TECHNISCHE
UNIVERSITÄT
DARMSTADT

Anpassungsfähige Kontextbestimmung zur Unterstützung von Kommunikationsdiensten

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

Dissertationsschrift

von

Dipl.-Inform. Johannes Schmitt
geboren am 31. Januar 1979 in Frankfurt am Main

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

Tag der Einreichung: 27. August 2009
Tag der Disputation: 03. Dezember 2009

Darmstadt, 2009
Hochschulkennziffer D17

Zusammenfassung

Der Fortschritt der Telekommunikation war im 20. Jahrhundert hauptsächlich davon geprägt, die Leistungsfähigkeit und Funktionalität monolithischer Kommunikationssysteme weiter zu erhöhen. Die Teilnehmer waren über einige wenige, jedoch wohldefinierte Kommunikationskanäle zu erreichen. Die technische Entwicklung der Telekommunikation zu Beginn des 21. Jahrhunderts hingegen eröffnet eine Vielzahl gleichzeitig verfügbarer und heterogener Kommunikationskanäle (z.B. Festnetztelefonie, Mobiltelefonie, Kurznachrichten, E-Mail). Für den Initiator einer Kommunikation bedeutet dies einen erhöhten Aufwand, um eine möglichst zielführende Form der Kommunikation auszuwählen. Auf der Seite des Empfängers besteht ebenso durch möglicherweise unpassende Kommunikationskanäle oder unerwünschte, nicht der Situation angemessene Kommunikation ein erhöhtes Störungspotential.

Gleichzeitig erlauben heute verfügbare Informationstechnologien die Erfassung von digitalen Informationen, die Menschen, bei der Kommunikation und allgemein im Alltag, umgeben. Eine zielgerichtete Auswertung dieser Informationsmenge, d.h. die Bestimmung des *Kontextes* (Umgebungsparameter) eines Teilnehmers, ermöglicht eine Unterstützung bei der Optimierung der Zielführung von Kommunikationsanfragen. Die Entscheidung, welche Form der Kommunikation gewählt wird, sollte abhängig von der Situation, der Relation zwischen den Teilnehmern und den Fähigkeiten der zur Verfügung stehenden Endgeräte erfolgen und wird in dieser Arbeit thematisiert.

Die grundlegenden Herausforderungen, welche diese Arbeit adressiert, bestehen in der Bereitstellung, Auffindung und Nutzung relevanter und aussagekräftiger Informationsquellen, so dass die gesammelten Informationen eine ausreichende Basis zur Entscheidungsfindung bieten. Hierzu ist die Verwendung semantischer Beschreibungen über die Fähigkeiten und Relationen von Sensoren unumgänglich.

Eine Auswertung der gesammelten Informationen muss adaptiv gegenüber einer dynamischen Informationsgrundlage, sowie gegenüber den Eigenschaften und Wünschen des Teilnehmers erfolgen. Ein zentrales Ziel dieser Arbeit ist eine Erweiterung der bisher vorwiegend manuellen Auswertung im Umgang mit diesen hochdynamischen und heterogenen Informationsstrukturen. Hierzu werden in dieser Arbeit die Rahmenbedingungen, die Architektur und die Umsetzung eines *Kontextdienstes* aufgezeigt. Dieser Kontextdienst ermöglicht es, ein Modell zur maschinell unterstützten Auswertung des Kontextes eines Nutzers anhand des Nutzerverhaltens zu erstellen, anzupassen und anzuwenden. Des Weiteren werden Verfahren zur Modellerstellung für das ausgewählte Szenario, der anpassungsfähigen Verarbeitung von Kommunikationsanfragen, miteinander verglichen und optimiert.

Durch die Implementierung eines prototypischen Systems wird die Umsetzbarkeit des gewählten Ansatzes zur Verarbeitung heterogener und dynamischer Informationsquellen gezeigt. Eine Leistungsbewertung der entwickelten Verfahren zur Informationsauswertung zeigt signifikante Verbesserungen in der erzielbaren Ergebnisqualität und Verarbeitungsgeschwindigkeit gegenüber existierenden Ansätzen.

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die an meiner Promotion beteiligt waren und/oder mich dabei unterstützt haben.

Zunächst möchte ich mich bei meinem Referent und Arbeitgeber Herrn Prof. Dr.-Ing. Ralf Steinmetz sehr herzlich dafür bedanken, dass er all dies möglich gemacht hat. Dank seiner Unterstützung konnte ich meine Ideen im Rahmen von Projekten am Fachgebiet Multimedia Kommunikation umzusetzen. Ebenfalls möchte ich mich herzlich bei Herrn Prof. Dr.-Ing. Jörg Eberspächer für die Übernahme des Zweitgutachtens und die Unterstützung bei der Vorbereitung der Disputation bedanken.

Meine Arbeit wurde im Rahmen einer Reihe von Projekten mit der Firma Siemens-Enterprise durchgeführt und finanziert. Für die gute Zusammenarbeit über viele Jahre hinweg und die Möglichkeit meine Ideen auch im Rahmen dieser Projekte umzusetzen, danke ich besonders Herrn Totzke und Herrn Klug von der Firma Siemens-Enterprise.

Bei meinen Kollegen Parag Mogre, Matthias Kropff, Andreas Reinhard und Farid Zaid aus meiner Forschergruppe Mobile Communication and Sensornetworks möchte ich mich sehr für die Zusammenarbeit, die Unterstützung, die aktive Teilnahme an meinen Probevorträgen und vor allem auch für die Entlastung während der Zeit des Schreibens und der Vorbereitung der Disputation bedanken. Bei Matthias Hollick, der ein Großteil der Zeit mein Gruppenleiter war, bedanke ich mich für die gute Zusammenarbeit, sowie für das zahlreiche Feedback beim Schreiben. Desweiteren möchte ich mich bei meinem ehemaligen Kollegen Manuel Görtz, der damals meine Diplomarbeit betreut, mich beim Fachgebiet eingeführt und letztlich auch meine Dissertation gegengelesen hat, bedanken. Mein Dank geht auch an Christoph Roos, Christian Schäfer, Frank Remetter, Matthias Priebe und Alexander Vitanyi, deren Diplom-, Studien oder Masterarbeiten von mir betreut wurden und die mich bei der Umsetzung vieler Ideen unterstützt haben. Bei meinem (aktuellen und ehemaligen) Kollegen und Kolleginnen am Fachgebiet Multimedia Kommunikation bedanke ich mich für die gute Arbeitsatmosphäre, den Austausch von Ideen und das zahlreiche gute Feedback bei meinen Vorträgen.

Bei meinen Freunden und Bekannten möchte ich für ihre Mithilfe und für ihren Beistand während meines Studiums, meiner Promotion und meinen Prüfungen bedanken. Ich bedanke mich bei meinem Freund Daniel (der mich damals davon überzeugt hat, jeden Tag mit der S-Bahn von Frankfurt nach Darmstadt zu pendeln, um hier an der TU zu studieren) für die gute Zeit und dafür, dass wir das Studium gemeinsam durchgezogen haben.

Besonderer Dank gebührt meiner Frau Sara. Ohne ihre Hilfe wäre diese Arbeit so nicht möglich gewesen. Sie hat mit Ihrer Arbeit mein Studium ermöglicht und sich anschließend, während der Arbeit am Fachgebiet Multimedia Kommunikation, um unsere Kinder gekümmert. Sie hat wohl am häufigsten meine Probevorträgen gehört und mir durch die anstrengenden Zeiten im Studium und am Ende der Promotion geholfen. Ebenso danke ich meinen Eltern, welche mich und meine Familie bei diesem Weg immer unterstützt haben.

Inhaltsverzeichnis

1 Einleitung

Dieses Kapitel beginnt mit der Motivation für diese Arbeit und gibt im Anschluss einen Abriss über die Historie in diesem Bereich. Anschließend werden die Ziele dieser Arbeit skizziert und der Fokus der Arbeit festgelegt. Die Ziele der Arbeit werden schließlich in einzelne Teilaspekte untergliedert. Deren Abfolge und der darin enthaltene eigene Beitrag werden am Ende dieses Kapitels aufgezeigt.

1.1 Motivation

Telekommunikation ist ein wesentlicher Bestandteil unserer Gesellschaft. Bestehende Technologien bieten heute eine Vielzahl von Möglichkeiten zur Telekommunikation. Es ist prinzipiell möglich, immer und überall für jeden erreichbar zu sein.

Problem A: Erreichbarkeit

Neben der *Individualkommunikation*, bei der sich zwei Personen direkt miteinander unterhalten, existiert eine Vielzahl von Formen der Kommunikation, welche durch neue Technologien ermöglicht werden. Diese *technische Kommunikation* umfasst nicht mehr nur Telekommunikation im Sinne der festnetzgebundenen Telefonie. Neben Mobilfunk, SMS, Fax, E-Mail entstehen immer neue Möglichkeiten zur Kommunikation, wie Instant Messaging (IMS), IP-Telefonie (VoIP), Skype, Chat oder Twitter. Das Problem der technischen Kommunikation besteht daher heute nicht mehr darin, die Erreichbarkeit zu erhöhen. Im Gegenteil: Aus der Möglichkeit, immer erreichbar zu sein, erhöht sich die Zahl unerwünschter Anrufe oder nicht zielführender Kommunikationsanfragen. Es geht also vielmehr darum aus der Vielzahl von Möglichkeiten diejenige Form der Kommunikation auszuwählen, die der jeweiligen Situation am angemessensten ist, beziehungsweise die beste Zielführung der Anfrage verspricht. Verfügbare Technologien bieten eine große Menge an Funktionalitäten und Konfigurationsmöglichkeiten. Viele dieser Funktionalitäten können zwar genutzt werden, um die Erreichbarkeit zu optimieren, kommen aber nie oder nur selten zum Einsatz, da die Handhabung zu kompliziert, die Einrichtung zu aufwendig oder das ständige Aktualisieren der Einstellungen zu umständlich ist.

Ansatz A: Kommunikationsdienste

Bisher ist die Verwendung von Kommunikationsdiensten (siehe 2.4.5), welche Anrufe verarbeiten (beispielsweise Anrufweiterleitung) immer verbunden mit einem gewissen Aufwand, um diesen Dienst einzurichten und zu aktivieren. Die Einstellungen solcher Dienste immer auf dem aktuellen beziehungsweise gewünschten Stand zu halten, erzeugt einen zusätzlichen Aufwand. Ein manuelles Eingreifen bei der Steuerung der Dienste erzeugt zusätzlich das Risiko, dass die vorgenommene Einstellung nicht korrekt oder nicht mehr aktuell ist (beispielsweise wenn es versäumt wurde, die Weiterleitung wieder zu deaktivieren).

Problem B: Komplexität und Aufwand

Viele Nutzer entscheiden intuitiv durch Abwägung zwischen dem Mehrwert und dem gefühlten Aufwand, ob sie einen Dienst einsetzen möchten oder nicht. Der erhöhte Aufwand bei dem Einsatz eines Dienstes, führt dazu, dass ein Dienst nicht oder nur selten verwendet wird. Eine Anrufweiterleitung wird beispielsweise oft (wenn überhaupt) nur bei länger andauernden Ereignissen wie Urlaubszeiten aktiviert. Wäre es möglich, die Anrufweiterleitung automatisch einzurichten und zu aktivieren, könnten auch Ereignisse von kurzer Dauer, wie beispielsweise das Verlassen des Büroraumes, Besprechungen oder die Mittagspausen berücksichtigt werden, ohne dass ein Mehraufwand erzeugt wird.

Ansatz B: Reduzierung des Aufwands

Zur Minimierung des Aufwandes ist es daher notwendig, die Konfiguration eines solchen Dienstes zu automatisieren. Das bedeutet der Dienst muss regelmäßig an die aktuelle Situation, in der sich der jeweilige Nutzer befindet angepasst werden. Um dieses Ziel zu erreichen, müssen Informationen herangezogen werden, um die Situation des Nutzers (seinen *Kontext* - siehe Abschnitt 2.4.3) zu beschreiben. Im Falle der Anrufweiterleitung wären demnach

Informationen, welche Aufschluss über die aktuelle Situation des Anzurufenden geben, wie beispielsweise seine aktuelle Position, aktuell verfügbare Kommunikationsendgeräte oder die Relation zum Anrufenden von Vorteil.

Problem C: Nutzung vorhandener Informationen

Wir Menschen sind meist von einer Fülle von Informationen umgeben. Technische Geräte, welche den Nutzer umgeben oder die von ihm verwendet werden, werden immer leistungsfähiger und sind immer häufiger vernetzt. Dies hat zur Folge, dass die Zustände dieser Geräte oder die Informationen, welche auf ihnen gespeichert sind, abgerufen und genutzt werden können. Als Beispiele für solche Geräte sind Mobiltelefone, Hausautomatisierungssysteme, Navigationssysteme oder der Computer des Nutzers zu nennen. Von diesen Geräten können unter anderem Informationen über die aktuelle Lokation des Nutzers, anstehende Termine, vollzogene Tätigkeiten oder die räumliche Nähe und Relationen zu anderen Teilnehmern bezogen werden. Jede dieser Informationen kann, sofern sie verfügbar ist, als Indiz genutzt werden, um Aufschluss über die Situation zu geben, in der der Nutzer sich gerade befindet. Durch das Auslesen und Verarbeiten solcher Informationen ergeben sich Vorteile für die Teilnehmer. Durch eine sinnvolle Auswertung und Nutzung dieser Informationen können dem Nutzer Entscheidungen abgenommen bzw. nach seinen Wünschen durchgeführt werden.

Ansatz C: Automatisierung durch Einbeziehung und Auswertung vorhandener Informationen

Wenn zukünftig auf eine Vielzahl dieser Gerätschaften digital und in Echtzeit zugegriffen werden kann, können daraus eine Reihe neuer *intelligenter* Dienste und Dienstleistungen entstehen, welche abhängig vom Zustand des Nutzers agieren (*kontextberücksichtigende Dienste* – engl. *context aware services*). Die Verwendung von solchen kontextberücksichtigenden Diensten beinhaltet die Erfassung und Auswertung von Informationen, welche den Kontext beschreiben. Durch die Nutzung der Kontexte von Teilnehmern ist es möglich, Dienste zu erstellen, welche sich abhängig vom aktuellen Kontext selbstständig auf die gewünschte Konfiguration umstellen. Es gilt also in einem ersten Schritt, verfügbare und vor allem relevante Informationsquellen zu identifizieren und deren Zustände zur erfassen, beziehungsweise deren bereitgestellte Informationen auszulesen. Diese Informationen werden jedoch von sehr unterschiedlicher Struktur, Genauigkeit, Vollständigkeit und Bedeutung sein. Daher ist es in einem zweiten Schritt notwendig, diese Menge an gesammelten Informationen geeignet auszuwerten, um letztendlich zu einer Entscheidung zu gelangen und eine Aktion durchzuführen. Welche Entscheidung vom Nutzer gewünscht wird, kann jedoch nur vom Nutzer selbst bestimmt werden. Hinzu kommt, dass sich jeder Nutzer möglicherweise in einem anderen Umfeld bewegt oder andere Arten von Gerätschaften nutzt. Daraus folgt, dass jeder Nutzer von unterschiedlichen Mengen von potentiell relevanten (externen) Informationsquellen umgeben ist. Daher ist es notwendig, die Entscheidung an die Anforderungen des jeweiligen Nutzers und die verfügbare Menge von relevanten Informationsquellen anzupassen.

Dieses Szenario und die Rahmenbedingungen, welche sich daraus ergeben und die angedeuteten Ziele werden im Rahmen dieser Arbeit weiter verfeinert und behandelt. Das Ziel ist es, ein Rahmenwerk zu erarbeiten und dessen Umsetzung vorzustellen, welches es ermöglicht, Teilnehmer durch neue, anpassungsfähige und kontextberücksichtigende Dienste zu unterstützen.

