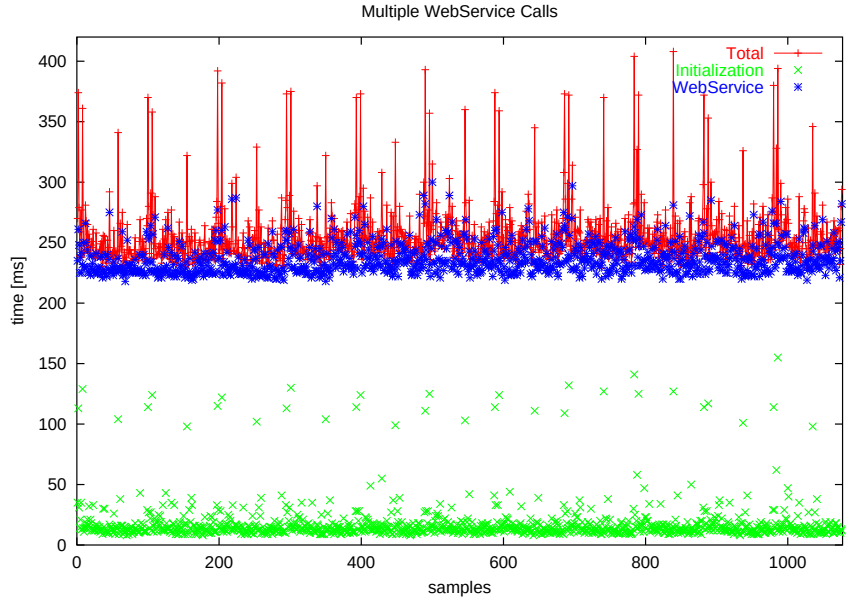


Abbildung D.4: Performanzmessung für Web Service Mehrfachaufrufe

Tabelle D.2: Indizes für die Auswertung der Messung der Startzeit eines Web Service Clients

Indizes	**Init**	**Web Service**	**Total**
Mittelwert	18,51	236,57	255,08
Median	13	233	248
Mode	12	227	241
Std Abw (s)	18,93	12,92	24,68
Min	8,00	218,00	229,00
Max	155,00	300,00	408,00
Spanne	147,00	82,00	179,00

D.3.2.1 Fazit

Die Aufrufzeiten aus einem bereits laufenden Programm sind derart, dass sie sich für die Verwendung in dem avisierten System eignen. Der Aufruf von Programmen, die in C/C++ unter der Verwendung von gSOAP [www9] geschrieben worden sind, ist sogar noch schneller. Darüberhinaus bietet die Web Service Infrastruktur alle benötigten Kommunikationstypen. Das "multiple languages – multiple platforms" Paradigma von Web Services erlaubt die Integration von heterogenen Systemen, Komponenten und Anwendungen, die auf verschiedenen Plattformen laufen und mit unterschiedlichen Programmiersprachen einbezogen werden sollen, in das Gesamtsystem.

Anhang E

CPL-Editor

E.1 Bedienung des CPL-Editors

Jede unterstützte Komponente und Kontrollstruktur kann durch einen Rechtsklick auf eine freie Stelle der Arbeitsfläche aus dem Kontextmenü ausgewählt werden. Die Komponente wird in Form einer Box dargestellt. Jede der Boxen kann $1 \ldots n$ Ausgänge haben, die mit einem Eingang einer nächsten Box verbunden werden muss. *Action*-Elemente besitzen als einzige Komponente keinen Ausgang, da sie das Skript terminieren. Zur Verbindung der Boxen fügt der Nutzer einen Ausgang hinzu und zieht den entstehenden Pfeil mit der Maus auf die zu verbindende Box.

Die Parametrisierung der Elemente und deren Verbindungen kann durch Doppelklick auf die Boxen oder die Pfeile aktiviert werden. Es öffnen sich die dazugehörigen Menüs, in denen die Parameter spezifiziert werden können. Das Parametrisierungsmenü teilt sich in die zwei Untermenüs *Settings* und *Advanced*. Die Basiseinstellung eines Elementes umfasst notwendige Felder, wie zum Beispiel das Adress-Feld (incoming, originating) bei einem Address-Switch. Im erweiterten Menü sind nähere Spezifikationen zu den Operatoren untergebracht.

Jede Verbindung zwischen den einzelnen Komponenten kann mit einem Bezeichner (*label*) versehen werden, der die Verbindung beschreibt. Die verbundenen Elemente bleiben auch bei einer Neuplatzierung eines Elements miteinander verbunden und arrangieren sich neu.

E.2 Konfigurationsdatei für den CPL-Editor

Ein Ausschnitt aus der Konfigurationsdatei für den CPL-Editor ist nachfolgend gegeben. Es zeigt die Erweiterung *Context Switch*.

Listing E.1: Context Switch configuration

```
<actionDef disabled="false" name="context-switch" classRef="
    switch">
    <icon standard="switch-context.gif" active="switch-context-
        over.gif" moved="switch-context-over.gif" menu="switch-
        context-mini.gif">
```

```
      <arrowIcon x="-12" y="12" standard="arrow.gif" active="
         arrow-over.gif"></arrowIcon>
      <vertex x="-28" y="-28"></vertex>
      <vertex x="28" y="-28"></vertex>
      <vertex x="28" y="28"></vertex>
      <vertex x="-28" y="28"></vertex>
   </icon>
   <translation lang="en" term="context-switch"></translation>
   <description lang="en">
      <xingtext><![CDATA[
         Context switches allow a CPL script to make
            descisions
         based on the current user's context.
         ]]>
      </xingtext>
   </description>
   <description lang="de">
      <xingtext><![CDATA[
         Der  <code>context-switch</code> erlaubt
         einem CPL-Skript, Entscheidungen in Abh&auml;ngigkeit
             von
         aktuellen Kontexts des Nutzers zu treffen.
         ]]>
      </xingtext>
   </description>
   <paramDef name="source" required="true" advanced="false">
      <translation lang="en" term="source"></translation>
      <description lang="en">
         <xingtext><![CDATA[
            source of the DB
            ]]>
         </xingtext>
      </description>
      <valueDef type="URL"></valueDef>
      <defaultValue set=""></defaultValue>
   </paramDef>
  <!-- login name for DB-->
   <paramDef name="login" required="false" advanced="true">
      <translation lang="en" term="login"></translation>
      <description lang="en">
         <xingtext><![CDATA[
            name of database
            ]]>
         </xingtext>
      </description>
      <valueDef type="string"></valueDef>
   </paramDef>
  <!-- login name for DB-->
   <paramDef name="password" required="false" advanced="true">
      <translation lang="en" term="password"></translation>
      <description lang="en">
         <xingtext><![CDATA[
```

```
            password
            ]]>
         </xingtext>
      </description>
      <valueDef type="string"></valueDef>
   </paramDef>
  <!-- name OF DB-->
   <paramDef name="db-name" required="true" advanced="true">
      <translation lang="en" term="db-name"></translation>
      <description lang="en">
         <xingtext><![CDATA[
            name of database
            ]]>
         </xingtext>
      </description>
      <valueDef type="string"></valueDef>
   </paramDef>
  <!-- name OF DB-->
   <paramDef name="table-name" required="true" advanced="true"
      >
      <translation lang="en" term="table-name"></translation>
      <description lang="en">
         <xingtext><![CDATA[
            name of table
            ]]>
         </xingtext>
      </description>
      <valueDef type="string"></valueDef>
   </paramDef>
  <!-- check for actual context-->
   <outputDef name="context" minOccurs="1" maxOccurs="*"
      reqParam="true">
      <translation lang="en" term="context"></translation>
      <description lang="en">
         <xingtext><![CDATA[
            Each <code>context</code> output defines a
            possible continuation of the script processing.
               Exactly one
            of the parameters
            <code>is</code> or <code>contains</code>
            specified.
            ]]>
         </xingtext>
      </description>
      <description lang="de">
         <xingtext><![CDATA[
            Jeder der <code>context</code>-Ausg&auml;nge
            beschreibt eine m&ouml;gliche Fortsetzung des
               Skriptablaufs.
            Von den Parametern
            <code>is</code> und <code>contains</code>
            muss genau einer angegeben werden.
```

```
            ]]>
         </xingtext>
      </description>
      <paramDef name="is" required="false" advanced="false">
         <translation lang="en" term="is"></translation>
         <description lang="en">
            <xingtext><![CDATA[
               The specified subfield must exactly match the
               given value.
               ]]>
            </xingtext>
         </description>
         <description lang="de">
            <xingtext><![CDATA[
               Der im subfield spezifierte Teil der Adresse
                  muss
               exakt mit dem angegebenen Wert &uuml;
                  bereinstimmen.
               ]]>
            </xingtext>
         </description>
         <valueDef type="string"></valueDef>
      </paramDef>
      <paramDef name="not" required="false" advanced="false">
         <translation lang="en" term="not"></translation>
         <description lang="en">
            <xingtext><![CDATA[
               The specified subfield must not match the
               given value.
               ]]>
            </xingtext>
         </description>
         <description lang="de">
            <xingtext><![CDATA[
               Der im subfield spezifierte Teil der Adresse
                  muss
               exakt mit dem angegebenen Wert &uuml;
                  bereinstimmen.
               ]]>
            </xingtext>
         </description>
         <valueDef type="string"></valueDef>
      </paramDef>
      <paramDef name="contains" required="false" advanced="
         false">
         <translation lang="en" term="contains"></translation>
         <description lang="en">
            <xingtext><![CDATA[
               Applies only if the subfield parameter is set
                  to
               <code>display</code>.
               <br>
```

```
                Matches if the display name contains the
                   argument as a
                substring.
                ]]>
             </xingtext>
          </description>
          <valueDef type="string"></valueDef>
       </paramDef>
       <!--
       <constraints>
          <!- - @contains requires ../@subfield="display" - ->
          <assert test="not(@contains␣and␣../@subfield!='
             display')"/>
          <!- - @subdomain-of requires
                ../@subfield="host" or ../@subfield="tel" - ->
          <assert test="not(@subdomain-of␣and␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
             ␣␣␣␣␣␣(../@subfield!='host'␣or␣../@subfield!='tel
             '))"/>
       </constraints>
       -->
    </outputDef>
    <outputDef name="otherwise" minOccurs="0" maxOccurs="1"
       reqParam="false">
       <translation lang="en" term="otherwise"></translation>
       <description lang="en">
          <xingtext><![CDATA[
             This output is true if no other condition matches.
             If it is specified, it has to be the last output.
             ]]>
          </xingtext>
       </description>
       <description lang="de">
          <xingtext><![CDATA[
             Dieser Ausgang wird benutzt, wenn keine der in den
                anderen
             Ausg&auml;ngen spezifizierten Bedingungen zutrifft
                . Wenn
             <code>otherwise</code> angegeben wurde, muss es
             der letzte von allen Ausg&auml;ngen sein.
             ]]>
          </xingtext>
       </description>
    </outputDef>
 </actionDef>
```

E.3 CPL-Elemente Konfiguration

Die Parametrisierungsmenüs der vier in dieser Arbeit eingeführten Elemente für CPL im CPL-Editor sind nachfolgend gezeigt.

E.3.1 Context Lookup

Die *Context Lookup*-Komponente stellt analog zu anderen definierten Lookup-Komponenten wie *Location-Lookup* Informationen für die CPL-Engine bereit. Diese Komponente stellt den Kontext des Nutzers in Form eines Kontext-Objektes im PIDF-CE-Format zur Verfügung. Der Kontext wird von einem ContextServer oder einer anderen Quelle bezogen. Das Symbol und die Parametrisierungsmenüs für die Komponente sind in Abbildung E.1 abgebildet.

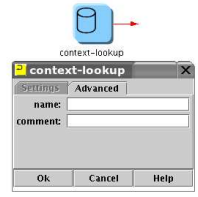


Abbildung E.1: CPLEditor-Menü für Context Lookup

Über das Parametrisierungsmenü wird dem Element mitgeteilt, unter welcher Adresse sich die Kontextquelle befindet. Zusätzlich können Nutzername `login` und Passwort `password` angegeben werden, die für eine Authentisierung bei der Kontextquelle notwendig sein könnten. Um eine Weiterverarbeitung des eingegangenen Anrufs zu gewährleisten, kann ein Timeout angegeben werden, nach dessen Ablauf mit einer Standardprozedur (*default*) fortgefahren wird.

An Ausgängen (*outputs*) sind bei dieser Komponente *success*, *notfound* und *failure* definiert. So kann in Abhängigkeit des Ergebnisses der Kontextabfrage (erfolgreich, Kontext nicht gefunden, Fehler) verzweigt werden. Die jeweiligen Ausgänge dieser Komponente können nicht parametrisiert werden.

E.3.2 Context Notify

Das *Action*-Element *Context Notify* kann innerhalb der *Context Lookup*-Komponente zum Versenden eines Kontexts in Form einer PIDF-CE-Repräsentation an einen Anrufer verwendet werden. Die graphische Repräsentation des CPL-Elements sowie die dazugehörigen Parametrisierungsmenüs sind in Abbildung E.2 gezeigt.

Der Nutzer kann im *Advanced*-Menü für den Kontext die Detailstufe `detail level` festlegen, in der der anfragende Nutzer die Information enthält. Diese Stufen entsprechen den in PIDF-CE definierten Detailstufen. Das Element hat die definierten Ausgänge *process*, *cancel*, *noanswer* und *failure*. Der Ausgang *noanswer* kann mit einer definierten Zeitspanne, während der auf die Antwort gewartet wird, parametriert werden. Die weiteren Ausgänge lassen sich nicht weiter parametrisieren.