1.2 Historie

Die Notwendigkeit, zusätzliche Informationen heranzuziehen, um Dienste zu automatisieren, beziehungsweise deren Arbeitsablauf zu beeinflussen, wird schon in verschiedenen Bereichen eingesetzt (siehe Abschnitt 2.5). Meist werden dort jedoch nur Informationen aus der direkten Umgebung des Dienstes genutzt. In diesen Systemen ist vorab bekannt, welche Informationsquellen verfügbar sind, welche Informationen sie liefern und auf welche Weise diese Informationen interpretiert und verarbeitet werden können. Die Menge an Informationsquellen ist zudem meist statisch, relativ klein, und Effekte, wie beispielsweise Ausfälle von Informationsquellen, kommen nur selten vor. Ein solches Szenario ist für den Nutzer überschaubar. In diesen Systemen ist es daher oft möglich, *Regeln* manuell zu erstellen, um die Auswertung der Informationen zu definieren. Einige Telefonanlagen für Firmen oder VoIP-Software unterstützen beispielsweise schon die Nutzung von Daten aus angegliederten Presence-Diensten oder bieten die Möglichkeit, skriptbasierte Regelwerke zu erstellen, welche einfache Abhängigkeiten zwischen zusätzlichen Informationen und der Anrufverarbeitung definieren. Eine manuelle Erstellung von Regeln kann in einem solchen System sogar von Vorteil sein, da hier der Nutzer direkt bestimmen kann, wie der Entscheidungsprozess ablaufen soll.

Nach der Konfiguration eines solchen Dienstes ist es möglich, ohne weitere manuelle Eingriffe alle gewünschten Verhaltensweisen abzudecken - das bedeutet es wird abhängig vom aktuellem Zustand zwischen verschiedenen Leistungsmerkmalen wie beispielsweise Anrufweiterleitung, Anrufabweisung oder dem Anrufbeantworter gewechselt, ohne dass dazu weitere Eingriffe der Benutzer notwendig sind. Zu diesen Themen wurden im Vorfeld einige Arbeiten am Fachgebiet KOM durchgeführt [Gör05, Sch04].

Das Design der bisher betrachteten Dienste, welche den Kontext berücksichtigen, ist oft einer Reihe von Einschränkungen unterworfen. Diese ergeben sich vor allem durch die Limitierung auf lokal verfügbare oder fest vorgegebene Informationsquellen. Dadurch lassen sich eine Reihe von kritischen Punkten vermeiden. Andererseits begrenzt eine Limitierung auf lokal verfügbare oder fest vorgegebene Informationsquellen stark die Menge der nutzbaren Informationen. Der Mensch ist umgeben von einer wachsenden Menge von vernetzten Geräten. Die Menge der Geräte, welche den Menschen in einer jeweiligen Situation gerade umgeben, beziehungsweise für eine Entscheidungsfindung relevant sind, ist jedoch dynamisch und nicht vorab fest definierbar. Solche Informationen, welche theoretisch nutzbar und ausschlaggebend für eine korrekte Entscheidungsfindung sein könnten, bleiben bei den bisher betrachteten Diensten unbeachtet.

1.3 Ziele

Aus einer Vielzahl von möglichen Formen der Kommunikation, soll eine angemessene Form bestimmt und angewendet werden. Was unter einer angemessenen Form der Kommunikation zu verstehen ist und welche Faktoren dazu berücksichtigt werden müssen, geht aus Abschnitt 2.1.1 sowie Abschnitt 2.3 hervor.

Durch Nutzung des Kontextes sollen Entscheidungsprozesse von Diensten automatisiert und durch eine Anpassung an die jeweilige Situation des Nutzers zur Wahl einer geeigneten Form der Kommunikation führen. Zudem sollen durch eine automatisierte Erfassung und Verarbeitung von Informationen neue, dem Nutzer assistierende Funktionen, ermöglicht werden.

Wie aus Abschnitt 2.3 hervorgeht, existieren Systeme, welche einzelne Aspekte dieser Arbeit erfüllen. Wie ebenfalls in diesem Abschnitt zu lesen ist, erzeugt jedoch das Gesamtszenario, beziehungsweise die Kombination aller daraus resultierenden Anforderungen eine Vielzahl neuer Aufgaben, welche im Rahmen dieser Arbeit erarbeitet werden.

Fokus dieser Arbeit ist es, ein Rahmenwerk zu erarbeiten, welches diese neuen Aspekte und Funktionalitäten zur Erfassung und Nutzung von Kontextinformationen erbringen kann. Eine zentrale Herausforderung der Arbeit besteht darin, dynamisch externe Informationsquellen zur Entscheidungsfindung heranzuziehen, sowie die Entscheidungsfindung an die Wünsche des Nutzers anzupassen. Dieser offene Ansatz führt zu einer Reihe von weiteren Herausforderungen. Ein grundlegendes Anforderungsprofil wird hierzu im Abschnitt 2.3 erarbeitet. Wie in Kapitel 3 genauer erläutert, gliedern sich die Ziele in die Bereiche der Informationsgewinnung und der Informationsauswertung. In den jeweiligen Bereichen werden die in diesem Abschnitt umrissenen Ziele im Detail aufgezeigt und bearbeitet.

Ein weiterer, nicht unwesentlicher Aspekt des Rahmenwerkes ist es, dem Nutzer letztlich einen Mehrwert zu liefern. Um diesen zu liefern, muss der Aufwand, welchen das System erzeugt, geringer sein, als der Nutzen, den der Nutzer daraus zieht. Dieser Aspekt beinhaltet die Notwendigkeit einer vereinfachten Anbindung neuer Informationsquellen, sowie geringe Anforderungen bei der Anpassung der Entscheidungsfindung an die jeweiligen Wünsche eines Nutzers. Externe Informationsquellen führen zu Effekten wie beispielsweise verzögerten Anfragen sowie fehlenden oder fehlerhaften Daten zur Auswertung. Die Informationsgewinnung und die Auswertung der Daten muss jedoch im Rahmen von Qualitätsanforderungen erfüllt werden, wie sie durch die Nutzung innerhalb eines Kommunikationssystems entstehen.

Zur Bearbeitung der Problemstellung müssen Arbeiten auf verschiedenen Ebenen durchgeführt werden und somit auf ein breites Spektrum an Themenbereichen zurückgegriffen werden. Im Gegensatz zu vielen anderen Arbeiten im Bereich der Kontextverarbeitung stehen Aspekte wie *multimodale Kommunikation*, Heimautomation, drahtlose Sensornetzwerke und automatische Nutzung der in der Umgebung verfügbaren Endgeräte nicht im Vordergrund dieser Arbeit. Diese Arbeit legt den Fokus auf die *adaptive Kontextbestimmung zur Steuerung von Kommunikationsdiensten* zur Optimierung der Verarbeitung von Kommunikationsanfragen. Hierzu ist es notwendig sowohl adaptiv gegenüber der Menge an verfügbaren Informationen zu sein, welche den Kontext des Nutzers beschreiben, als auch adaptiv gegenüber den Vorstellungen und Wünschen des Nutzers bei der Auswertung dieser Informationen.

1.4 Struktur und eigener Beitrag

Die Arbeit gliedert sich wie folgt:

Kapitel 1
beschrieb die **Motivation** für diese Arbeit. Es wurde aufgezeigt, dass Kommunikationsdienste das Potential haben, die Kommunikation zu optimieren, jedoch Probleme im Bereich der automatisierten Anpassung an die Situation und Bedürfnisse des Nutzers aufweisen. Aufgrund dieser Problemstellung wurden die Ziele für diese Arbeit umrissen, welche im Wesentlichen eine Nutzung von extern verfügbaren Informationen zur Beschreibung der aktuellen Situation und eine automatische Anpassung der Auswertung vorsehen.

Kapitel 2
beinhaltet die **Grundlagen** für diese Arbeit. Hierzu wird eine Reihe von Anwendungsszenarien erläutert, für welche das Rahmenwerk vielversprechend ist. Die Rahmenbedingungen, welche sich durch die Ziele und die angestrebten Anwendungsfälle ergeben, werden anschließend über eine Problemanalyse erarbeitet. Für eine einheitliche Terminologie innerhalb dieser Arbeit, werden in diesem Kapitel eine Reihe notwendiger Definitionen für Begriffe festgelegt. Abschließend werden am Ende dieses Kapitels Arbeiten aufgezeigt und verglichen, welche ähnliche Themengebiete behandeln.

Kapitel 3
stellt das **Konzept** für das Rahmenwerk vor und erläutert die einzelnen Aspekte. Durch eine Strukturierung der in dem Gesamtkonzept enthaltenen Elemente und der benötigten Funktionalitäten in abstrakte Komponenten, wird eine generische **Architektur** für das Rahmenwerk erarbeitet. Die dabei entstehenden Komponenten sowie deren Interaktionen werden anschließend genauer erläutert. In dem Kapitel wird die Architektur erarbeitet und ein Überblick über die darin enthaltenen Komponenten und deren Funktionen gegeben. Wie aus dem Gesamtkonzept hervorgeht, lässt sich die Arbeit in zwei grundlegende Themenkomplexe aufteilen: Die Informationsgewinnung und die Informationsauswertung. Zu Beginn jedes dieser Themenkomplexe werden die Ziele und Rahmenbedingungen aus dem Gesamtkonzept herangezogen und ausgewertet, um die Anforderungen zu entwickeln, die für die Arbeiten in diesem Bereich gelten. Des Weiteren wird in diesem Kapitel eine Qualitätsbetrachtung durchgeführt.

Kapitel 4
behandelt die Arbeiten zum Thema **Informationsgewinnung**. Eine wesentliche Herausforderung in diesem Bereich liegt in dem offenen Ansatz. So sollen zur Laufzeit auch neue Sensoren mit minimalem Aufwand integriert und genutzt werden können. Zur Umsetzung eines Systems zur Informationsgewinnung durch Informationsquellen mit *Plug-and-Play* Eigenschaften, werden in diesem Kapitel eine Reihe von Technologien aus den Bereichen Netzdienste (Webservices), Dienstauffindung (Service Lookup), Middleware, sowie Methoden zur semantischen Beschreibung von Diensten, Schnittstellen und Informationen herangezogen und auf ihre Anwendbarkeit hin untersucht. Darauf aufbauend wird eine Architektur zur Informationsgewinnung entwickelt. Eine zentrale Bedeutung hat die Funktion des Auffindens von Informationsquellen, welche relevant zur Suchanfrage sind. Mechanismen, welche dies ermöglichen, sowie geeignete Suchstrategien werden erarbeitet.

Kapitel 5
bearbeitet das Thema **Informationsauswertung**. Hier werden Möglichkeiten zur Nutzung der durch die Informationsgewinnung gesammelten Informationen erarbeitet. Hierzu gehört vor allem die Möglichkeit, den Kontext einer Person oder eines Objektes zu bestimmen und diese Bestimmung an die Wünsche und Anforderungen des Nutzers anzupassen. Es wird ein Vorgehen entwickelt, mit dem es möglich ist, das Verfahren zur Bestimmung des Kontextes schrittweise an die Bedürfnisse des Nutzers und die vorhandenen Informationen aus der Umgebung anzupassen. Es wird eine Reihe von Verfahren aufgezeigt, welche sich für diese Aufgabe eignen. Die Verfahren werden anhand ihrer Eigenschaften kategorisiert und analysiert. Hierzu wird ein geeigneter Testaufbau entwickelt und im Bezug auf die Rahmenbedingungen des Anwendungsszenarios hin analysiert. In weiteren Schritten werden geeignete Methoden zur Informationsauswertung herangezogen und erweitert, um sie an die zu Beginn definierten Anforderungen anzupassen.

Kapitel 6

beschreibt, die **Umsetzung des Rahmenwerkes**. Dadurch wird der Beweis erbracht, dass die entwickelte Architektur den gestellten Anforderungen entspricht. Alle Komponenten des Rahmenwerkes wurden dabei implementiert, um ein voll funktionsfähiges System zu erhalten. Das System wurde zudem auch in die Telefonanlage eines namenhaften Herstellers integriert, welcher die Arbeit an dem System unterstützt hat. Auch bei der Umsetzung wurde auf eine gute Wartbarkeit, einfache Erweiterbarkeit und hohe Wiederverwendbarkeit der Komponenten geachtet.

Kapitel 7

gibt zum Abschluss dieser Arbeit wird eine **Zusammenfassung**. In diesem Abschnitt wird außerdem das Thema Datenschutz behandelt. Zudem werden mögliche **Anknüpfungspunkte** für Erweiterungen des Rahmenwerkes aufgezeigt.

Anhang

Hier finden sich detaillierte Beschreibungen zu den verwandten Arbeiten und zu den Basistechnologien im Bereich der Kommunikation zwischen verteilten Diensten. Ebenso finden sich dort weitere Umsetzungsdetails, weiterführende Ergebnisse der Evaluation der Lernalgorithmen, sowie die Akronyme, die in dieser Arbeit mehrfach verwendet werden.

2 Grundlagen

Dieses Kapitel beschreibt die Grundlagen dieser Arbeit. Hierzu werden zunächst die allgemeine Problemstellung und deren Ausprägung in einer Reihe von Anwendungsszenarien skizziert. Anschließend wird aus der Problemstellung und den Anwendungsszenarien ein grundlegendes Anwendungsprofil für das System erarbeitet (siehe Abschnitt 2.3). Eine Terminologie der wichtigsten Begriffe, welche im Rahmen dieser Arbeit herangezogen werden, befindet sich in Abschnitt 2.4. Abschließend werden in Abschnitt 2.5 andere Arbeiten aufgezeigt, welche vergleichbare Problemstellungen behandeln.

2.1 Problemstellung

Nehmen wir ein heute übliches Kommunikationsszenario an: Ein Teilnehmer A möchte mit einem Teilnehmer B (dem *Empfänger*) kommunizieren. Wie in der Motivation angesprochen, hat Teilnehmer A meist eine Vielzahl von Möglichkeiten zur Kommunikation mit Teilnehmer B.

Teilnehmer A wird vermutlich diejenige Variante der Kommunikation wählen, die aus seiner Sicht zur Kommunikation mit Teilnehmer B am angemessensten erscheint (Telefonie, E-Mail, SMS, Fax, Skype, Instant Messaging (IMS), usw.). Die Entscheidung von Teilnehmer A basiert meist auf dem Grund und dem Inhalt seiner Anfrage und der Einschätzung welche Form der Kommunikation in der aktuellen Situation die höchste Zielführung verspricht. Mit dem gewählten Kommunikationsmittel versucht er nun, Teilnehmer B zu kontaktieren. Synchrone Kommunikationsmittel (Telefonie, IMS, Chat, usw.) benachrichtigen (*notifizieren*) den Empfänger beim Versuch des Verbindungsaufbaus, beispielsweise durch das Klingeln am Mobiltelefon. Je nach Situation und den Wünschen des Empfängers kann allerdings diese Notifikation bereits als störend empfunden werden [Ker09]. Bei mehreren möglichen Adressen, unter denen Teilnehmer B zu erreichen ist, erhöht sich zudem der Aufwand für Teilnehmer A, eine Verbindung zu Teilnehmer B herzustellen, da möglicherweise mehrere Adressen nacheinander angefragt werden müssten.

Asynchrone Kommunikationsmittel (E-Mail, SMS, Fax, usw.) erreichen Teilnehmer B erst zu einem Zeitpunkt, an dem er ebenfalls das entsprechende Endgerät/System verwendet. Teilnehmer A bekommt in der Regel nicht mitgeteilt, ob und wann die übermittelte Information vom Kommunikationspartner empfangen wird. Daher können asynchrone Kommunikationsmittel beispielsweise bei Rückfragen zu größeren Verzögerungen führen, bis eine Antwort des Kommunikationspartners vorliegt.