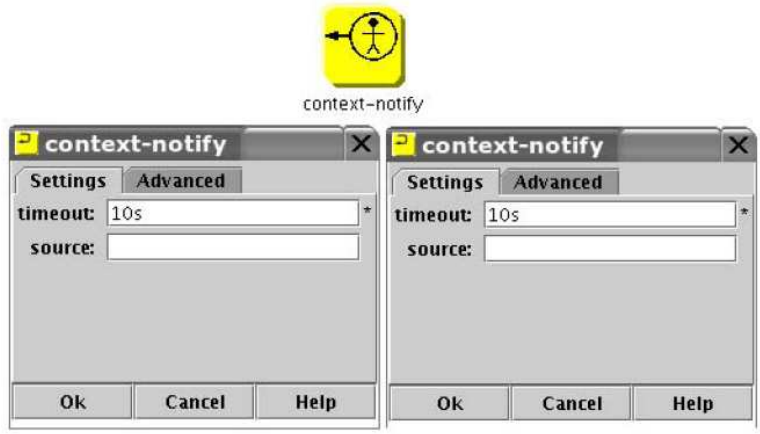


Abbildung E.2: CPLEditor-Menü für Context Notify

E.3.3 Context Switch

Die *Context Switch*-Komponente ist eine bedingte Verzweigung wie diese beispielsweise im Standard-CPL vom *Address Switch* für die Bewertung von SIP-Addressen zur Verfügung steht. Der *Context Switch* verzweigt in den nächsten Pfad eines CPL-Skriptes in Abhängigkeit vom aktuellen Kontext des Nutzers. Hierzu wird der Kontextbezeichner λ bewertet.

Um die Bedingung zu testen, stehen die beiden Übereinstimmungs-Operatoren `Is` und `Contains` zu Verfügung, die auf eine vollständige Übereinstimmung bzw. auf eine Teilübereinstimmung prüfen. Zusätzlich wurde noch der Negationsoperator `Not`, mit dem die Bedingungen für eine Übereinstimmung negiert werden können, eingeführt.

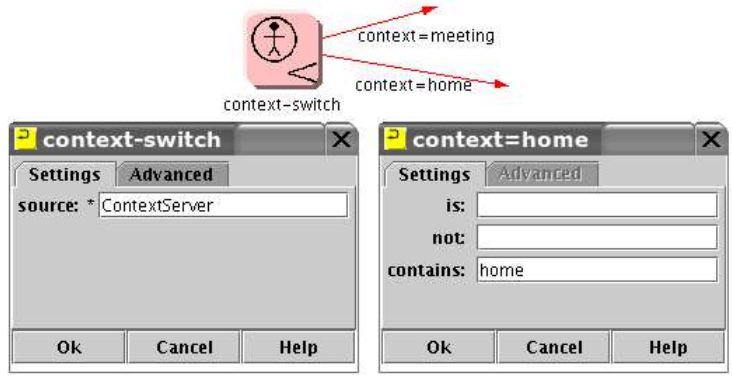


Abbildung E.3: CPLEditor-Menü für Context Switch

Die Parametrisierungsmenüs des Elements sind in Abbildung E.3 abgebildet. So kann

im *Advanced*-Menü eingestellt werden, von welcher Quelle der aktuelle Kontext bezogen wird. Standareinstellung ist, dass der Kontext durch die Context Lookup-Komponenten bereits dem
System zur Verfügung gestellt worden ist. Zusätzlich kann der Nutzername und ein Passwort zur Authentisierung angeben werden. Das *Settings*-Menü zeigt für die Ausgänge die Felder für die verschiedenen Übereinstimmungsmöglichkeiten an, die zum Testen genutzt werden können.

E.3.4 Answer Switch

Der *Answer Switch* verzweigt in Abhängigkeit von einer Antwort des Anrufers. Die Frage wurde zuvor vom Angerufenen an den Anrufer gesendet. Ähnlich wie bei der *Context Switch*-Komponente können verschiedene Übereinstimmungsvarianten gewählt werden. In der Konfiguration der Komponente kann das Feld ausgewählt werden, das in der SIP-Nachricht zur Bewertung herangezogen wird. Im Parametrisierungsmenü der Ausgänge kann die jeweilige Zeichenkette in den Feldern `Is`, `Contains` und `Not` eingetragen werden.

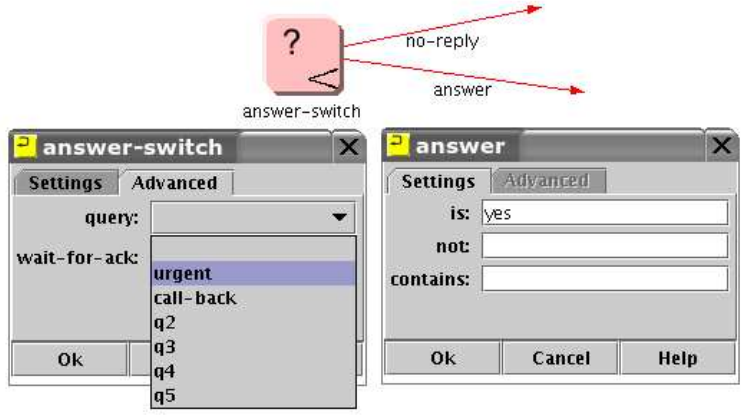


Abbildung E.4: CPLEditor-Menü für Answer Switch

Anhang F

Session Initiation Protocol: Erweiterungen

Eine Auflistung der für SIP definierten Methoden und Responses sind nachfolgend angegeben. Der Standard-Verbindungsaufbau ist ebenfalls gezeigt, um diese mit den in Unterabschnitt 7.4.4 gezeigten Signalisierungsabläufen vergleichen zu können.

F.1 SIP Methods und Responses

Neben den 6 im Kernstandard definierten und in Tabelle 2.1 aufgeführten Methoden, sind weitere Methoden spezifiziert worden. Diese sind in Tabelle F.1 kurz beschrieben.

F.1.1 SIP-Responses

Die Response-Codes im SIP beinhalten die Response-Codes von HTTP/1.1 [FGM+99] und erweitern sie. Alle in HTTP/1.1 nicht definierten Codes haben die Codenummer $x80$ oder größer, um Konflikte mit künftigen HTTP/1.1 Codes zu vermeiden. Zusätzlich definiert das SIP eine neue Klasse: $6xx$. Die Fehlercodeklassen sind nachfolgend beschrieben. Tabelle F.2 enthält die in RFC 3261 definierten Responses (Statusnachrichten).

1xx - Information Information-Responses signalisieren, daß der kontaktierte Server zur Zeit nicht in der Lage ist, den eingegangen Request definitiv zu beantworten. Der kontaktierte Server leitet weitere Aktionen ein, um den Request zu beantworten. Der Client wartet auf die Antworten der vom Server eingeleiteten Aktionen. Hat der Server genauere Informationen erhalten, schickt er ohne weitere Prompts eine Nachricht an den Client. Ein Server sollte eine Meldung der Art $1xx$ dann senden, wenn er glaubt, dass die endgültige Antwort auf den Request in mehr als 200 ms erfolgt. Es ist einem Server freigestellt, diese Informationen zu verschicken.

Clients müssen keinen ACK auf $1xx$-Responses senden. Da Server nicht verpflichtet sind, eine $1xx$-Meldung abzusetzen, ist es teilweise ratsam, dass Clients nach gewissen Intervallen erneut einen Request senden, um Informationen über den augenblicklichen Stand des Rufaufbaus zu erhalten.

2xx - Erfolgreich Der Request ist erfolgreich abgearbeitet worden und wird beendet.