2.1.1 Ziel - Optimale Form der Kommunikation

Ziel ist es, aus einer Vielzahl von möglichen Formen der Kommunikation, die *optimale* Form zu bestimmen und einzusetzen. Die Wahl der optimalen Form ist erstens abhängig von der jeweiligen Situation der Kommunikationspartner und zweitens von deren Vorstellungen und Wünschen.

Für den ersten Teil ist es notwendig, die Situation so umfassend wie möglich zu erfassen, um diese ausreichend beschreiben und in Parameter fassen zu können. Es ist hierbei essenziell, mindestens eine ausschlaggebende Information über die Situation zu erhalten.

Ein Maß für die *optimale* Entscheidung lässt sich jedoch schwer festlegen. Was letztlich die optimale Entscheidung ist, liegt im Empfinden der Teilnehmer selbst. Die Entscheidung ist subjektiv und hängt daher von einer Vielzahl von Faktoren ab, welche sich (bisher) nicht erfassen lassen.

Gerade die Echtzeitkommunikation zwischen Personen benötigt eine Phase, in der beide Teilnehmer ihre Verfügbarkeit und Bereitschaft zur Kommunikation abstimmen, da bereits die Kommunikationsanfrage, etwa das Klingeln des Telefons oder das Eintreffen einer E-Mail, eine Ablenkung des adressierten Teilnehmers verursachen kann. Der Grad der hierbei entstehenden Störung ist unter anderem abhängig von der aktuellen mentalen Arbeitslast, Arbeitstätigkeit, Unterbrechungshistorie und der Persönlichkeit des Empfängers.

Vorbild: Persönlicher Assistent

Heute kann ein persönlicher Assistent am besten die optimale Form der Kommunikation bestimmen. Ein Assistent unterstützt und entlastet seinen Vorgesetzten, indem er Aufgaben wie beispielsweise die Vorverarbeitung oder gar Bearbeitung eingehender Anfragen und Anrufe übernimmt.

Er besitzt weitreichende Kenntnisse über seinen Vorgesetzten, wodurch es dem Assistenten meist möglich ist, die Situation einzuschätzen, in welcher sich sein Vorgesetzter befindet. Zudem kann er auf Informationsquellen, wie beispielsweise den Terminkalender oder das Adressbuch zurückgreifen. Er kann die Beziehung zwischen einem Anrufer und seinem Vorgesetzten einschätzen. Außerdem kennt er die zur Verfügung stehenden Endgeräte und kann zwischen ihnen vermitteln.

Ein Assistent kann aus Erfahrung, sowie durch kurze Interaktionen einschätzen, ob momentan ein günstiger Zeitpunkt ist, ein Gespräch an seinen Vorgesetzten weiterzuleiten oder nicht.

Eine zentrale Eigenschaft des Assistenten ist zudem, dass er die Entscheidungen an die Wünsche seines Vorgesetzten anpassen kann. Hinzu kommen seine Interaktionsmöglichkeiten mit den Kommunikationspartnern. So ist es ihm möglich, Rückfragen durchzuführen, um beispielsweise die Dringlichkeit oder den Inhalt der Anfrage zu ermitteln.

Der Fokus dieser Arbeit ist es, den Teilnehmern eines Kommunikationssystems Funktionen anzubieten, welche sich an den Fähigkeiten eines Assistenten orientieren.

2.2 Anwendungsszenarien

Die folgenden Anwendungsszenarien sollen das bisher abstrakt gehaltene Gesamtkonzept veranschaulichen. Zudem sollen die Ziele der Arbeit durch die Beispiele widergespiegelt und die Rahmenbedingungen für eine Umsetzung des Konzepts verdeutlicht werden.

2.2.1 Situationsabhängige Verarbeitung von Kommunikationsanfragen

Bei der situationsabhängigen Anrufverarbeitung sollen eingehende Anrufe je nach aktueller Situation und Wunsch des Empfängers verarbeitet werden.

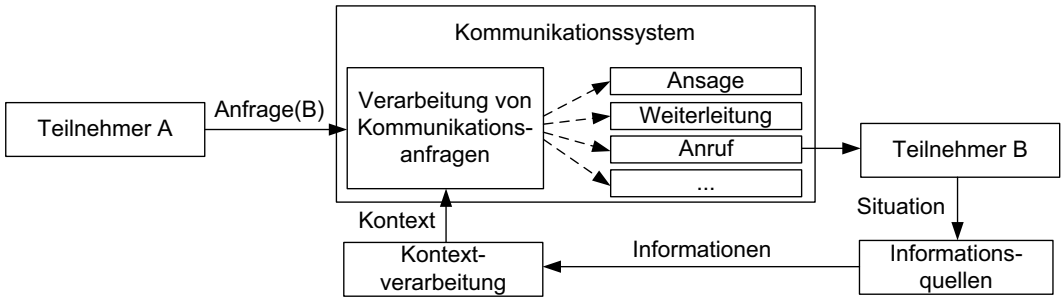


Abbildung 2.1: Situationsabhängige Anrufverarbeitung

Szenario:

Angenommen Teilnehmer A möchte Teilnehmer B erreichen, wie in Abbildung 2.1 dargestellt wurde. Je nach Situation, in welcher sich Teilnehmer B befindet, möchte B wahlweise für Anrufe erreichbar sein oder alle Anrufe auf seine Mailbox umleiten (beispielsweise während Besprechungen, Vorträgen oder Konzerten).

Mehrwert:

Bisher muss Teilnehmer B jeweils manuell sein Mobiltelefon abschalten, lautlos stellen, oder die Anrufe umleiten. Ein Kommunikationsdienst, welcher eine situationsabhängige Anrufverarbeitung anbietet, kann eine solche Situation automatisch erkennen. Durch die Nutzung eines solchen Dienstes wird ein manuelles Umstellen des Mobiltelefons unnötig.

Informationsquellen:

In diesem Szenario müssen Informationen erfasst und ausgewertet werden, welche Aufschluss über die aktuelle Situation von Teilnehmer B geben. Von hoher Relevanz sind beispielsweise der Terminkalender, Sensoren am Teilnehmer (Beschleunigung, Geräuschpegel, Helligkeit, Puls, Nachbarschaft zu anderen Bluetooth-Geräten bzw. zu den Personen, welche sie besitzen), Zugriff auf Lokationssysteme (WiFi, Ultraschall- oder Infrarot-basierte Lokationsbestimmung, GPS). Viele solcher Sensoren sind heute schon auf Mobiltelefonen verfügbar.

Eigenschaften:

Zum Anrufzeitpunkt muss eine Klassifizierung der aktuellen Situation möglichst zeitnah vorliegen, da jede Verzögerung bei der Auswertung zu einer verzögerten Anrufverarbeitung führt. Eine Klassifizierung einer Situation kann jedoch prinzipiell zu jedem Zeitpunkt durchgeführt werden.

Die Situation eines Teilnehmers zu bestimmen ist ein komplexes Unterfangen, welches stark vom einzelnen Teilnehmer abhängig ist. Zum einen unterscheidet sich die Menge der Informationsquellen je nach Benutzer, zum anderen hat jeder Teilnehmer ein anderes Empfinden, was die Interpretation der jeweiligen Situation angeht.

2.2.2 Relationsabhängige Verarbeitung von Kommunikationsanfragen

Bei der relationsabhängigen Anrufverarbeitung sollen Anrufe abhängig von der Beziehung zwischen Anrufer und Angerufenem verarbeitet werden.

Szenario:

Angenommen Teilnehmer B wechselt häufig seine Arbeitsumgebung und verfügt dort jeweils über andere Endgeräte. Er arbeitet zeitweise auch zu Hause und möchte in diesem Fall auch dort auf seinem Telefon erreichbar sein. In solchen Szenarien werden Dienste wie *0700*-Nummern (oder auch *one-number-services*) eingesetzt. Durch Nutzung dieser Dienste kann Teilnehmer B unter einer Adresse mehrere Endgeräte registrieren und ist hierdurch (sowohl im privaten wie auch geschäftlichen Umfeld) für alle Anrufe auf dieser Nummer erreichbar. Um jedoch außerhalb seiner Arbeitszeit keine geschäftlichen Anrufe auf seinem Endgerät zu Hause zu erhalten, muss Teilnehmer B entweder manuell den Dienst umstellen oder über ein weiteres Endgerät zu Hause verfügen.

Mehrwert:

Um zwischen verschiedenen Bereichen beziehungsweise zwischen Beziehungen zu Kontakten zu unterscheiden, werden meist unterschiedliche Adressen benötigt. Häufig werden mehrere Endgeräte eingesetzt, um zwischen verschiedenen Adressen zu unterscheiden. Eine Reduzierung der notwendigen Endgeräte spart Kosten und Aufwand. One-number-services oder statische Anrufumleitungen auf einzelne Endgeräte führen häufig dazu, dass der Teilnehmer auch für unerwünschte Anrufe erreichbar ist. Für diese Problematik ist eine Klassifizierung der Anrufe in unterschiedliche Bereiche (beispielsweise geschäftlich und privat) hilfreich. So kann einerseits die Anzahl notwendiger Endgeräte reduziert werden und andererseits können unerwünschte Anrufe vermieden werden.

Informationsquellen:

Dieses Anwendungsszenario beinhaltet die Erfassung und Verwertung von Informationen, welche Aufschluss über die Beziehung des Anrufers zum Angerufenen bieten. Für eine Bestimmung der Relation zwischen Anrufer und Angerufenem ist es notwendig, zumindest die Anruferkennung bzw. Adresse des Anrufers ermitteln zu können. Unter dieser Voraussetzung können eine Reihe von Informationsquellen für dieses Szenario genutzt werden. Eine sehr aussagekräftige Informationsquelle ist das Adressbuch des Teilnehmers. Zum einen können hier Anrufe nach dem Bekanntheitsgrad eingestuft werden, zum anderen ermöglichen einige Adressbücher auch eine manuelle Klassifizierung des Teilnehmers in bestimmte Gruppen. Hierzu kann theoretisch auch jeweils das Adressbuch des jeweiligen Anrufers genutzt werden. Weitere Aussagen können auch aus Anrufhistorien gewonnen werden, wie beispielsweise: Wie häufig schon mit dem Anrufer kommuniziert wurde und zu welchen Zeiten. Datenbanken und Systeme, welche ein Abbilden von Adressen auf Orte, Personen, Firmen zulassen (unter anderem auch eine Suche über Webservices wie Google oder Social Networks), erlauben zudem die Nutzung weiterer hilfreicher Aussagen über den Anrufer.

Eigenschaften:

Auch in diesem Szenario ist es notwendig, dass bei einem Anruf eine Aussage über die Beziehung zwischen Anrufer und Angerufenem möglichst zeitnah vorliegt. Prinzipiell kann eine Liste der Anrufer geführt werden, mit denen

schon zu einem früheren Zeitpunkt eine Kommunikation stattgefunden hat. Problematisch wird eine Einstufung bisher unbekannter Anrufer, da eine Erfassung aller notwendigen Informationen unter Einhaltung von zeitlichen Einschränkungen nur schwer durchführbar ist.

2.2.3 Endgeräteabhängige Verarbeitung von Kommunikationsanfragen

Bei der endgeräteabhängigen Anrufverarbeitung sollen Anrufe abhängig von den aktuell verfügbaren Endgeräten und deren Fähigkeiten verarbeitet werden.

Szenario:
Teilnehmer A möchte Teilnehmer B eine Nachricht zukommen lassen und ruft Teilnehmer B auf seinem Mobiltelefon an. Das System zur Anrufverarbeitung erkennt, dass Teilnehmer B gerade sein Mobiltelefon abgeschaltet hat, jedoch an seinem Notebook arbeitet. Da festgestellt werden kann, dass das Notebook vom Teilnehmer B und dessen E-Mail-Programm aktiv ist, bietet das System Teilnehmer A an, eine Nachricht aufzuzeichnen und diese Teilnehmer B per E-Mail zukommen zu lassen.

Mehrwert:
Ein solches System zur endgeräteabhängigen Anrufverarbeitung bietet sehr gute Möglichkeiten die Erreichbarkeit des Teilnehmers zu erhöhen und den Nachrichtenfluss zu optimieren. Es geht dabei nicht darum, die Funktion eines Unified Messaging Systems zu bieten, sondern vielmehr darum, den der jeweiligen Situation angemessensten Kommunikationskanal zu bestimmen und diesen zu nutzen. Welcher Kommunikationskanal der angemessenste ist, hängt jedoch stark von der Vorstellung der Kommunikationspartner und der zu transportierenden Informationen ab. Erhöhter Komfort und eine Verringerung nicht zielführender Kommunikationsanfragen wird durch eine Anpassung an die individuellen Wünsche der Teilnehmer erreicht.

Informationsquellen:
Als Grundlage für die richtige Entscheidung, welche Kommunikationsform gewählt werden soll, ist das Wissen um die zur Verfügung stehenden möglichen Endgeräte und deren Fähigkeiten unerlässlich. Hierbei kann ein Endgerät selbst angeben, ob es gerade aktiviert ist und über welche technologischen Fähigkeiten es verfügt. Zudem können manche Geräte Informationen dazu anbieten, ob sie sich gerade in Benutzung befinden (beispielsweise durch Mausbewegungen) oder von dem Teilnehmer transportiert werden (beispielsweise durch einen Beschleunigungssensor). Der Versuch eine Kommunikation zu einem Endgerät aufzubauen, welches aktuell vom Teilnehmer verwendet wird, wird voraussichtlich zum Erfolg führen.

Mit dem Wissen über die Lokation von Teilnehmer B kann zudem festgestellt werden, welche weiteren (stationären) Endgeräte möglicherweise in Frage kommen, um B zu erreichen.

Eigenschaften:
Wenn Teilnehmer A seine Anfrage stellt, steht fest, welche Form der Kommunikation von Teilnehmer A gewählt wurde und daher von diesem bevorzugt wird. Zu diesem Zeitpunkt muss zeitnah bestimmt werden, welche Form der Kommunikation zu Teilnehmer B aufgebaut werden kann und ob diese mit der von Teilnehmer A gewählten Form vereinbar ist. Welche Kommunikationsformen zueinander vereinbar sind, hängt stark von den Fähigkeiten des Systems ab, welche die Anrufverarbeitung durchführt (beispielsweise dem Vorhandensein eines ISDN-SIP Gateways). Letztendlich muss bestimmt werden, ob die vom System gewählte Form der Kommunikation auch dem Wunsch des Anrufers entspricht bzw. von den Fähigkeiten seines Endgerätes geleistet werden kann. Viele der Entscheidungen, beispielsweise welche Form der Kommunikation anderen Formen vorgezogen werden sollte, sind jedoch voraussichtlich bei vielen Teilnehmern ähnlich und daher oft aufeinander übertragbar.