Tabelle F.1: Erweiterte SIP-Methoden mit ihren Bedeutungen

Methode	Bedeutung
SUBSCRIBE	Anmelden einer Notifikation auf die Änderungen von Ereignissen bei einer anderen Entität. Die Änderung wird asynchron und bis zum Ende der Subskription gesendet.
NOTIFY	Meldung von Ereignissen. Die Rückmeldung auf die Änderung eines Ereignisses wird an die vorher subskribierten Entitäten gesendet.
REFER	Aufforderung an eine andere Entität, einen weitere Entität zu kontaktieren, wie dies z. B. im Dienst *Anrufweiterleiten* der Fall ist.
MESSAGE	Transport von Kurznachrichten, wie z. B. für den Dienst *Instant Messaging*
PRACK	Kurzform von *Provisional Response ACKnowledge*. Diese Methode ist eine Antwort auf *provisional repsonses* aus der $1xx$-Reihe. Wird für die Unterbrechung der Etablierung der Sitzung, z. B. für die Aushandlung und Reservierung Dienstgüteparameter, eingesetzt.
UPDATE	Modifikation der Parameter einer Sitzung, während diese noch aufgebaut wird.
INFO	Transport von Steuer- und Kontrollinformationen, die außerhalb einer SIP-Sitzung liegen. So können z. B. ISDN bzw. ISUP-Nachrichten durch das IP-Netz getunnelt werden.

3xx - Zurückgeleitet Antworten der Art $3xx$ geben Auskunft über einen neuen Standort des gewünschten Benutzers oder über eine alternative Adresse. Eine aktuelle Suche sollte nach deren Erhalt terminiert werden. Je nach Wunsch des Teilnehmers kann eine neue Suche mit den neuen Erkenntnissen gestartet werden. Bei der Rückmeldung ist darauf zu achten, dass die Nachricht nicht an eine des im `Via:`-Headerfeld des `Contact:`-Headers eingetragene Adresse des vorangegangenen Requests geschickt wird. Um solche Schleifen zu vermeiden, vergleichen User-Agent-Clients oder Proxies den Adressaten der eingegangenen $3xx$-Antwort mit den Eintragungen früher kontaktierter Redirect-Server.

4xx - Client-Fehler Antworten der $4xx$-Reihe sind genaue Antworten auf Fehler von einzelnen Servern. Der Client sollte nicht den gleichen Request ohne jegliche Modifikation wiederholen.

5xx - Server-Fehler $5xx$-Antworten treten auf, wenn Server selbst fehlerhaft sind. Es sind keine definitiven Fehler. Eine laufende Suche nach einem Teilnehmer muß aufgrund eines solchen Fehlers nicht abgebrochen werden, wenn sie auf einem anderen Server, der noch nicht in die Suche einbezogen wurde, fortgesetzt werden kann.

6xx - Globaler Fehler $6xx$-Responses signalisieren, daß der Server definitive Informa-

tionen über einzelne Teilnehmer besitzt und nicht nur über bestimmte Instanzen. Jede weitere Suche nach diesen Teilnehmern ergeben Fehler, noch nicht abgeschlossene Suchen werden abgebrochen.

Tabelle F.2: SIP-Statusmeldungen

Response	Bedeutung
100 Trying	Es wird versucht, eine Verbindung aufzubauen
180 Ringing	Anruf wird beim Teilnehmer signalisiert
181 Call is being forwarded	Anruf wird weitergeleitet
182 Queued	Anruf befindet sich in der Warteschleife des Teilnehmers
183 Session progress	Sitzung ist im Aufbau
200 OK	Request ist erledigt. Informationen werden nach Art der benutzten Request-Methode zurückgegeben.
300 Multiple choices	Auswahlmöglichkeit
301 Moved permanently	Permanent geänderte Kontaktadresse
302 Moved temporarily	Temporär geänderte Kontaktadresse
305 Use proxy	Benutze Proxy: Auf die angefragte Resource kann nur über Proxy zugegriffen werden.
380 Alternative service	Alternative Dienste: Anruf war nicht erfolgreich, alternative Dienste sind möglich.
400 Bad request	Falscher Request
401 Unauthorized	Nicht authorisiert
402 Payment requiered	Bezahlung nötig
403 Forbidden	Verboten
404 Not found	Nicht gefunden
405 Method not allowed	Methode nicht erlaubt
406 Not acceptable	Nicht akzeptabel
407 Proxy authentication required	Proxy-Authentifikation nötig
408 Request timeout	Request-Timeout
410 Gone	Existiert nicht
413 Request entity too large	Request-Entity zu groß
414 Request-URI too long	Request-URI zu lange
415 Unsupported media type	Medientype wird nicht unterstützt
416 Unsupported URI scheme	Nicht-unterstütztes URI Schema
420 Bad extension	Falsche Erweiterung
421 Extension required	Erweiterung nötigt
423 Interval too brief	Intervall zu kurz
480 Temporarily unavailable	Temporär nicht erreichbar
481 Transaction does not exist	Anruftransaktion existiert nicht
482 Loop detected	Schleife entdeckt
483 Too many hops	Zuviele Hops
484 Address incomplete	Unvollständige Adresse
485 Ambiguous	Mehrdeutige Adresse
486 Busy here	Hier ist besetzt

wird auf der nächsten Seite fortgesetzt

Tabelle F.2 – fortgesetzt von vorheriger Seite

Response	Bedeutung
487 Request terminated	Request terminiert
488 Not acceptable here	Nicht akzeptiert
491 Request pending	Aktiver Request
493 Undecipherable	Nicht-entschlüsselbar
500 Server internal error	Serverinterner Fehler
501 Not implemented	Nicht implementiert
502 Bad gateway	Falsches Gateway
503 Service unavailable	Dienst ist nicht verfügbar
504 - Gateway timeout	Zeitüberschreitung am Gateway
505 Version not supported	Version wird nicht unterstützt
513 Message too large	Nachricht ist zu groß
600 Busy everywhere	Überall besetzt
603 Decline	Abgelehnt
604 Does not exist anywhre	Existiert nirgends
606 Not acceptable	Wird nicht akzeptiert

F.2 Signalisierungsabläufe

Zur Abbildung von Signalisierungsabläufe wird in dieser Arbeit die von der ITU standardisierten Message Sequence Charts (MSC) [Int99b]. MSC ist eine Sprache, welche die Interaktion zwischen unabhängigen nachrichtenverarbeitenden Instanzen beschreibt. Formal ist ein ein MSC ein Tupel, das folgendermaßen definiert ist:

Definition F.1 (Message Sequence Chart)
*Ein **MSC** ist ein Tupel $M = \langle \mathcal{P}, E, \mathcal{A}, \ell, m, < \rangle$. Hierbei ist $\mathcal{P}$ eine endliche Menge an Prozessen, E eine endliche Menge an Ereignissen, $\mathcal{A}$ eine endliche Menge an Namen für Nachrichten und (lokalen) Aktionen. $\ell : E \rightarrow \mathcal{T}$ ist ein Bezeichner, der die Ergeignisse auf einen Type $\mathcal{T}$ abbildet. $m : S \rightarrow R$ ist eine Bijektion, die gesendete Nachrichten mit den dazugehörigen Antworten zuordnet. Eine azyklische Relation der zeitlichen Anfolge der Ereignisse ist mit $< \subseteq E \times E$ gegeben.*

Ein Beispiel-MSC ist in Abbildung F.1 abgebildet.