2.2.4 Kombination der Szenarien: Virtueller Assistent

Alle zuvor genannten Szenarien lassen sich miteinander verbinden. Große Teile der jeweils notwendigen Funktionalitäten lassen sich hierbei wiederverwenden, so dass eine Kombination aller zuvor genannten Funktionalitäten zu einem sehr leistungsfähigen Kommunikationsdienst führt. Wenn diese Kombination um weitere bereits etablierte Technologien wie beispielsweise Sprachsynthese, Spracherkennung und Gateway-Funktionen ergänzt wird, ent-

steht daraus ein Kommunikationsdienst, der mit einem *virtuellen Assistenten* vergleichbar ist. Dabei wären folgende Anwendungsszenarien möglich:

Szenario A: Relations-, Situations- und Endgeräteabhängige Anrufverarbeitung
Angenommen Teilnehmer B befindet sich in einer Besprechung und möchte nur noch Anrufe seines Vorgesetzten weitergeleitet bekommen, da er für ihn immer erreichbar sein möchte. Sein Kollege A versucht ihn in dieser Situation telefonisch zu erreichen. Das System zur Anrufverarbeitung erkennt, dass Teilnehmer B sein Notebook nutzt und ein IMS-Programm im Hintergrund läuft. Das System bestimmt, dass die Nutzung des IMS-Programms die der Situation am angemessenste Form der Kommunikation ist, also die höchste Wahrscheinlichkeit bietet, eine Information schnell an Teilnehmer B zu übermitteln und bietet daher dem Kollegen an, per Spracherkennung eine telefonische Nachricht an das IMS-Programm zu senden.

Szenario B: Auskunft
Angenommen Teilnehmer A möchte Teilnehmer B erreichen, Teilnehmer B befindet sich jedoch aktuell in einer Situation, in der er keine Anrufe entgegennehmen kann. Das System kann nun verfügbare Informationen über Teilnehmer B nutzen, um Teilnehmer A Auskunft zu erteilen. Indem Teilnehmer A Auskunft über die aktuelle Situation von Teilnehmer B gegeben wird (welche durch ein System wie in den Abschnitten 2.2.1, 2.2.2 oder 2.2.3 beschrieben ermittelt wird), kann ihm so mitgeteilt werden, wieso sein Anruf nicht durchgestellt werden kann. Ob Auskunft gegeben werden soll und in welchem Detailgrad, kann hierbei von Teilnehmer B von mehreren Faktoren abhängig gemacht werden, wie beispielsweise der aktuellen Situation oder der Beziehung zum Anrufer.

Szenario C: Notifikation
Angenommen Teilnehmer A versucht Teilnehmer B zu kontaktieren. Teilnehmer B befindet sich jedoch aktuell in einer Situation, in welcher er für die gewünschte Form der Kommunikation nicht erreichbar ist. Teilnehmer A möchte darüber informiert werden, wenn sich Teilnehmer B wieder in einer Situation befindet, in welcher er erreichbar ist und kann in einem nachfolgenden Schritt um eine Notifikation beziehungsweise einen Rückruf bitten, sobald es die Situation von Teilnehmer B wieder erlaubt [GAS04].

Szenario D: Erinnerungsfunktion
Angenommen Teilnehmer A hat einen Eintrag mit Ortsangabe in seinem Terminkalender und möchte erinnert werden, wann es notwendig sein wird sich auf den Weg dorthin zu begeben. Das System erkennt über ein Lokationssystem, an welchem Ort sich Teilnehmer A befindet. Befindet sich Teilnehmer A noch nicht an dem angegebenen Ort des Termins meldet sich das System bei dem Teilnehmer, um ihn an den Termin und den Aufbruch dorthin zu erinnern. So rechtzeitig, dass die Dauer der Route zum eingetragen Termin und der noch zur Verfügung stehende Zeitraum ausreichend sind, um pünktlich dort zu erscheinen.

2.3 Grundlegendes Anforderungsprofil

Das in Abschnitt 2.1 abstrakt formulierte Ziel fordert die Bestimmung der jeweils einer Situation angepassten, optimalen Form der Kommunikation. In den darauf folgenden Abschnitten wurden die dafür notwendigen Dienste durch Beispielszenarien beschrieben. Daraus sollen die Aufgaben und Rahmenbedingungen für die Umsetzung dieser Dienste abgeleitet werden. Die grundlegenden Aufgaben die sich durch Betrachtung der Problemstellung und der Beispielszenarien ergeben, werden im folgenden Teil erläutert:

1. **Anpassungsfähigkeit an die Vorstellung und Wünsche der Teilnehmer:** Wie schon in Abschnitt 2.1.1 erläutert, entscheidet letztlich der Teilnehmer selbst, ob die vom System gewählte Form der Kommunikation optimal und angemessen war. Diese Entscheidung des Teilnehmers kann jedoch hoch komplex sein und wird oft subjektiv getroffen. Ein Ansatz die Entscheidung des Teilnehmers zu modellieren besteht darin, Feedback vom Nutzer zu verarbeiten und davon seinen Wunsch im Bezug auf vergleichbare Situationen abzuleiten. Hierdurch ist eine Annäherung der Entscheidungsfunktion an die Wünsche des Teilnehmers möglich.
2. **Schnelle und gute Zielführung einer Anfrage:** Von ebenfalls großer Wichtigkeit für das System ist es, Anfragen erfolgreich zum Ziel zu führen. Das bedeutet, die Absicht mit der der anfragende Teilnehmer die Anfrage absendet, soll unter Berücksichtigung der Situation, in welcher sich der angefragte Teilnehmer befindet, möglichst erfüllt werden. Eine Arbeit, welche sich mit dieser Thematik detailliert befasst, ist unter [Gör05] im

Abschnitt 2.5.2 zu finden. Umgekehrt würden gerade im Bereich der Kommunikation Fehlentscheidungen oder nicht zielführende Anfragen sehr schnell dazu führen, dass sich ein solcher Dienst nicht etabliert.

3. **Umfassende Nutzung verfügbarer Informationen:** Die Grundlage für eine gute Zielführung auf Basis einer automatisierten Entscheidung liegt in der Menge und der Aussagekraft der zur Verfügung stehenden Informationen. Hierzu sollen vor allem bereits zur Verfügung stehende Informationen nutzbar gemacht werden. Informationen, welche sich bereits heutzutage von vernetzen Geräten wie beispielsweise Computern oder Mobiltelefonen auslesen lassen, stehen dabei im Vordergrund.
4. **Hoher Mehrwert, sowie minimaler Konfigurationsaufwand:** Ein Mehrwert ergibt sich nur dann, wenn die Teilnehmer den Eindruck haben, durch die Verwendung des Dienstes deutlich mehr Nutzen als Aufwand zu erhalten. Ein Ziel ist daher, die Wahrnehmung des Dienstes durch die Teilnehmer zu minimieren und Interaktionen mit dem Dienst auf notwendige Elemente zu reduzieren. Auf der einen Seite gilt dies für den Teilnehmer, dessen eingehende Anfragen verarbeitet werden sollen. Hier ist es notwendig, eine einfache Nutzerschnittstelle zur Steuerung des Dienstes zu bieten. In Abschnitt 5.1.1 wird die Problematik bei bestehenden Technologien zur Steuerung und Konfiguration von Kommunikationsdiensten näher erläutert.
   Auf der anderen Seite soll auch für den anfragenden Teilnehmer kein wesentlicher Zusatzaufwand entstehen. Hier gilt es zum einen, keine übermäßigen Verzögerungen bei der Durchführung einer Anfrage beziehungsweise bei der Entscheidungsfindung entstehen zu lassen, sowie den anfragenden Teilnehmer nicht durch unnötige Elemente wie beispielsweise Ansagen oder sprachgesteuerte Menüs zu belasten.
5. **Minimierung der Kosten und des Aufwands zur Umsetzung und Integration:** Ein System, welches hohe Kosten durch zusätzliche Anschaffungen zur Integration erfordert, hat meist deutlich geringere Chancen sich zu etablieren, als ein System, welches sich transparent in bereits vorhandene Komponenten integrieren lässt. Gerade bei der Erfassung von Informationen über die aktuelle Situation durch verteilt existierende Informationsquellen, sollte angestrebt werden, ohne den Kauf zusätzlicher Hardwarekomponenten und mit geringem Aufwand an die Informationen zu gelangen.
6. **Offenes System:** Um möglichst geringe Kosten für die Installation und Administration zu erhalten, soll das System mit möglichst wenigen zusätzlichen Einstellungen und Wartungsarbeiten betrieben werden können. Abläufe sollen weitgehend automatisiert durchgeführt werden. Hierzu gehört auch, dass das Design des Systems offen gegenüber neuen, unbekannten Diensten und Informationsquellen ist. Mit steigender Anzahl von Informationsquellen und darauf zugreifenden Diensten erhöht sich auch die Last, welche von dem System geleistet werden muss. Hier gilt es, beim Design des Systems auf Skalierbarkeit in den entsprechenden Bereichen zu achten.

Diese Anforderungen beschreiben den Fokus dieser Arbeit und lassen sich jeweils noch in eine Vielzahl einzelner, konkreterer Anforderungen unterteilen. Da sich diese Arbeit in einzelne Teilbereiche aufgliedern lässt (wie später in Kapitel 3 erläutert), werden die auf die jeweiligen Teilbereiche zutreffenden Anforderungen dort weiter konkretisiert und behandelt.

2.3.1 Anforderungen

Aus dem Anforderungsprofil von Abschnitt 2.3 lassen sich eine Reihe von Aufgaben und Rahmenbedingungen für den Entwurf eines solchen Systems ableiten. Ziel dieser Arbeit ist es ein Konzept zu erarbeiten und umzusetzen, welches jede einzelne der folgenden Rahmenbedingungen berücksichtigt:

- Integration externer Informationsquellen in den Entscheidungsprozess.
- Heterogenität der Informationsquellen.
- Generischer Ansatz zur Integration.
- Zeitbeschränkte Anwendungsfälle.
- Anpassungsfähiger Entscheidungsprozess.
- Abstraktion von der Komplexität.
- Akzeptanz automatisierter Entscheidungsprozesse.

Diese Rahmenbedingungen des Anforderungsprofils werden im Folgenden detailliert erläutert.

Integration externer Informationsquellen in den Entscheidungsprozess

Die Entscheidung, ob eine Form der Kommunikation der aktuellen Situation angemessen ist, ist stark abhängig von der Situation selbst. Eine Situation lässt sich besser beschreiben, je mehr Informationen über die Situation

vorliegen. Verfügbare Technologien erlauben den Zugriff auf eine Vielzahl externer, verteilter Informationsquellen. Eine Hinzunahme aller verfügbaren Informationsquellen führt meist zu besseren Resultaten, wohingegen ein Verzicht auf die Verwendung dieser Informationen bedeuten würde, potentiell ausschlaggebende Daten zu ignorieren, was dazu führen kann, dass der Kontext möglicherweise nicht korrekt oder nur partiell bestimmt werden kann.

- **Anforderungen durch Hinzunahme externer Informationsquellen:** Mit der Hinzunahme externer Informationsquellen entsteht eine Reihe von Anforderungen an das System, welche von bestehenden Systemen zur Kontextbestimmung meist nicht oder nur teilweise berücksichtigt werden. Externe Informationsquellen haben unter anderem folgende Eigenschaften: Sie sind verteilt, heterogen und dynamisch. Systeme, welche diese Informationsquellen nutzen wollen, müssen diese auffinden, mit unterschiedlichen Architekturen kommunizieren können, sowie vorab unbekannte Typen von Informationsquellen und Informationen mit unbekannter Bedeutung (Semantik) mit in den Ablauf integrieren können.
  Der Entscheidungsprozess über die gesammelten Informationen ist ebenfalls stark davon abhängig, ob externe Informationsquellen genutzt werden sollen oder nicht. Viele Systeme zur Kontextbestimmung führen eine Entscheidung über vorab bekannte Mengen an Informationsquellen durch. Durch diese Einschränkung, können weitaus einfachere Entscheidungsmethoden angewandt werden, wie beispielsweise nutzergenerierte Regelwerke oder Skripte. Die Komplexität dieser, meist schon im kleinen Rahmen für normale Anwender schwierig zu handhabenden, Entscheidungsmethoden steigt rasant mit einer wachsenden Menge von Funktionen und Informationsquellen.
- **Informationsgewinnung bei Hinzunahme externer Informationsquellen:** Bei einer unbeschränkten Menge externer Informationsquellen ergibt sich eine Vielzahl neuer Anforderungen an das System. Mit wachsender Anzahl potentiell verfügbarer Informationsquellen, welche beispielsweise über das Internet erreichbar wären, muss vor allem die Anzahl von Anfragen pro Kontextbestimmung möglichst gering gehalten werden. Da eine Anfrage an alle Informationsquellen zur Kontextbestimmung nicht geeignet ist, muss bestimmt werden können, welche Informationsquellen für eine jeweilige Entscheidung relevant sind. Ein solches System muss daher die Semantik einer Informationsquelle bestimmen und auswerten können. Technologien zur Nutzung semantischer Beschreibungsmerkmale sind vor allem aus dem Bereich der Netzdienste (Webservices) bekannt, werden aber bisher selten in Systemen zur Kontextbestimmung genutzt.
- **Entscheidungsprozess bei Hinzunahme externer Informationsquellen:** Bei einer automatischen Bestimmung aktuell verfügbarer und relevanter Informationsquellen ist es erforderlich, dynamisch wechselnde und relativ große Mengen von Informationsquellen auswerten zu können. Da zudem auch noch davon ausgegangen werden muss, dass jederzeit neue Typen von Informationsquellen hinzukommen können, ist dies mit bestehenden Systemen, welche statische Methoden zur Kontextbestimmung anwenden, nicht möglich.

Heterogenität der Informationsquellen

Einige bestehende Systeme greifen auf speziell für diesen Einsatz konstruierte Komponenten als Informationsquellen zurück. Solche Systeme können darauf optimiert werden, möglichst optimale Leistung für diesen Anwendungsfall zu erbringen. Eine Anschaffung solcher spezieller Komponenten erfordert jedoch zusätzliche Investitionen.

Durch die Möglichkeit, bestehende Hardware, wie beispielsweise Mobiltelefone, Router oder Navigationssysteme, durch Modifikationen an der Software oder Installation zusätzlicher Software, um neue Funktionalitäten zu ergänzen, können diese Technologien einfach in ein solches System zur Kontextbestimmung integriert werden. Daher sollen gerade auch Informationsquellen herangezogen werden können, welche aus Standardkomponenten bestehen und nicht direkt für den Anwendungsfall konzipiert wurden. Es muss auf Informationsquellen unterschiedlichster Art zurückgegriffen werden können, woraus ebenfalls die Anforderung folgt, dass auch stark unterschiedliche Eigenschaften der anzufragenden Systeme, wie Verfügbarkeit und Anfragedauer berücksichtigt werden müssen. Generisch nutzbare Schnittstellen erzeugen zwar mehr Aufwand zur Integration, sichern jedoch langfristig die Erweiterbarkeit des Systems.

Generischer Ansatz zur Integration

Viele Systeme zur Kontextbestimmung sind für einen speziellen Anwendungsfall konzipiert. Das in dieser Arbeit vorgestellte System soll genutzt werden können, um in einer Reihe von Anwendungsfällen genutzt werden zu können. Je nach Anwendungsfall und dem Dienst, welcher von dem kontextberücksichtigenden System erbracht werden soll, werden andere Anforderungen an die Kontextbestimmung gestellt werden. Der Ansatz muss möglichst flexibel sein, um die jeweils gestellten Anforderungen zu erfüllen.

Zeitbeschränkte Anwendungsfälle

Einige der Dienste werden unter strengen Zeitvorgaben durchgeführt. Dies wäre beispielsweise bei einem Dienst zur Steuerung einer Anrufweiterleitung der Fall. In einem zeitkritischen Fall werden innerhalb strenger Zeitvorgaben alle relevanten Informationsquellen zum Anrufzeitpunkt abgefragt und die erhaltenen Informationen ausgewertet. Das Ergebnis muss in jedem Fall vor Ablauf der gesetzten Frist zurückgeliefert werden. Je nach Informationsquelle, beziehungsweise je nach System, welches abgefragt werden soll, dauert eine Abfrage jedoch unterschiedlich lange. Eine Überschreitung der zulässigen Anfragedauer würde dazu führen, dass der Kontext nicht rechtzeitig vorliegt und daher nicht genutzt werden kann oder dass die Suche frühzeitig beendet werden muss und möglicherweise relevante Informationen nicht erfasst und zur Auswertung herangezogen werden können.