F.2.1 Verbindungsaufbau

Bei der INVITE-Transaktion handelt es sich um eine 3-Wege-Aushandlung (*3-way handshake*) zum Aufbau einer multimedialen Sitzung. Der fehlerfreie Ablauf einer solchen Aushandlung ist in Form eines MSC in Abbildung F.2 dargestellt. Ausgehend von der für den Protokollablauf einer INVITE-Transaktion zuständigen Zustandsmaschine, die

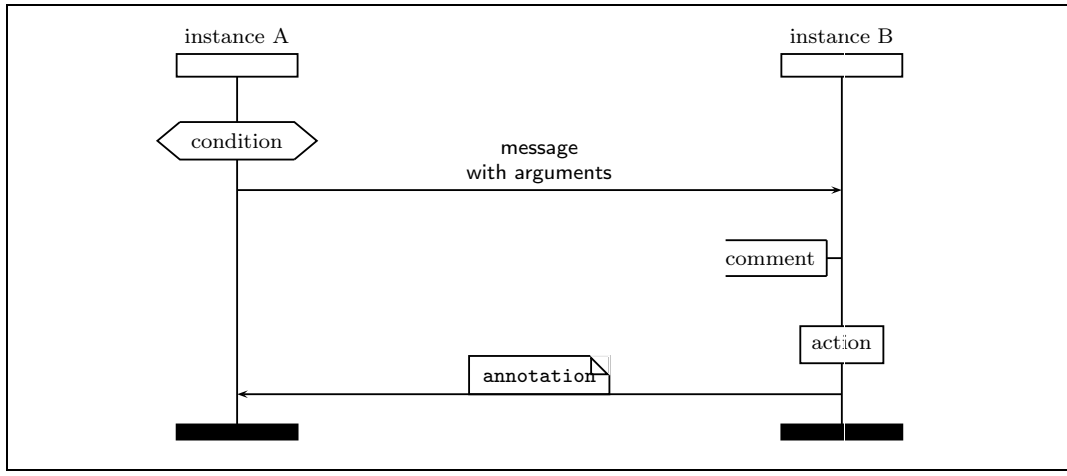


Abbildung F.1: Beispiel-MSC mit Erläuterungen

für jeweils den Anrufer und den Angerufenen in den Abbildungen 9.9(a) und 9.9(b) zusehen sind, wurden die Erweiterungen für den UA integriert.

F.3 SIP Entitäten in der VOCAL-Suite

Für die Erweiterungen der Arbeiten wurden die Komponeten der VOCAL-Suite als Ausgangsbasis verwendet. Die Software wird von VOVIDA [www34] unter einer BSD-ähnlichen Lizenz als Open-Source Projekt angeboten. Das Projekt stellt eine komplette Kommunikationsplattform für SIP-basierte Telefonie in hoher industrieller Softwarequalität zur Verfügung.

In Abbildung F.3 ist eine Übersicht des Systems gezeigt. Die wesentlichen Server sind:

Redirect Server Zuständig für die Registrierung von Endgeräten und Weiterleitung von SIP-Nachrichten.

Marshal Server Empfang von SIP-Nachrichten, Authentifikation der Benutzer

Provisioning Server Speichern und Abrufen der Einstellungen von Benutzern

CDR Server Gebührenberechnung für Teilnehmer

Policy Server Überwachung der Quality of Service (QoS)

Feature Server (FS) Ausführen von Features / CPL Skripts

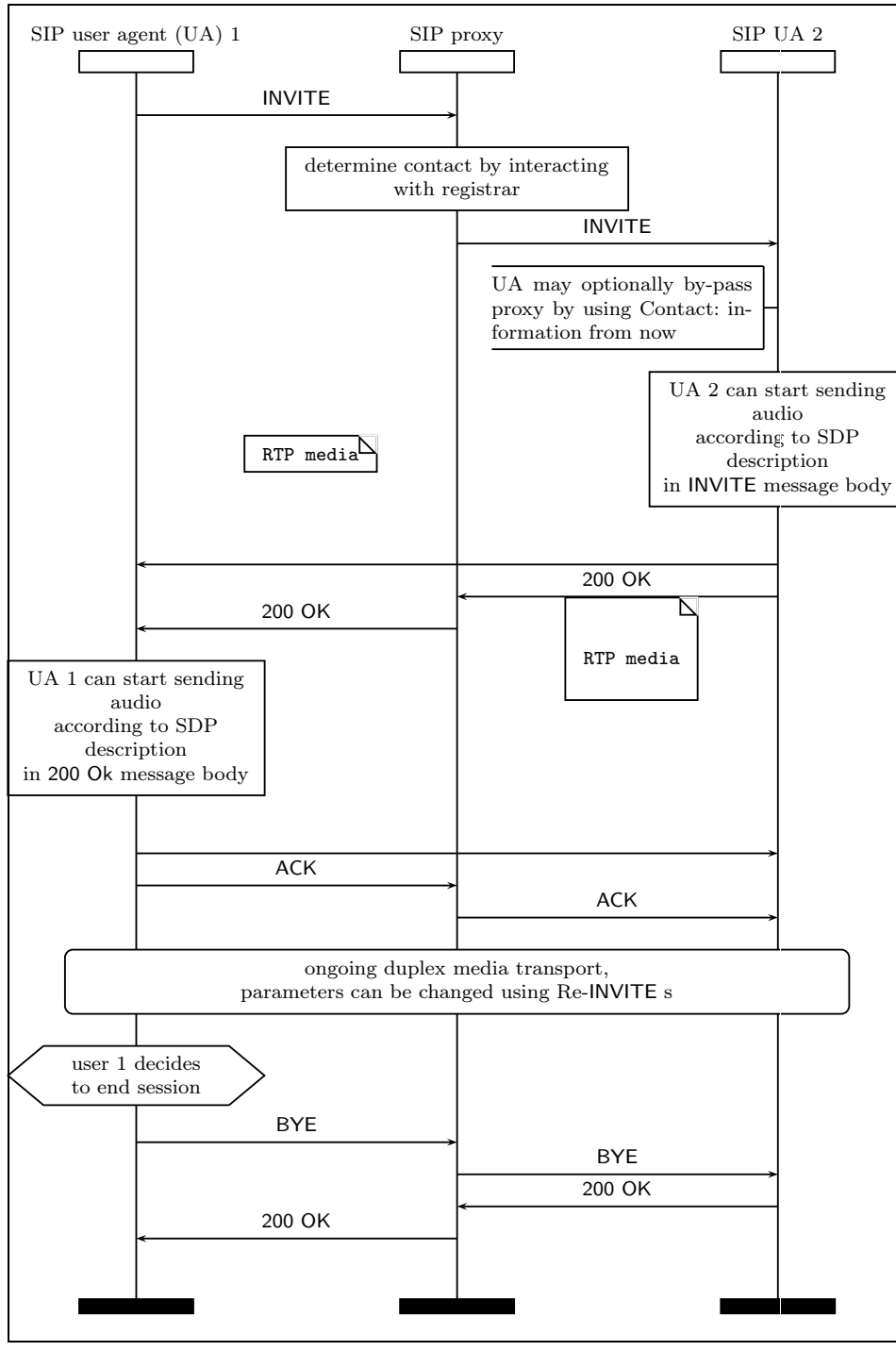


Abbildung F.2: SIP Signalisierung zum Aufbau einer Multimediasitzung (angepasst aus [Ack03])

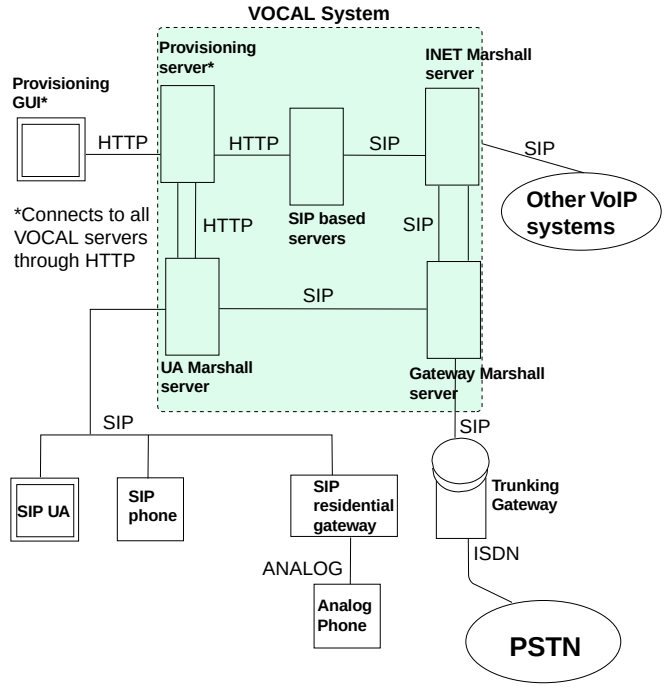


Abbildung F.3: Übersicht der Server und Komponenten der VOCAL SIP IP-Telephony Suite

F.3.1 Der Provision Server

Der Provision Server (PS) speichert die Benutzereinstellungen aller dort registrierten Teilnehmer. Der PS erlaubt den Zugriff über ein Java Webinterface. Damit ist der Server für den Administrator und die Benutzer erreichbar. Die Java GUI ermöglicht es dem Administrator neue Benutzer einzurichten, existierende zu modifizieren und die Systemeinstellungen zu bearbeiten.

Jeder User Agent , der sich an dem Server registrieren können soll, muss voher mit einem Eintrag im Provisioning Server administriert worden sein. Hierbei können Benutzerdaten wie z. B. Passwort, Gruppenzugehörigkeit und erlaubte Features eingestellt werden.

Einzelne Dienste (hier Features genannt) können im Java Webinterface eingerichtet werden. Diese müssen jedoch erst vom Administrator aktiviert werden. Einige Features können vom Nutzer selbst parametrisiert werden, wie diese z. B. bei einer Anrufweiterlung der Fall ist. Eine Reihe von Features wurden bereits vordefiniert. Nach der Parametrisierung der Dienste werden aus den eingegebenen Informationen selbständig das gewünschte CPL-Skript generiert. Folgende Dienste sind definiert:

- Caller ID Blocking (Unterdrückung der Rufnummernübermittlung)
- Call Forward All (Anrufe weiterleiten)
- Call Forward No Answer (Anrufweiterleitung bei nicht-antwort)

- Call Forward Busy (Anrufweiterleitung bei besetzt)
- Call Blocking (Ausgehende Verbindungen einschränken)
- Call Screening (Anzeige von Rufnummern)
- Voicemail (Anrufbeantworter)

F.3.2 Der Feature Server

Der Feature Server (FS) ist eine Server-Komponente. Sobald im Provisioning Server ein Feature für einen Teilnehmer aktiviert wird, wird ein neuer Task des Feature Servers mit dem dazugehörigen CPL-Skript gestartet. Ein solcher Task erhält einen eigenen Port, über welchen das Feature angesprochen werden kann. Bei Aktivierung eines Features für einen Teilnehmer, werden Port und CPL-Skript des Features im Provisioning Server gespeichert.

F.3.3 SIP User Agent

Der VOVIDA User Agent ist in Form einer Zustandsmaschine aufgebaut und von sehr komplexer Struktur, wie dies auch in Abbildung F.4 zu sehen ist. Die Einarbeitungszeit, die für das Verständnis des Aufbaus des Codes erfoderlich ist, ist hoch. Der Code des Systems besteht aus vielen hunderten Objekten, die relativ wenig dokumentiert sind, Zusätzlich wird der Einarbeitungsaufwand durch die sehr fortgeschrittene C++-Konstrukte, die verwendet worden sind, erhöht.