Anpassungsfähiger Entscheidungsprozess

Wie zuvor erläutert wurde, ist in System wie dem in dieser Arbeit angestrebten Konzept, eine Auswertung der gesammelten Informationen über statische oder manuell zu konfigurierende Methoden nicht sinnvoll. Das bedeutet, das System muss den Entscheidungsprozess automatisieren, indem es zu jeder Menge von gesammelten Informationen selbständig eine Entscheidung fällt. Wie das gewünschte Ergebnis dieser Entscheidung lautet, hängt jedoch vom Teilnehmer ab, für den das System Entscheidungen trifft. Es ist daher wichtig, dass die Entscheidungsfunktion an die Wünsche des Teilnehmers angepasst werden kann. Je nach Anwendungsfall kann hierfür jedoch auch die Entscheidungsfunktion anderer Teilnehmer wiederverwendet werden. In letzter Instanz sollte jedoch der Teilnehmer selbst die Möglichkeit haben, seine Wünsche an das System in die Entscheidungsfunktion einzubringen. Gerade aus dem Bereich des *Data Mining* kommen einige geeignete Verfahren, die diese Anforderungen erfüllen (siehe Abschnitt 5.3.2).

Abstraktion von der Komplexität

Bestehende Systeme belasten die Teilnehmer oft durch eine Flut von Funktionalitäten, Konfigurationsmöglichkeiten und Informationen. Ein Dienst, welcher im Hintergrund arbeitet, die aktuelle Situation bestimmt, bei Ereignissen scheinbar selbstständig Entscheidungen trifft, sowie den Entscheidungsprozess an den Teilnehmer anpassen kann, kann den Teilnehmer entlasten. Eine automatische Informationsgewinnung und Kontextbestimmung kann vollständig im Hintergrund ablaufen, ohne Eingriffe seitens des Teilnehmers notwendig zu machen. Dadurch können auch hoch komplexe Abläufe und Abhängigkeiten außerhalb der Wahrnehmung der Nutzer verarbeitet werden. Der erbrachte Dienst kann vergleichbar zu einem anpassungsfähigen *virtuellen Assistenten* arbeiten. Neben der wachsenden Komplexität, kann es in vielen Anwendungsfällen sogar gewünscht sein, dass der Nutzer nicht mehr direkt in den Entscheidungsprozess involviert wird. Beispielsweise kann die Abstraktion von den zugrundeliegenden Informationen zum Schutz der Privatsphäre beitragen (siehe Abschnitt 7.1).

Akzeptanz automatisierter Entscheidungsprozesse

Der Teilnehmer ist ein wesentlicher Bestandteil des Gesamtprozesses. Dadurch ergeben sich auf beiden Seiten eine Reihe weiterer Herausforderungen. In dem angestrebten Anwendungsszenario müssen automatisierte Prozesse mit den Vorstellungen und Fähigkeiten von Teilnehmern harmonieren. Eine Arbeit, die diese Herausforderungen diskutiert, wurde von Edwards und Grinter veröffentlicht [EG01].

Durch das Verbergen von Informationsgewinnung und des Entscheidungsprozesses wird ein System weitgehend automatisiert. Heute sind Teilnehmer jedoch noch wenig an solche selbstständig agierende Systeme gewöhnt. Ein System, welches die Entscheidungsfunktion automatisch generiert, liefert eine relativ komplexe Funktion, aus der es meist nur schwer möglich ist, eine Entscheidung nachzuvollziehen. Da häufig die Zustände vieler Informationsquellen miteinander verrechnet werden, ist es ebenfalls nicht möglich konkret zu bestimmen, welche Informationsquellen für eine Entscheidung ausschlaggebend waren. Durch diese Eigenschaften wird das Verhalten eines solchen Systems weitgehend automatisiert, was für die Sicht des Teilnehmers von der Komplexität abstrahiert. Gerade die Anfangsphase ist ausschlaggebend, da in diesem Zeitraum ein Teilnehmer meist entscheidet, ob er ein System weiter nutzen möchte oder nicht.

Bestehende lernfähige Systeme wie beispielsweise E-Mail-Filter nutzen meist initiale Entscheidungsfunktionen, um gerade in der Anfangsphase schon gute Ergebnisse zu erzielen. Auch in einigen anderen Anwendungsfällen können initial definierte Entscheidungsfunktionen in der Anfangsphase angewandt werden. Bei der Verwendung von initialen Modellen ist es jedoch notwendig, dass bei der Entscheidung Informationsquellen verfügbar sind, welche vergleichbar sind mit denen, auf denen das initiale Modell aufbaut. Bei Anwendungsfällen, in denen keine initialen Entscheidungsfunktionen herangezogen werden können (beispielsweise weil die verfügbaren Informationsquellen

vorab nicht bekannt sind), muss es möglich sein auch ohne initiales Modell möglichst schnell zu richtigen Entscheidungen zu gelangen.

2.4 Terminologie

In diesem Abschnitt werden Begriffe eingeführt, welche für diese Arbeit von hoher Relevanz sind. Bei einigen Begriffen geht es zusätzlich darum, ein gemeinsames Verständnis für die Bedeutung der Begriffe herzustellen, da diese in anderen Anwendungsgebieten anders interpretiert werden.

2.4.1 Teilnehmer, Nutzer

Bei dieser Arbeit spielt der Mensch, als Nutzer beziehungsweise Teilnehmer von Kommunikationssystemen, eine zentrale Rolle. Indem es darum geht, das Kommunikationsmanagement an die Wünsche, das Verhalten und die aktuelle Situation der Nutzer anzupassen, wird der Mensch in das Gesamtkonzept mit einbezogen.

Im Fokus dieser Arbeit steht vor allem Kommunikation, welche zwischen zwei Teilnehmern durchgeführt werden soll. Es gibt jedoch unterschiedliche Rollen, welche die Nutzer dabei einnehmen. In den meisten Fällen kann im Vorfeld einer Kommunikation unterschieden werden zwischen folgenden Rollen:

- **Absender:** Der anfragende Nutzer (Initiator), der eine Kommunikationsabsicht hat.
- **Empfänger:** Der angefragte Nutzer, welcher Ziel beziehungsweise Adressat dieser Kommunikationsabsicht ist.

Je nach Rolle haben Nutzer unterschiedliche Sichten und Anforderungen an das System. Wie in Abschnitt 2.1 umschrieben, ist es Ziel dieser Arbeit Dienste zu entwickeln, welche die Funktionen eines Assistenten zum Vorbild haben. Analog zu diesem Szenario müssen bei der Anpassung der Entscheidung an die Wünsche der Nutzer und vor allem die Anforderungen des Empfängers berücksichtigt werden. Daher hat der Empfänger (für die in dieser Arbeit betrachteten Szenarien) als einziger die Möglichkeit, Feedback an das System zu geben, um das Entscheidungsmodell seinen Anforderungen anzupassen.

2.4.2 Kommunikation

Allgemein kann man unter Kommunikation die Übertragung von Informationen zwischen zwei Entitäten verstehen. In dieser Arbeit wird mit Kommunikation jedoch vor allem der Informationsaustausch zwischen Personen beziehungsweise Teilnehmern bezeichnet. Bei den Teilnehmern spricht man von Kommunikationspartnern.

Es gibt heute eine Vielzahl von Möglichkeiten der Kommunikation. Sie lassen sich anhand der folgenden Merkmale unterscheiden:

- **Form:** Dieses Merkmal unterscheidet hauptsächlich sprachliche und textuelle Kommunikation.
- **Übertragungsmedium:** Die genutzte Technologie zur Übertragung der Informationen (beispielsweise das Festnetz (*Public Switched Telephone Network* - PSTN oder *Voice over IP* - VoIP) ist ein weiteres Merkmal.
- **Synchron/Asynchron:** Das Merkmal beschreibt die zeitliche Abfolge des Informationsaustausches zwischen den Teilnehmern. Während Telefonie ein Beispiel für eine synchrone Kommunikation darstellt, ist der SMS-Kurznachrichtendienst ein Beispiel für eine asynchrone Form der Kommunikation.

Kommunikationsanfrage

Eine Kommunikationsanfrage ist im Allgemeinen eine Nachricht, welche der Absender über sein Endgerät an das Kommunikationssystem sendet. Je nach Form und Inhalt der Kommunikationsanfrage lässt sich eine Reihe von Informationen ableiten, welche den *Kontext* der Anfrage beschreiben:

- **Ziel:** Eine (meist eindeutige) Kennung des Empfängers.
- **Absenderkennung:** In vielen Fällen gibt es die Möglichkeit, den Absender zu identifizieren. Gerade bei der Bestimmung der Relation zwischen den Kommunikationspartnern spielt die Identität des Absenders eine entscheidende Rolle. Jedoch lässt sich im Bereich der Telefonie diese Kennung unterdrücken.
- **Form:** Zum Zeitpunkt, in dem die Anfrage gestellt wird, hat sich der Absender für eine Form der Kommunikation (beispielsweise sprachlich oder textuell) entschieden. Allgemein können zwar sprachliche Nachrichten

in textuelle Nachrichten (sowie umgekehrt) umgewandelt werden, dies ist jedoch abhängig von den Möglichkeiten des genutzten Kommunikationssystems.

- **Übertragungsmedium:** Das gewählte Medium des Absenders steht ebenfalls zum Zeitpunkt der Anfrage fest. Allerdings kann oft zwischen verschiedenen Medien vermittelt werden (beispielsweise durch die Weiterleitung einer SMS als E-Mail[1].).
- **Inhalt:** Gerade bei asynchroner Kommunikation steht der Inhalt der Nachricht (der *Anfrage Kontext*, siehe Abschnitt 2.4.3) zum Zeitpunkt der Anfrage fest und kann genutzt werden, um beispielsweise die Dringlichkeit der Nachricht zu bewerten.

Kommunikationssystem

Ein Kommunikationssystem unterstützt mindestens eine Form der Kommunikation unter Nutzung einer Technologie zur Übertragung von Informationen. Aktuell verfügbare Kommunikationssysteme kombinieren meist mehrere Formen und Technologien zur Kommunikation miteinander zu einem *Unified Messaging System* (UMS). Ein solches Kommunikationssystem hat meist die Möglichkeit zwischen den verschiedenen Formen und Übertragungsmedien zu vermitteln. Damit wächst die Zahl der möglichen Varianten der Kommunikation zwischen Personen. Welche Variante der Kommunikation letztlich genutzt werden soll, muss über ein *Kommunikationsmanagement* bestimmt werden.

Kommunikationsmanagement

Das Kommunikationsmanagement kann die vom Kommunikationssystem bereitgestellten Kommunikationsdienste ansteuern. Je nach zu Grunde liegendem Kommunikationssystem entsteht eine Vielzahl von Varianten der Kommunikation. Die Aufgabe des Kommunikationsmanagements ist es daher, aus dieser Menge von Möglichkeiten, eine geeignete Variante zu bestimmen.

Es kann in erster Linie zwischen dem *statischen* und dem *adaptiven* Kommunikationsmanagement unterschieden werden. Das statische Kommunikationsmanagement wendet feste Regeln oder Modelle zur Entscheidungsfindung an. Hierzu gehören unter anderem existierende Ansätze zur Einbindung externer Informationen in die Anrufverarbeitung, sowie die Nutzung speziell für diesen Anwendungsfall erstellter Skriptsprachen, wie sie in Abschnitt 6.5 erläutert werden. Adaptives Kommunikationsmanagement hingegen passt die Modelle zur Entscheidungsfindung an, wie es auch in dieser Arbeit angestrebt wird.

Zudem kann zwischen dem *aktiven* und dem *passiven* Kommunikationsmanagement unterschieden werden. Passives Kommunikationsmanagement bestimmt vor allem den Ablauf einer Kommunikationsanfrage, ausgelöst durch das Eintreffen der Anfrage selbst. Aktives Kommunikationsmanagment erweitert den Funktionsumfang des passiven Kommunikationsmanagements durch die Fähigkeit, selbstständig Aktionen durchzuführen. Hierzu zählen beispielsweise Erinnerungsfunktionen oder das Herstellen einer Verbindung, sobald beide Gesprächspartner sich in einem Zustand befinden, in welchem sie für Anrufe erreichbar sind.

2.4.3 Kontext

Der Begriff Kontext wird aus dem lateinischen Wort *contextus* (Zusammenhang) abgeleitet. Er kann als eine abstrakte Beschreibung des Zustandes, in dem sich eine Entität befindet, umschrieben werden.

In dieser Arbeit wird die folgende formale Definition von Kontext verwendet:

> Es seien zwei Mengen G und M mit reellen Zahlen, Vektoren oder Symbolen gegeben, sowie eine binäre Relation $R \subseteq (G \times M)$ zwischen diesen. Die Menge G bezeichnet dabei Gegenstände, die Menge M Merkmale. Die Relation R hingegen setzt diese beiden Mengen in Zusammenhang und beschreibt das Auftreten von Kombinationen ihrer Elemente – Gegenstände und Merkmale. Ein formaler Kontext ist also durch ein Tripel (G, M, R) repräsentiert [GW97].

In anderen Worten zusammengefasst, ist Kontext ein Zustand, welcher von Objekten eingenommen werden kann. Der Kontext eines Objektes ist variabel und kann abhängig sein von der Zeit, dem Ort, dem Objekt selbst oder beispielsweise den Kontexten anderer Objekte.

Der Begriff Kontext an sich wird oft unterschiedlich genutzt und seine Bedeutung, beziehungsweise die Möglichkeit seiner Darstellung, lässt sich meist nur schwer eingrenzen. In dieser Arbeit wird daher der Begriff Kontext

[1] Diese Fähigkeit hängt maßgeblich von den Fähigkeiten des Kommunikationssystems ab. Es existieren eine Reihe von Arbeiten, welche zum Ziel haben diese Fähigkeiten zu erweitern, wie beispielsweise [Ack03]

weiter differenziert betrachtet. Der Begriff Kontext wird häufig als Synonym für *Kontextinformation* oder *Kontextklasse* genutzt. Daher wird die Bedeutung dieser Begriffe innerhalb dieser Arbeit in den folgenden Abschnitten detailliert aufgezeigt.

Kontextklasse

In dieser Arbeit werden konkrete Zustände benötigt, um letztlich eine Entscheidung zu treffen. Eine Kontextklasse ist daher eine Klassifizierung des Kontextes eines Teilnehmers in einen konkreten Zustand.

Wenn man die zuvor genannte formale Definition des Begriffs Kontext über das Tripel (G, M, R) auf die Verwendung von Kontext in dieser Arbeit anwendet, können mit der Menge G die *Kontextobjekte* (beispielsweise die Teilnehmer) bezeichnet werden. Eine *Kontextbestimmung* bildet die Merkmale M auf eine Kontextklasse K ab. Eine Kontextklasse K erhält zudem eine Bezeichnung, welche möglichst die Bedeutung der Kontextklasse wiedergibt. Beispiele für Kontextklassen wären: *Erreichbar*, *Beschäftigt*, *Nicht verfügbar*. Hierbei gilt zu beachten, dass die Menge der Kontextklassen K vorab nicht beschränkt ist, sich also während der Laufzeit erweitern lassen sollte.

Differenzierung der Kontextklassen

Für die Zuordnung von Kontextklassen zu Kontextobjekten ist es notwendig festzulegen, welche Kontextklassen genutzt werden sollen, beziehungsweise nach welchen Kriterien unterschieden werden soll. Die Menge der möglichen Kontextklassen K, welche sich beispielsweise bei einem Verbindungsaufbau zwischen Teilnehmern ergeben, sind vielseitig. Zur Bestimmung von Merkmalen müssen die dazugehörigen Kontextinformationen jedoch gezielt, also ihrer Bedeutung nach, verarbeitet werden. Daher ist eine weitere Klassifizierung der Kontextklassen notwendig. Die genaue Form der Darstellung von Kontextklassen ist jedoch abhängig vom Anwendungsfall. Das bedeutet, die Menge der Kontextklassen und deren Bezeichner ist in der Regel abhängig von:

- Dem **Kontextobjekt:** Dem Objekt auf das sich der Kontext beziehungsweise das Merkmal bezieht (siehe Abschnitt 3.1.1).
- Der **Kontextdimension:** Eine Klassifizierung der Eigenschaft, welche betrachtet werden soll (siehe Abschnitt 3.1.2).

Kontextobjekt

Mit dem Begriff Kontextobjekt wird das Element bezeichnet, auf den sich ein Kontext bezieht. Beispiele hierfür wären Benutzerkontext (Eigenschaften, Situation des Nutzers), Dienstkontext (Funktion, Zustand des Dienstes), Netzwerkkontext (Art, Auslastung des Netzwerkes) oder Endgerätekontext (Fähigkeiten, Zustände des Endgerätes). Die Menge der Kontextobjekte kann wiederum in Objektklassen O kategorisiert werden. Eine Objektklasse O vereint Mengen von Kontextobjekten, welche zur gleichen Art gehören oder vergleichbare Rollen aufweisen. Objektklassen können beispielsweise Personen von Gegenständen unterscheiden. Je nach Anwendungsbereich werden andere Objektklassen aufgestellt und genutzt. Einige verwandte Projekte behandeln die Bestimmung von Kontexten gänzlich anderer Objektklassen. Somit kann sich ein Kontext ebenso auf einen Ort, auf einen Gegenstand oder ein virtuelles Objekt, wie beispielsweise eine Internetseite oder einen Avatar [BHRS09], beziehen.

Kontextdimension

Der Kontext eines Objektes ist in der Regel sehr vielfältig. Je nach Anwendungsfall wird daher zumeist nur ein bestimmter Aspekt des Kontextes betrachtet [BGS99]. Der Begriff Kontextdimension beschreibt einen solchen Teil-Aspekt des Kontextes.

Während die Einteilung in Kontextobjekte unterscheidet, *für wen* der Kontext ermittelt wird, wird mit der Einteilung in Kontextdimensionen unterschieden, *für was* der Kontext genutzt werden soll. Anders ausgedrückt bezeichnet die Kontextdimension, welche Eigenschaft des Objektes betrachtet werden soll. In einer Kontextdimension $d \in D$ werden Kontextklassen K, welche Aussagen bezüglich einer bestimmten Eigenschaft beinhalten können, gruppiert ($d \subseteq K$). Eine Kontextdimension kann beispielsweise die Position, der Zustand oder das Umfeld eines Objektes sein. Genauso vielseitig, wie die Eigenschaften eines Objektes sein können, können auch die Dimensionen gestaltet sein. Welche Kontextdimensionen für diese Arbeit in Betracht gezogen wurden, wird in Abschnitt 3.1.2 beschrieben.

In dieser Arbeit wird als Annahme vorausgesetzt, dass es möglich ist, die Kontextdimensionen derart aufzuspannen, dass innerhalb einer Kontextdimension die Kontextklassen disjunkt sind, also ein Kontextobjekt innerhalb einer Kontextdimension zu einem Zeitpunkt immer genau einer Kontextklasse zugeordnet werden kann.

Kontextinformation

Zur Unterscheidung der Begriffe Kontext und Kontextinformationen wird die folgende Definition des Begriffs Kontextinformation herangezogen:

> „Context[-information] is any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and the application" (aus der Arbeit von [Dey00] angepasst durch [Gör05]).

Zusammengefasst sind Kontextinformationen diejenigen Informationen, welche genutzt werden, um den Kontext zu bestimmen. Ein einzelner Wert v eines Sensors s, beispielsweise die Temperatur, besitzt für den Betrachter wenig Bedeutung, so lange er nicht weiß, in welchem Zusammenhang dieser Wert steht und wie er zu interpretieren ist. Erst durch die Hinzunahme oder logische Verknüpfung mit dem Ort und der Zeit an dem die Temperaturmessung stattfand, erlangt dieses Datum einen Sinn. Eine Information steht also in Relation mit der Entität von dem diese Information bezogen wurde. Eine Erweiterung des eigentlichen Wertes, um weitere Informationen bezüglich des Zusammenhangs mit der dazugehörigen Entität, wird als Kontextinformation i bezeichnet. Somit ergibt sich der Zusammenhang $i = (s, v)$. Diese Erweiterung des Sensors mit Zusatzinformationen erfolgt im Allgemeinen durch eine Sensorbeschreibung (siehe Abschnitt 2.4.7).

Kontextbestimmung

Bei der Kontextbestimmung wird der aktuelle Kontext eines Objektes ausgewertet. Eine Kontextbestimmung wird jeweils über ein Kontextobjekt $o \in O$ für eine bestimmte Kontextdimension $d \in D$ durchgeführt. Das bedeutet, aus der jeweiligen Kontextdimension wird die Menge der Kontextklassen bestimmt, welche zum aktuellen Zeitpunkt t auf das Kontextobjekt zutreffen $K_{o,d,t}$. Dazu werden Kontextinformationen $i \in I$ herangezogen, welche für die Kontextbestimmung relevant sind. Die Relevanz der Kontextinformationen für die Entscheidung lässt sich unter anderem durch die Kontextdimension und das betrachtete Kontextobjekt eingrenzen $I_{d,o}$. Es können jedoch weitere Kontextinformationen relevant sein, welche sich nicht direkt auf das Objekt oder die Dimension beziehen lassen. So können beispielsweise Sensoren entscheidende Informationen liefern, welche sich zu dem Zeitpunkt am selben Ort, wie das betrachtete Objekt befinden. Diese Fragestellung wird eingehend in Kapitel 4 dieser Arbeit behandelt. Bei der Kontextbestimmung werden die betrachteten Kontextinformationen schließlich über eine Entscheidungsfunktion f ausgewertet (siehe auch Abschnitt 2.4.4). Damit lässt sich eine Kontextbestimmung wie folgt beschreiben: $K_{o,d,t} = f(I_{o,d,t})$.

2.4.4 Entscheidungsfunktion, Entscheidungsmodell, Nutzerkonzept

Bei der Kontextbestimmung wird eine Entscheidungsfunktion f genutzt, um aus einer Menge von Kontextinformationen I den Kontext K zu bestimmen $K = f(I)$. Welche Methode als Entscheidungsfunktion f herangezogen wird, ist jedoch sehr unterschiedlich. Da diese Methoden im Allgemeinen statt mathematischer Formeln eher komplexe Abläufe zur Auswertung beschreiben, spricht man oft von *Modellen* oder auch von *Entscheidungsmodellen*. In den in Abschnitt 2.5 betrachteten Arbeiten kommt eine Reihe von unterschiedlichen Ansätzen zur Kontextbestimmung zum Einsatz. Die Methoden lassen sich in folgende Kategorien einteilen:

- **Statische Modelle:** Diese oft auf einfachen Skripten oder Regeln basierenden Modelle stellen die einfachste Form von Modellen dar.
- **Ontologiebasiertes Schließen (engl. Reasoning):** Ontologien erlauben es, Beziehungen zwischen Objekten herzustellen, so auch zwischen Kontexten und Kontextinformationen. Über einen sogenannten *Reasoner* werden die gesammelten Kontextinformationen auf die Ontologie angewendet. Dieser Reasoner wendet die Aussagen der Ontologie an, um die gültigen Kontexte zu bestimmen. In der Regel werden vordefinierte Ontologien verwendet.
- **Adaptive Modelle:** Diese Form von Entscheidungsmodell wird über einen weiteren Prozess generiert. Dieser Prozess zum Erstellen von Modellen passt das Entscheidungsmodell an neu gewonnene Aussagen an und wird daher meist als Modellgenerierung (oder *Learner*) bezeichnet.

Die Eigenschaften und Fähigkeiten von Verfahren aus den einzelnen Kategorien, sowie Beispiele und verwandte Arbeiten dazu finden sich in Abschnitt 5.3.2.

Nutzerkonzept

Das Konzept eines Nutzers ist vergleichbar zu einer unbekannten Funktion h (engl. *hidden concept*), welche die Abhängigkeiten zwischen Situation I und dem gewünschtem Ergebnis K beschreibt:

$$K = h(I)$$

Ziel ist es, ein Modell $f()$ zu erhalten, welches das Nutzerkonzept, beispielsweise durch Regelwerke, beschreibt. Im Idealfall entspricht das Modell dem Nutzerkonzept $f(I) = h(I)$. Das Konzept des Nutzers kann jedoch nur soweit erfasst und modelliert werden, wie Informationen darüber vorliegen. Daher ist ein Modell in der Regel nur eine Annäherung an das vorliegende Nutzerkonzept.

2.4.5 Dienst

Der Begriff Dienst kann im Allgemeinen wie folgt definiert werden:

> „Ein Dienst ist definiert als ein sinnvoller Satz an Fähigkeiten, der durch ein existierendes oder gedachtes System für alle, die es nutzen wollen und dürfen, bereitgestellt wird" (angepasst von [Gör05]).

Je nach Einsatzgebiet existieren Dienste mit speziellen Fähigkeiten. Für die Informationsgewinnung und das Kommunikationssystem kommen die im Folgenden vorgestellten Formen von Diensten zum Einsatz.

Web Services

Dienste aus dem Bereich der Web Services haben im Allgemeinen die besondere Eigenschaft, ihre Funktionalitäten für andere Systeme bereitzustellen. Der Zugang zu diesen Diensten erfolgt über ein Netzwerk. Je nach Umsetzung der Zugriffsbeschränkungen kann somit ein Dienst beispielsweise über ein lokales Netzwerk (LAN) oder weltweit über das Internet verfügbar gemacht werden.

Kommunikationsdienst

Diese besondere Form von Diensten sind Bestandteile von Kommunikationssystemen, deren Funktionalitäten sich meist auf die Kommunikation selbst beziehen.

Man unterscheidet zwischen einer Reihe verschiedener Diensttypen. Abbildung 2.2 zeigt einen Ausschnitt einer Dienst-Taxonomie, mit dem Schwerpunkt auf IP-Telefonie. Zu den bekanntesten Kommunikationsdiensten zählen Telefonie, E-Mail, Fax und Instant Messaging.

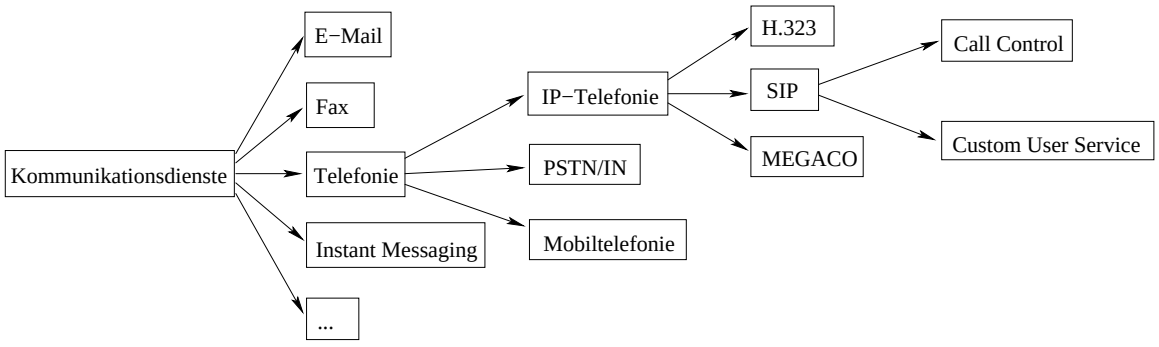


Abbildung 2.2: Taxonomie von Kommunikationsdiensten [Fro97]

Besondere Aufmerksamkeit gilt der Trennung zwischen den Rufkontrolldiensten(engl. *Call Control Services*) und den nutzergenerierten Diensten (engl. *Custom User Service*). Die Rufkontrolldienste bleiben dem Anwender eines Kommunikationssystems meistens verborgen. Hierunter fallen Übertragungsdienste, wie beispielsweise Übermittlungsprotokolle und Steuerungsdienste zur Regelung und Kontrolle der Übertragung bzw. des Kommunikationssystems. Die Nutzung eines solchen Dienstes erfolgt in der Regel implizit bei dem Aufbau einer Verbindung.

Vom Anwender hingegen konkret nutzbare und abstraktere Funktionalitäten werden von nutzergenerierten Kommunikationsdiensten *Custom User Service* zur Verfügung gestellt. Der Nutzer kann durch Nutzung solcher Dienste eigene Dienstfunktionalität realisieren.

In dieser Darstellung werden die Custom User Services und Call Control nur auf SIP bezogen. Dies beruht auf der Tatsache, dass bisher hauptsächlich in SIP Custom User Services möglich waren. Betrachtete man aktuelle Kommunikationssysteme, welche ebenfalls die Steuerung anderer Kommunikationsformen ermöglichen, so können diese auch durch Custom User Services ergänzt werden.

Rufkontrolldienst

Im Bereich Steuerungsdienste existieren neben bestimmten Basisdiensten, wie beispielsweise bei den Telefoniesystemen der Verbindungsauf- und Abbau, je nach System noch unterschiedliche Rufkontrolldienste: Beim ISDN-Dienst sind das etwa folgende:

- Übermittlung, Anzeige und Unterdrückung der Rufnummer
- Rückfrage und Makeln
- Anklopfen
- Dreierkonferenz
- Rückruf bei „besetzt"
- Anrufweiterschaltung

Es gibt nur eine kleine Zahl von Expertengruppen, wie die *International Telecommunication Union* (ITU)[2] oder die *European Telecommunications Standards Institute* (ETSI)[3], die am Entwurf für Standards innerhalb der Rufkontrolldienste für das traditionelle Telefonnetz arbeiten. Gerade im Zusammenspiel von anbieterübergreifenden Funktionen müssen Standards eingesetzt werden, welche eine korrekte Funktionalität, Fehlerfreiheit und die Kompatibilität zwischen verschiedenen Anbietern sicherstellen. Rufkontrolldienste bieten (beispielsweise durch Anrufweiterschaltung) eine grundlegende Möglichkeit, Anrufe zu verarbeiten, können jedoch meist nur manuell aktiviert, beziehungsweise deaktiviert werden.

Nutzergenerierter Kommunikationsdienst

Nutzergenerierte Dienste können vom Nutzer erstellt und modifiziert werden. Solche Dienste sind zumeist für den Einsatz innerhalb eines Systems vorgesehen und bieten dem Teilnehmer die Möglichkeit, verstärkt Einfluss auf die individuelle Verarbeitung von eingehenden Anrufen zu nehmen. In vielen Ansätzen kann der Nutzer durch die Erstellung von Regeln oder Skripten die Verarbeitung eingehender Anrufe selbst bestimmen. Dadurch übernimmt der Kommunikationsdienst Aufgaben aus dem Bereich des Kommunikationsmanagements. Dabei kann er meist nur wenige anrufbezogene Kontextinformationen nutzen (beispielsweise dem Zeitpunkt oder die Absenderkennung) und auf wenige Funktionalitäten des Kommunikationssystems zurückgreifen (beispielsweise Ansagetexte, Anrufweiterleitung, Anrufbeantworter oder E-Mail Benachrichtigung).