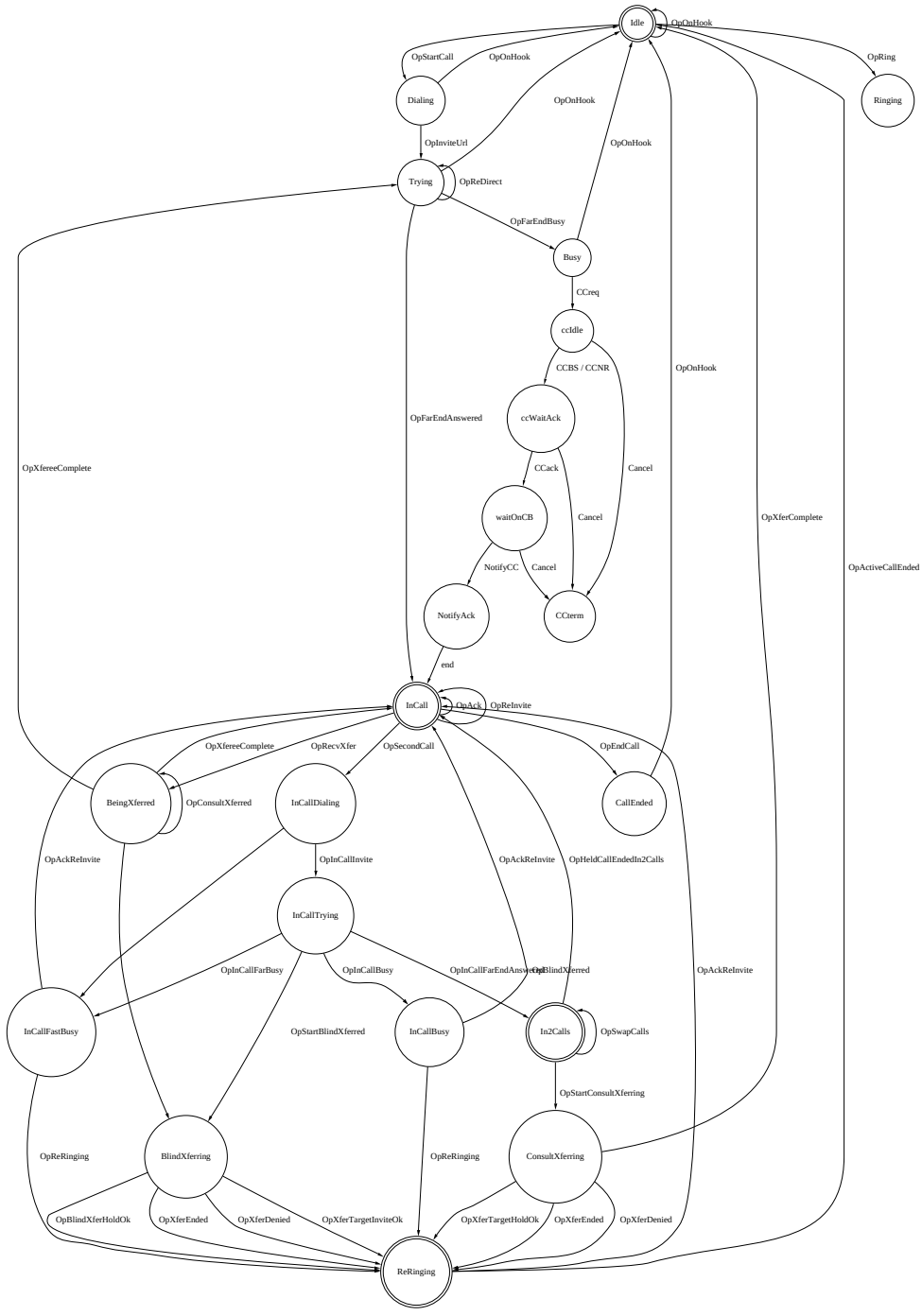


Abbildung F.4: CPLEditor-ContextLookup

Listen

Liste verwendeter Akronyme

AMSA	Advance Multimedia Signaling Architecture
ABNF	Augmented Backus-Naur-Form
AI	Artificial Intelligence
AIN	Advanced Intelligent Network
AMS	Active Map Service
AP	Access Point
API	Application Programming Interface
ASN-1	Abstract Syntax Notation No.1
ATM	Asynchronous Transfer Mode
B2BUA	Back-2-Back User Agent
BCSM	Basic Call State Model
BPEL	Business Process Execution Language
CALL	Context Aggregation Level Language
CAN	Context Aggregation Network
CCXML	Call Control Extensible Markup Language
CEA	Cambridge Event Architecture
CFN	Context Fusion Network
CGI	Common Gateway Interface
CIS	Context Information Service
COA	Center of Area
CORBA	Common Object Request Broker Architecture
CPIM	Common Profile for Instant Messaging
CPP	Common Profile for Presence
CPL	Call Processing Language

CSV	Comma Separated Values
CUS	Custom User Service
DAG	Directed Acyclic Graph
DCE	Distributed Computing Environment
DCOM	Distributed Component Object Model
DFC	Distributed Feature Composition
DHCP	Dynamic Host Configuration Protocol
DHT	Distributed Hash Table
DNF	Disjunctive Normal Form
DOM	Document Object Model
DSL	Digital Subscriber Line
DTD	Document Type Definintion
DTMF	Digital Tone Multi Frequency
EAI	Enterprise Application Integration
EBNF	Extended Backus-Naur-Form
ECA	Event-Condition-Action
EFSM	Extended Finite State Machine
ER	Entity-Relationship-Model
FR	Frame-Relay
FS	Feature Server
FSM	Finite State Machine
GPS	Global Position System
GSM	Global System for Mobile Telecommunication
GUI	Graphical User Interface
GUID	Global Unique Identifier
HCI	Human Computer Interaction
HTTP	HyperText Transport Protocol
HTTP-CGI	HTTP Common Gateway Interface
IANA	Internet Assigned Numbers Authority
iCal	Internet Calendaring
IETF	Internet Engineering Task Force

IMPP	Instant Messaging and Presence Protocol
IN	Intelligent Network
INS	Intentional Naming Scheme
IPC	Interprocess Communication
IR	Infrared
ISDN	Integrated Services Digital Network
iTIP	iCalendar Transport-Independent Interoperability Protocol
ITU	International Telecommunication Union
ITU-T	International Telecommunication Union – Telecommunication Sector
JAIN	JAVA APIs for Integrated Networks
JCC	JAIN Call Control
JCAT	JAIN Coordination and Transaction
JDBC	JAVA Database Connector
JDL	Joint Directors of Laboratories
KI	Künstliche Intelligenz
KSOM	Kohonen Self-Organizing Map
LAN	Local Area Network
LBS	Location Based Services
LCD	Liquid Crystal Display
LDAP	Lightweight Directory Access Protocol
LESS	Language for Endsystem Services
LOTOS	Language Of Ordered Temporal Ordering Specification
MCU	Multiparty Control Unit
MGCP	Media Gateway Cntrol Protocol
MIME	Multipurpose Internet Mail Extensions
MEGACO	MEdia GAteway COntrol
MMUSIC	Multiparty Multimedia Session Control Working Group
MOM	Mean of Maxima
MSC	Message Sequence Chart
NGN	Next Generation Network
OASIS	Organization for the Advancedment of Structured Information Systems

OGC	Open Geospatial Consortium
OMG	Object Management Group
P2P	Peer-to-Peer
PDA	Personal Digital Assistant
PDF	Probability Density Function
PDU	Program Data Unit
PIDF	Presence Information Data Format
PIDF-CE	Presence Information Data Format – Context Enhanced
PKI	Public Key Infrastructure
POTS	Plain Old Telephone Service
PS	Provisioning Server
PSTN	Public Switched Telephone Network
QoS	Quality of Service
QRPC	Queued Remote Procedure Call
RDF	Resource Description Framework
RDO	Relocatable Dynamic Objects
RF	Radio Frequency
RFC	Request For Comments
RMI	Remote Method Invocation
RPC	Remote Procedure Call
RPIDS	Rich Presence Information Data Format for SIP
RTCP	Real-time Transport Control Protocol
RTP	Real-time Transport Protocol
SAMSON	Server Architecture for network independent Multimedia Service cOntrol in heterogenous communication Networks
SAX	Simple API for XML
SIMPLE	SIP for Instant Messaging and Presence Leveraging Extensions
S/MIME	Secure MIME
S/RTP	Secure RTP
SCML	Service Control Markup Language
SCP	Service Control Point

SDL	Specification and Description Language
SDP	Session Description Protocol
SGML	Standard Generalized Markup Language
SER	SIP Express Router
SIB	Service Independent Building Block
SIP	Session Initiation Protocol
SIP-CGI	SIP Common Gateway Interface
SLA	Service Level Agreement
SMS	Short Message Service
SMTP	Simple Mail Transport Protocol
SNR	Signal-to-Noise Ratio
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOM	Self-Organizing Map
SQL	Structured Query Language
SSP	Service Switching Point
STP	Service Transfer Point
TAPI	Telephony Application Programming Interface
TDM	Time Division Multiplex
TEA	Technology for Enabling Awareness
TINA	Telecommunications Information Networking Architecture
TLS	Transport Layer Security
TOF	Time-of-Flight
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
UDDI	Universal Description, Discovery and Integration
UML	Unified Modelling Language
URI	Unified Resource Identifier
URL	Uniform Resource Locators
UTM	Universal Transverse Mercator

VAD	Voice Active Recognition
VOCAL	Vovida Open Communication Application Library
VOVIDA	VOice VIdeo DAta
VS	Virtueller Sensor
W3C	World Wide Web Consortium
WAF	Wall Attenuation Factor
WAN	Wide Area Network
WAP	Wireless Application Protocol
WG	Working Group
WLAN	Wireless Local Area Network
WSDL	Web Service Description Language
WSIL	Web Service Inspection Language
XDR	eXternal Data Representation
XML	eXtensible Markup Language
XSD	XML Schema Definitions
XUA	eXtended User Agent

Liste verwendeter Symbole

Kapitel 2: Dienste

Symbol	Bedeutung
σ	Dienst
$\oplus$	Dienstverknüpfungsoperator
ϕ	Feature
p	Eigenschaft
$\models$	erfüllt
$\mathcal{S}$	System