Im Bereich der Internet-Telefonie gibt es eine Vielzahl solcher Ansätze für die Erstellung und Bereitstellung von Kommunikations- beziehungsweise Mehrwertdiensten. Schulzrinne beschäftigt sich in [Sch05] mit den Möglichkeiten der Internet-Telefonie und geht dabei auch auf verschiedene Möglichkeiten der Erstellung von Kommunikationsdiensten durch den Nutzer selbst ein.

Kontextbasierter Kommunikationsdienst

Diese Form von Kommunikationsdienst verwendet Kontextinformationen zur Beeinflussung der Art und Weise der Verarbeitung von Anfragen. Abbildung 2.3 zeigt eine schematische Darstellung eines solchen Kommunikationsdienstes.

Ein solcher kontextbasierter Kommunikationsdienst arbeitet vergleichbar mit herkömmlichen Kommunikationsdiensten, mit dem Unterschied, dass Kontextdaten als Parameter in den Prozess integriert werden können. Diese Kontextdaten können einzelne Parameter sein, welche direkt von den Informationsquellen erfasst werden oder auch höherwertige Aussagen, welche durch eine Synthese der gesammelten Informationen gewonnen werden.

Adaptiver (kontextbasierter) Kommunikationsdienst

Diese Form eines Kommunikationsdienstes stellt eine Erweiterung des kontextbasierten Kommunikationsdienstes dar. Während einfache kontextbasierte Kommunikationsdienste eine manuelle und dabei möglicherweise komplexe Konfiguration ihrer Verhaltensweise benötigen, haben diese adaptiven kontextbasierten Kommunikationsdienste die Fähigkeit ihre Verhaltensweise dem Wunsch beziehungsweise dem Verhalten des Nutzers automatisch anzupas-

[2] http://www.itu.int/
[3] http://www.etsi.org/

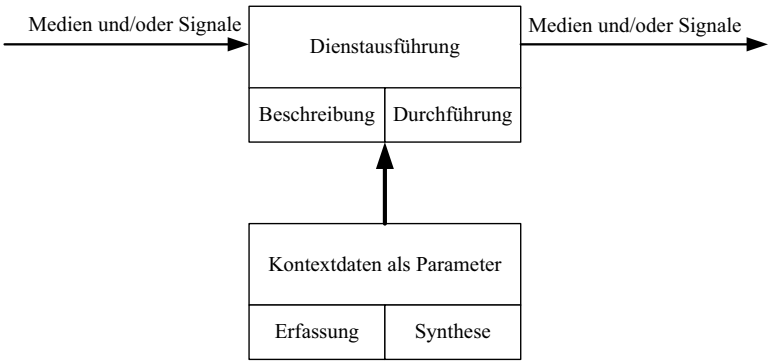


Abbildung 2.3: Kontextbasierter Kommunikationsdienst [Gör05]

sen. Das Design und die Umsetzung eines adaptiven Kommunikationsdienstes ist ein Schwerpunkt dieser Arbeit und wird in Kapitel 5 detailliert behandelt.

2.4.6 Feedback

Feedback wird erst im Zusammenhang mit dem vorher genannten adaptiven Kommunikationsdienst relevant. Mit Feedback wird die Antwort beziehungsweise die Reaktion des Nutzers auf eine Aktion des adaptiven Kommunikationsdienstes bezeichnet. Die genaue Gestalt des Feedbacks ist stark vom Anwendungsfall abhängig und wird in Kapitel 3.2.4 genauer erklärt.

2.4.7 Informationsquelle, Sensor, Supplier

In dieser Arbeit werden Informationsquellen im Allgemeinen als Sensoren bezeichnet. In vielen Anwendungsgebieten wird unter einem Sensor ein physikalisches Messinstrument verstanden, welches die gemessenen Informationen auf elektronischem Wege zur Verfügung stellt. Thermometer, Hygrometer oder Chronometer sind Beispiele für Sensoren im eigentlichen – physikalischen – Sinne. Sensoren treten jedoch nicht nur in physischer, sondern auch in logischer Form in Erscheinung. Die Erfassung des Nutzerverhaltens an einem Computer ist zum Beispiel durch logische Sensoren realisierbar.

In dieser Arbeit wird eine abstrakte Definition für Sensoren angewendet:

> „Ein Sensor ist eine Informationsquelle, welche Werte von Umgebungsattributen auf quantitative Größen abbildet. Infolge dessen ist die Aufgabe eines Sensors nicht alleine durch die Datenerfassung gegeben, sondern beinhaltet ebenfalls eine Abbildung bzw. Funktion auf eine übermittelbare und interpretierbare Größe" (angepasst von [Gör05]).

Daher können neben traditionellen physikalischen Sensoren auch Programme zur Verarbeitung und Bereitstellung von digital verfügbaren Informationen als Sensoren bezeichnet werden. In manchen Arbeiten wird in diesem Zusammenhang von einem *Supplier* gesprochen. Das Thema Sensoren wird in Abschnitt 4.3 detailliert behandelt.

Sensortyp

Sensoren mit gleicher Funktionalität können demselben Sensortyp zugeordnet werden. Sensoren eines Sensortyps besitzen dieselbe Sensorbeschreibung, beziehungsweise bieten die gleiche Schnittstelle an.

Sensornetzwerke

Viele Arbeiten bezeichnen eine Menge von drahtlos und/oder ad hoc kommunizierenden Systemen als Sensornetzwerk. In diesen Arbeiten wird zudem meist die Annahme getroffen, dass das Sensornetz homogen ist, was bedeutet, dass die Menge der Sensortypen (oft auf einen oder zwei Typen - beispielsweise normaler Sensornetzknoten und Datensenke) beschränkt und vorab bekannt ist. In dieser Arbeit wird eine allgemeinere Definition für Sensornetze genutzt:

> Ein Sensornetz ist eine Menge von Sensoren, welche die erfassten Informationen über ein Netzwerk übermitteln beziehungsweise abfragbar machen.

Unter diese Definition fällt jede Art von klassischen drahtlosen Sensornetzwerken, als auch Sensoren in Form von Softwarekomponenten auf fest installierten Rechnern der Teilnehmer oder deren mobilen Endgeräten. Die Infrastruktur, welche in dieser Arbeit herangezogen wird, um Kontextinformationen zu erfassen, ist ein heterogenes, dynamisches Sensornetz, in dem sowohl die Menge der verfügbaren Sensoren als auch deren Sensortypen variabel sind.

Sensordaten

Die Messwerte beziehungsweise Daten, welche vom Sensor bezogen werden können, werden als Sensordaten bezeichnet. Sensordaten liegen als Rohdaten vor, das bedeutet ohne jegliche Informationen bezüglich der Bedeutung der Daten oder des Sensors. Zur späteren Verarbeitung, werden Sensordaten in Form einzelner Werte benötigt, wie beispielsweise Textbausteine (Strings) oder Zahlen (Fließkomma- oder Ganzzahlwerte). Komplexe Sensordaten werden daher, entweder direkt durch den Sensor selbst oder bei der Sensorvorverarbeitung in einzelne Werte aufgeteilt, umgewandelt oder interpretiert. Damit ein Sensordatum seiner Bedeutung nach verarbeitet werden kann, werden Sensordaten mit Hilfe einer Beschreibung der Daten erweitert. Aus der Kombination von Sensordaten mit Informationen aus der Sensorbeschreibung lassen sich Wertepaare (engl. *Key-Value Pairs*) bilden, welche ein Sensordatum einer (möglichst eindeutigen) Bedeutung zuordnet.

Sensorbeschreibung

Eine Sensorbeschreibung liefert sogenannte Meta-Informationen über den Sensor. Meta-Informationen sind notwendig, sobald die Menge der Sensortypen vorab unbekannt oder unbeschränkt ist. Über die Sensorbeschreibung wird bestimmt, ob ein Sensor relevant für eine Anfrage ist, und falls zutreffend, wie seine Informationen abgerufen werden können. Eine Sensorbeschreibung beinhaltet daher zum einen eine Beschreibung der Bedeutung des Sensors und zum anderen Informationen über die Schnittstellen zum Abfragen oder Ansprechen des Sensors. Sensorbeschreibungen lassen sich vergleichen mit der Beschreibung von Netzdiensten.

Semantische Sensorbeschreibung

Semantik beschreibt im Allgemeinen den Sinn oder die Bedeutung eines Elements. Bezogen auf Sensorbeschreibungen sind durch semantische Beschreibungselemente weitere Zusatzinformationen über die Eigenschaften eines Sensors möglich. Das Wissen der Person, die den Sensor installiert, über die Bedeutung des Sensors kann so dem Sensor zugeordnet und genutzt werden. So kann beispielsweise die Angabe, in welcher Weise die vom Sensor gelieferten Daten zu interpretieren sind und für welche Anfragen ein Sensor relevant ist, in einer für Computer lesbaren Form hinterlegt werden.

Semantik wird meist in Kombination mit Ontologien angewendet. Ontologien ermöglichen es Beziehungen zwischen Objekten auszudrücken. Semantische Beschreibungen verweisen daher meist auf Elemente innerhalb einer Ontologie. Werden semantische Beschreibungen von Objekten mit Ontologien kombiniert, können Aussagen über die Beziehungen zwischen den Objekten selbst getroffen werden.

Sensordaten Vorverarbeitung (engl. *Preprocessing*)

Vor der Auswertung der gesammelten Sensordaten kann eine Vorverarbeitung der Sensordaten durchgeführt werden. Es gibt mehrere Gründe aus welchen eine Vorverarbeitung der Sensordaten durchgeführt wird oder sogar notwendig ist.

- **Fehlerreduktion:** In der Vorverarbeitung können verschiedene Verfahren angewendet werden, um fehlende oder fehlerhafte Daten zu erkennen und geeignet zu behandeln (wird in Abschnitt 5.3.1 behandelt).
- **Format-Umwandlung:** Daten gleicher Bedeutung können in unterschiedlichen Formaten vorliegen. Eine Umwandlung in ein einheitliches Format macht diese Daten vergleichbar (beispielsweise Celsius und Fahrenheit bei Temperatursensoren). Je nach Auswertungsverfahren müssen die Daten in einer geeigneten Repräsentation vorliegen. Hierzu ist es beispielsweise notwendig die Daten zu diskretisieren oder zu normalisieren.
- **Merkmalsextraktion (engl. *Feature Extraction*):** Wenn die Bedeutung oder Aussage eines Sensors bekannt ist, lassen sich oft zusätzliche Informationen aus den Rohdaten herausziehen. Durch Merkmalsextraktion werden aus dem eigentlichen Sensorwert weitere, möglicherweise aussagekräftigere Sensorwerte abgeleitet. So liefert beispielsweise die Betrachtung von Positionen, über die Zeit hinweg, zusätzliche Informationen über die Geschwindigkeit und die Richtung der Bewegung.

- **Sensordatenaggregation (engl. *Sensor Fusion*):** Durch die Kombination von Sensordaten ist es möglich deren Genauigkeit zu erhöhen, und es können zusätzliche Aussagen getroffen werden. Beispielsweise würde sich durch die Kombination einzelner Temperatursensoren am selben Ort eine erhöhte Genauigkeit und Zuverlässigkeit ergeben. Aus der Kombination von verteilten Temperatursensoren würde sich hingegen eine Aussage über die durchschnittliche Temperatur der Gegend ableiten lassen. Über die Aggregation von Sensordaten ist es zudem möglich die Datenmenge zu reduzieren.

2.5 Verwandte Arbeiten

Die Thematik der kontextverarbeitenden Systeme lässt sich auf unterschiedlichen Ebenen betrachten:

1. Identifizierung und Erfassung von relevanten Informationen.
2. Auswertung von Informationen zur Bestimmung von Kontexten.
3. Anwendung und Nutzung von Kontexten.

Diese Arbeit behandelt Problemstellungen, welche sich auf alle Ebenen verteilt wiederfinden.

2.5.1 Überblick über verwandte Forschungsarbeiten

Es gibt eine Vielzahl von Arbeiten, welche sich mit dem Bereich Kontext in Verbindung bringen lassen. In der Tabelle 2.1 wird ein Überblick über eine Auswahl verwandter Forschungsarbeiten gegeben[4]. Die ausgewählten Arbeiten betrachten in der Regel einen bestimmten Anwendungsfall oder adressieren einen bestimmten Aspekt, welcher in der Spalte *Schwerpunkt* umrissen wird. Diese Arbeit beinhaltet Aspekte im Bereich *Erfassung* von Informationen, *Beschreibung* von Diensten, sowie der *Auswertung und Nutzung* von Informationen. In dem Überblick werden die gewählten Ansätze der Arbeiten im Bezug auf diese Aspekte betrachtet[5]. Die dabei aufgezeigten Stichworte nehmen Bezug auf Inhalte, welche im weiteren Teil der Arbeit aufgezeigt werden. Die Begriffe in den Bereichen *Erfassung* und *Beschreibung*, umschreiben die Methoden, welche zur Anbindung von Sensoren genutzt werden (siehe Kapitel 4). Der Bereich *Verarbeitung* beschreibt die Art und Weise, in der die Daten weiterverarbeitet werden und nimmt dabei Bezug auf Abschnitt 5.1.1. In der Spalte *Nutzung* wird der Anwendungsbereich umschrieben. Der Kontext kann genutzt werden, um *Ereignisse* auszulösen, *Zustände* von Objekten zu beschreiben, direkt andere Systeme zu *steuern* oder er kann *bereitgestellt* werden, um von anderen Anwendungen weiterverwendet zu werden.

Die Details zu den einzelnen Arbeiten befinden sich im Anhang in Abschnitt A.1. Die letzte Spalte in Tabelle 2.1 verweist direkt auf den jeweiligen Abschnitt im Anhang.

[4] Weitere Ansätze, sowie weitere Formen der Kategorisierung finden sich in [Kja09].

[5] Bei der Zuordnung der Begriffe zu den jeweiligen Aspekten wurden die Schwerpunkte des Projektes betrachtet. Ein „-" bedeutet, dass die Arbeit innerhalb dieses Aspektes keinen Schwerpunkt aufweist.