Kapitel 3: Effiziente Echtzeitkommunikation

Symbol	Bedeutung
η	Effizienzratio
A	Aufwand
N	Anzahl der Versuche
t	Bearbeitungszeit
$t^{\curvearrowright}$	Zeitspanne für Anrufer
$t^{\curvearrowleft}$	Zeitspanne für Angerufenen
γ, Γ	Störfaktor, Gesamtstörfaktor
k, K	Kosten, Gesamtkosten
r, R	Ressource, Gesamtressourcen
B	Bandbreite
C	CPU-Zyklen

Kapitel 4: Kontext

Symbol	Bedeutung
c	Kontext(objekt)
C	Menge aller Kontexte
c'	äußerer Kontext
p	Aussage
G	Gegenstandsmenge
M	Merkmalsmenge
I	Relation zwischen G und M
K	formaler Kontext
υ	Sensorwerte
s	Sensordaten
ζ	Kontextinformation
ξ	Kontext
λ	Kontextbezeichner
ω	Wissen
$\varphi(\cdot)$	Merkmalsextraktion
$\chi(\cdot)$	Charakteristische Funktion
$\otimes$	Kontextverknüpfungsoperator
w	Wichtung
δ	Ursprung
t	Zeit
ε	Umgebung

Kapitel 6: Automatische Kontextgewinnung

Symbol	Bedeutung
S	Sensor
t	Zeit
D	Domäne
X	Daten
x	Elemente
$\chi(\cdot)$	Indikatorfunktion
$\mu(\cdot)$	Zugehörigkeitsfunktion
$\tilde{A}, \tilde{B}$	Fuzzy Mengen

Kapitel 7: Kontextinfrastruktur

Symbol	Bedeutung
$grad(\cdot)$	Grad der Verbindungen
T	Typen
M	Datenmenge

Kapitel 8: Vereinfachte Diensterstellung für Nutzer

Symbol	Bedeutung
r, R	Regel, Menge der Regeln
e, E	Ereignisse, Menge der Ereignisse
c, C	Bedingung, Menge der Bedingungen
A	Aktionen
v	Knoten
t_s, t_e	Start- und Endzeit
ψ, Ψ	Ressource, Ressourcen
E	Entitäten
p	Priorität
$\multimap$	Abbildung der Ressourcentypen

Kapitel 9: Entwickelte Verfahren & Komponenten

Symbol	Bedeutung
σ	Standardabweichung
μ	Erwartungswert
s	Standardabweichung
$\bar{x}$	Mittelwert
ε	maximal erlaubter Fehler
n	Anzahl der Messungen
r, R	Raum, Menge aller Räume
Φ	aktuell gemessener Signalstärken-Fingerprint
$\hat{\Phi}$	Charakteristischer Fingerprint eines Raumes
d, D	Distanz, Distanzmenge
A	Accesspoint
m	mobiler Client

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Liste der Algorithmen

Curriculum Vitae

Persönliche Daten

Name:	Manuel Görtz
geboren am:	19.03.1973 in Frankfurt am Main, Deutschland
Nationalität:	Deutsch

Ausbildung

2005 – 2000	Technische Universität Darmstadt, Darmstadt, Deutschland Fachgebiet Multimedia Kommunikation (KOM) Fachbereich Elektrotechnik & Informationstechnik (ETiT) angestrebter Abschluß: Doktor-Ingenieur (Dr.-Ing.)
1994 – 2000	Technische Universität Darmstadt, Darmstadt, Deutschland Fachbereich Elektrotechnik & Informationstechnik (ETiT) Abschluß: Diplom-Ingenieur (Dipl.-Ing.)
1993 – 1994	Zivildienst
1979 – 1983	August-Henze-Schule (Grundschule), Frankfurt am Main,
1983 – 1992	Musterschule (Gynmasium)Frankfurt am Main Abschluß: Allgemeine Hochschulreife

Auszeichnungen

2000	Beste Diplomarbeit am Lehrstuhl Multimedia Kommunikation

Berufliche Tätigkeiten

08/00 – *heute*	Wissenschaftlicher Mitarbeiter Fachgebiet Multimedia Kommunikation (KOM) Technische Universität Darmstadt
06/00 – *heute*	Senior Entwickler KIMK – Kommunikations- und Informationstechnologie, Medien und Kunst GmbH, Seeheim-Jugenheim

Lehrerfahrungen

2003 – *heute*	Vorlesung ausgewählter Kapitel aus „Kommunikationsnetze II“ Betreuung Praktikum & Projektseminar (jedes Semester) Technische Universität Darmstadt
2000 – *heute*	Betreuung der Prüfung und Klausur „Kommunikationsnetze II“ Technische Universität Darmstadt
2000 – *heute*	Tutorien zu „IP-Telefonie“ httc Vortragsreihe „Multimedia Kommunikation", ISCC 2001, IDMS 2001

Betreute Studien- und Diplomarbeiten

1. Martin Eschinger. *Entwurf und Implementierung der Client/Server-Application SipAdminTool zur Konfiguration von IP-Telefonie-Arbeitsplätzen in Java.* Studienarbeit KOM–S–113, August 2001.
2. Andreas Ehrenfried. *Untersuchung der Abbildung von SDP/ngSDP auf RSVP Reservierungen in IP-Telefonie Szenarien.* Studienarbeit KOM–S–0115, April 2002.
3. Jaume Masvidal. *Enrichted Presence Data Format for Context-aware Computing.* Studienarbeit KOM–S–185. Juni 2004.
4. Nuria Tous. *Context Sharing Methods.* Studienarbeit KOM–S–186. November 2004.
5. Victor Fortes. *Use of Fuzzy Logic for Context Aggregation.* Studienarbeit KOM–S–195. September 2004.
6. Tobias Kuckuck. *Implementierung von und Experimente mit einem WWW-Rekorder.* Diplomarbeit KOM–D–144. Januar 2001.
7. Dirk Kölhoff. *Digitale Aufnahme von Videokonferenzen in H.323-Umgebungen.* Diplomarbeit KOM–D–170. Juni 2002.
8. Albert Cervello. *An extension to the Session Initiation Protocol for Call Intrusion.* Diplomarbeit KOM–D–188. November 2002.
9. Martin Elger. *Dynamic Codecs for Text-Based Protocols. Testing SIP with TTCN-3.* Diplomarbeit KOM–D–191. Juni 2003.
10. Andreas Krieger. *Evaluation von formalen Verifikationstools für IP Telephony.* Diplomarbeit KOM–D–192. März 2003.
11. Robert Jurisch. *Untersuchung von IP-Telefonie Diensten und formalen Methoden für Dienst Interaktions Probleme.* Diplomarbeit KOM–D–193. April 2003.
12. Alejandro Perez. *Design and Implementation of a Measurement-Tool for Location-based Services.* Diplomarbeit KOM–D–203. Oktober 2003.
13. Philipp Korzeniewicz. *Konzeption und Implementierung einer Kontextkomponente für Geo-Informationssysteme mit textuellen und graphischen Anfragen.* Diplomarbeit KOM–D–212. November 2004.
14. Johannes Schmitt. *Extension of CPL for Context-Aware Communication Services.* Diplomarbeit KOM–D–217. September 2004.
15. Jaume Masvidal. *Concepts, Algorithm, and Implementation to Exchange Calendar Information to Ddetermine Free Time Slots for Communication.* Diplomarbeit KOM–D–225. März 2005.