Arbeit	Schwerpunkt	Erfassung	Beschreibung	Verarbeitung	Nutzung	Details
Context Toolkit [SDA99]	Rahmenwerk zur Kontextaggregation	Broker	Syntaktisch	Statisch	Ereignisse	A.1.1
Context-Aware and Location Systems (CALAIS) [Nel98]	Lokationsbestimmung in heterogenen Umgebungen	Broker	Syntaktisch	Statisch	Ereignisse	A.1.2
Context Aware Software (CIS) [Pas01]	Kontextverarbeitung, HCI	Broker	Syntaktisch	Fusion	Ereignisse (Notifikation)	A.1.3
Technology for Enabling Awareness (TEA) [Sch02c]	Kontextbestimmung auf mobilem Gerät	Statisch	-	Adaptiv	Zustände	A.1.4
CenceMe [MLF+08]	Kontextbestimmung auf mobilem Gerät	Statisch	-	Adaptiv	Zustände	A.1.5
MyConnector [DKS06a]	Bestimmung der Verfügbarkeit	Statisch	-	Adaptiv	Zustände, Steuerung	A.1.7
An Architecture for Context Prediction [FMR03]	Bestimmung und Vorhersage von Kontexten	Statisch	-	Adaptiv	Zustände	A.1.6
Umgebungsmodelle für Mobile Kontext-bezogene Systeme (NEXUS) [BBHS]	Breite Nutzung von Kontext	Middleware	Semantisch	Reasoning	Verschiedene Einsatzgebiete	A.1.8
Ambient Intelligence for the Networked Home Environment (AMIGO) [VRV05]	Kontextnutzung in der Heimautomation	Middleware	Semantisch	Reasoning, (Adaptiv)	Steuerung	A.1.9
Managing Context Data for Smart Spaces (MUSE) [CM00]	Vorverarbeitung von Sensordaten	-	Semantisch	Fusion	Bereitstellung	A.1.10
Web Presence for the Real World (CoolTown) [KBM+02]	Bestimmung relevanter Informationen in der Umgebung	Broadcast	Semantisch	-	Bereitstellung, Ereignisse	A.1.11
Notification Platform [HKPH03]	Bestimmung der Unterbrechbarkeit	Statisch	-	Adaptiv	Steuerung (von Notifikationen)	A.1.12
Smart Environments (MUNDO) [Ait06]	Kontextverteilung und -nutzung	Middleware	Syntaktisch	-	Bereitstellung	A.1.13
Suggested Upper Merged Ontology (SUMO), OntoSensor [PNL02]	Nutzung verteilter Ontologien	-	Semantisch	Reasoning	Zustände	A.1.14
Publish/Subscribe Applied to Distributed Resource Scheduling (PADRES) [LJ05]	Effiziente Verteilung von Kontext	Broker	Semantisch	Reasoning	Zustände, Ereignisse	A.1.15
Sensor Web Agent Platform (SWAP) [MS06]	Effiziente Aggregation von Sensordaten	Agenten	Semantisch	Reasoning	Bereitstellung	A.1.16
Heterogeneous Physical Devices in a Distributed Architecture (Hydra) [ZHF09]	Middleware zur Anbindung heterogener Geräte	Middleware	Semantisch	Reasoning	Bereitstellung	A.1.17

Tabelle 2.1: Überblick über verwandte Arbeiten

Alle diese Arbeiten haben als Gemeinsamkeit das Ziel, den Kontext in geeigneter Weise nutzbar zu machen. Ein System, welches Kontext nutzt, wird häufig als adaptiv bezeichnet, da der Gesamtprozess von den erfassten Informationen abhängig gemacht wird. Es gibt noch eine Vielzahl von Systemen, welche in diesem Bereich genannt werden können. In vielen dieser Systeme bestehen trotzdem weiterhin statische Elemente, wie eine Limitierung auf eine Menge bekannter Sensor- oder Datentypen, im Voraus bekannte oder festgelegte Sensoren oder eine statische Form der Auswertung (auch wenn sie durch die Nutzung von Ontologien sehr komplexe Strukturen annehmen kann).

Überschneidungen

Es gibt zu einigen der ausgewählten Arbeiten Überschneidungen zu Aspekten dieser Arbeit. Viele der Arbeiten fokussieren jedoch unterschiedliche Anwendungsgebiete, woraus sich andere Rahmenbedingungen ergeben. Manche der Arbeiten haben als Gemeinsamkeit die Herausforderung, dynamisch neue Elemente (seien es Dienste, Sensoren oder Agenten) an das System anzubinden und dabei die Interoperabilität zwischen den Elementen zu gewährleisten. Hierzu werden in manchen der vorgestellten Arbeiten Konzepte zur semantischen Beschreibung der Elemente genutzt. Häufig wird dabei der Schwerpunkt auf die automatische (jedoch im Endeffekt häufig statisch festgelegte) Auswertung der Informationen gelegt. Andere Arbeiten beschäftigen sich mit dem Problem der adaptiven Auswertung von Informationen. Viele der Systeme, welche diese Konzepte nutzen, stellen theoretische Ansätze da, sind jedoch noch in der Entwicklung oder werden bisher nur im wissenschaftlichen Umfeld genutzt.

Unterscheidungsmerkmale

Ein wesentliches Unterscheidungsmerkmal zwischen den verwandten Arbeiten und dieser Arbeit besteht in der Kombination aus Lernverfahren und lose gekoppelten Informationsquellen. In vielen Arbeiten liegt die *Intelligenz* des Systems in der Suche und der Aggregation von Informationen. Dies erfordert jedoch meist eine sehr umfassende Beschreibung der Zusammenhänge über Ontologien oder weitreichende Kenntnisse der verfügbaren Sensoren und deren Bedeutung zur Definition und Anwendung von geeigneten Anfragen und Aggregationsfunktionen. In dieser Arbeit dient die Suche in erster Linie zur Erfassung von *potentiell relevanten* Informationen. Es besteht somit vorerst eine lose Kopplung zwischen einer potentiell relevanten Informationsquelle und dem Konzept des Nutzers. Die Verbindung zwischen diesen beiden Elementen wird letztlich über das Modell hergestellt, welches sich an das Konzept des Nutzers anpasst. Welche Sensoren letztlich im Zusammenhang mit dem Konzept des Nutzers stehen, kann erst unter Berücksichtigung des Feedbacks des Nutzers entschieden werden. Die *Intelligenz* des angestrebten Systems liegt somit im Zusammenspiel aus Suche und Anpassung des Modells.

2.5.2 Verwandte Arbeiten am Fachgebiet KOM

Am Fachgebiet Multimedia Kommunikation (KOM) der TU Darmstadt, an dem auch diese Arbeit angefertigt wurde, sind eine Reihe von Arbeiten entstanden, welche mit dieser Arbeit verwandt sind oder welche ebenfalls einzelne Teilaspekte dieser Arbeit behandeln.

Kontextnutzung und Optimierung von Kommunikationssystemen

Die Dissertation von Görtz [Gör05] bildet die Basis für die grundlegende Idee dieser Arbeit. Sie behandelt ausführlich den Bereich der Kommunikationsdienste und bietet einen guten Überblick über den Themenbereich Kontextnutzung. Es werden einige Definitionen und Grundlagen erarbeitet, die auch für diese Arbeit Gültigkeit haben. Diese Arbeit greift manche der dort dargestellten Ideen auf und setzt diese um.

Beispielsweise wird in Kapitel 4.3 der Arbeit von Görtz ein *Kontext-Spiral-Modell* eingeführt, welches die Abfolge von Informationsgewinnung, Informationsauswertung, Kontextverteilung und Kontextnutzung darstellt. Eine ähnliche Abfolge ist auch in dieser Arbeit (beispielsweise in Abbildung 5.5) zu erkennen, mit dem Unterschied, dass als weiterer Schritt in der Abfolge das Modell zur Informationsauswertung adaptiv erweitert wird.

Die Arbeit von Görtz nennt Lernverfahren als eine Möglichkeit, eine Informationsauswertung durchzuführen. Diese Arbeit wendet diese Variante explizit an. In der Dissertation von Görtz wird in Kapitel 4.1.2 eine Kategorisierung von Systemen zur Kontextnutzung vorgenommen, indem anhand der Art und Weise der Kontextnutzung unterschieden wird. Mögliche Ziele der Kontextnutzung sind: Proaktive Auslösung, verbesserte Interaktion, Speicherung für spätere Ereignisse, Erinnerung zukünftiger Kontexte, Optimierung von Verhaltensmustern und Sammeln gemeinsamer Erfahrung. Das Rahmenwerk, welches in dieser Arbeit entwickelt wurde, legt den Fokus auf die Optimierung von Verhaltensmustern und nutzt dabei die Speicherung des Kontextes für eine spätere Adaption des Modells.

Die Funktionen, welche durch Einbeziehung des Kontextes möglich werden, sind auch von den Fähigkeiten des Kommunikationssystems abhängig. Am Fachgebiet KOM wurde diese Thematik im Rahmen der Dissertation von Ackermann [Ack03] behandelt.

Quality of Service und Netzdienste

Diese Arbeit benötigt möglichst geringe Bedienzeiten der Anfragen an Informationsquellen. Am Fachgebiet KOM wurde eine Reihe von Arbeiten angefertigt, welche sich detailliert mit der Thematik QoS in Netzwerken auseinandersetzen. Schmitt [Sch00b] und Karsten [Kar00] befassten sich mit Dienstgüte in heterogenen Netzwerken und den Möglichkeiten über Protokolle, wie dem *Resource Reservation Protocol* (RSVP) zur Reservierung von Bandbreiten, den Anforderungen seitens der Anwendung zu begegnen. Die Arbeit von Heckmann [Hec04] befasste sich mit der Herausforderung, die Qualitätsanforderungen an das unterliegende Netz von miteinander verbundenen Netzwerkbetreibern durch die Anpassungen des Datenverkehrs zu erfüllen. Ein ähnliches Problem, jedoch in einem anderen Kontext behandelt die Arbeit von Hollick [Hol04]. In seiner Arbeit behandelt er ebenfalls die Herausforderung Qualitätsanforderungen, wie die zuverlässige Übertragung von Daten, zu gewährleisten, jedoch in Ad hoc Netzwerken. Zwei weitere Arbeiten bei KOM beziehen sich auf die Verarbeitung und Erfüllung von Qualitätsanforderungen im Zusammenhang mit Netzdiensten. Diese Dienste sind voneinander abhängig und müssen entsprechend nacheinander angefragt werden, wobei jedoch Qualitätsanforderungen, welche über den gesamten Prozess hinweg Geltung haben, erfüllt werden sollen. Die Arbeiten von Berbner [Ber07] und Repp [Rep09] behandeln das Problem, indem für jede Aufgabe eine Menge von Netzdiensten zur Verfügung stehen, welche gegebenenfalls gegeneinander ausgetauscht werden können, um den Gesamtprozess im Rahmen der gestellten Rahmenbedingungen ablaufen zu lassen.

Semantik

Eine Reihe von aktuellen Arbeiten am Fachgebiet KOM befasst sich ebenfalls mit der Themenbereich Semantik. In den Arbeiten geht es primär um die Unterstützung von Lernsystemen durch die Nutzung von semantischen Annotationen von Dokumenten. Faatz [Faa04] beschäftigt sich ebenfalls mit der Nutzung von Ontologien und deren Anreicherung mit neuen Inhalten.

Sicherheit

Wenn es um die Verarbeitung von persönlichen Daten geht, nimmt Sicherheit eine zentrale Rolle ein. Wie dieser Aspekt im Rahmen dieser Arbeit betrachtet werden kann, wird in Abschnitt 7.1 erläutert. Im Zusammenhang mit der Absicherung eines Systems, ohne jedoch die Fähigkeiten zur Kommunikation zu beeinträchtigen, existieren bereits eine Reihe von Arbeiten. Die Dissertation von Roedig [Roe02] betrachtet in diesem Zusammenhang Architekturen, welche Firewall-Systeme nutzen, jedoch weiterhin für Multimedia-Applikationen eingesetzt werden sollen. Werden Anrufe anhand der Absenderkennung verarbeitet, so muss sichergestellt werden, dass diese nicht verändert oder gefälscht werden kann. Hierzu wurden im Rahmen dieser Arbeit mehrere Artikel veröffentlicht: [SAGS06, SHK+06, SHHS07].

3 Konzept und Architektur

Dieses Kapitel erläutert das Konzept des Gesamtsystems. Hierzu zählt zunächst eine grundlegende Behandlung des Begriffs *Kontext*, dessen Untergliederung in einzelne Teilaspekte und deren Definition. Des Weiteren wird in Abschnitt 3.2 die Architektur des Gesamtsystems aufgezeigt, welche die Aufteilung der Arbeit in einzelne Aufgabenbereiche beschreibt. In diesem Zusammenhang werden in Abschnitt 3.3 weitere Ziele und Eigenschaften für die Architektur beschrieben. Abschließend wird in Abschnitt 3.4 eine Qualitätsbetrachtung durchgeführt.

3.1 Kontextnutzung zur Verarbeitung von Kommunikationsanfragen

Dieser Abschnitt nimmt Bezug auf das Szenario zur Verarbeitung von Kommunikationsanfragen und beschreibt in diesem Zusammenhang die darin vorkommenden Elemente, sowie deren Rollen und Abhängigkeiten.

3.1.1 Kontextobjekte

Mit *Kontextobjekt* wird das Element bezeichnet, auf das sich eine Kontextinformation bezieht. Im Zusammenhang mit Kommunikationsdiensten kann hierbei zwischen folgenden Kontextobjekten unterschieden werden:

Absenderkontext (*Caller Context*)
Der Absenderkontext bezeichnet die Situation des Absenders beziehungsweise Anrufers. Er bezieht sich damit auf alle Informationen, welche zu dem Absender selbst in Relation stehen. Unter anderem gehören dazu: Die ihm zur Verfügung stehenden Endgeräte, seine Beziehung zum Angerufenen, seine Position oder seine Verfassung. Der Absender kennt den *Anfragekontext*. Daher ist es im Bedarfsfall theoretisch möglich, weitere Kontextinformationen über den Anrufkontext (beispielsweise durch Rückfragen über sprachgesteuerte Menüs) vom Absender zu beziehen. Dieser Zusatzaufwand seitens des Absenders sollte jedoch soweit wie möglich vermieden werden.

Empfängerkontext (*Callee Context*)
Da es in dieser Arbeit maßgeblich darum geht, die Vision eines Virtuellen Assistenten umzusetzen, welcher den Empfänger unterstützt, ist der Empfängerkontext von zentraler Bedeutung. Der Empfängerkontext bezieht sich auf alle Informationen, welche zu dem Empfänger in Relation stehen und Aufschluss über seine aktuelle Situation geben können.

3.1.2 Kontextdimensionen

Der Kontext lässt sich unter verschiedenen Aspekten betrachten. Zur Unterscheidung dieser verschiedenen Aspekte, wurden Kontextdimensionen eingeführt. Die Dimensionen des Kontextes einer Person, wurden in Abbildung 3.1 dargestellt und werden im folgenden Teil erläutert:

Standort (*Location*)
Einige Arbeiten haben vor allem die Bestimmung des Standortes zum Ziel, wie beispielsweise die Arbeit in Abschnitt A.1.2. Das Wissen, an welchem Ort sich ein Objekt oder eine Person aufhält, kann vielseitig genutzt werden. Es kann unter anderem dazu genutzt werden, um Dienste zu ermöglichen, welche Abhängig vom Standort eines Objektes sind (*Location Aware Services* oder *Location Based Services*) [KBM+02]. Des Weiteren kann das Wissen über den Aufenthaltsort einer Person genutzt werden, um Rufumleitungen und Erinnerungsfunktionen zu steuern oder um Rückschlüsse über die Verfügbarkeit eines Teilnehmers zu liefern. Zusätzliche Informationen über Geschwindigkeit und Ziel eines Objektes, lassen zudem Vorhersagen von Zustandsänderungen zu [FMR03, Meh08].

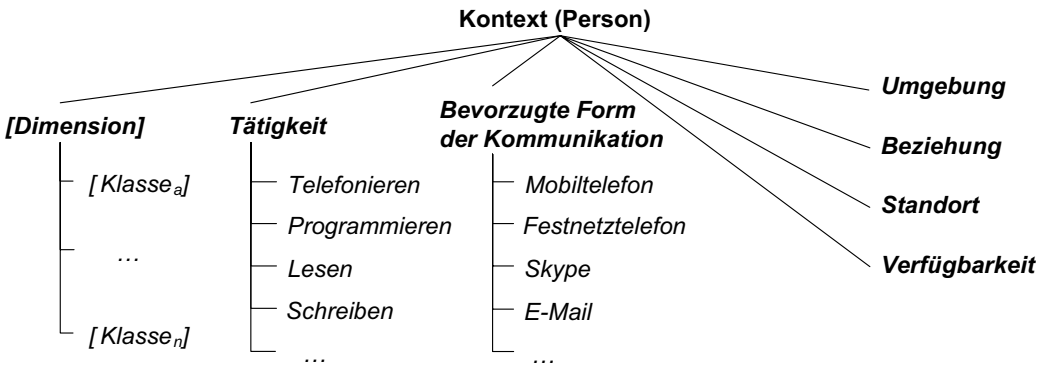


Abbildung 3.1: Kontextdimensionen einer Person

Beziehung (*Relation*)

Die Beziehung zwischen den Teilnehmern nimmt ebenfalls eine wichtige Rolle ein, z.B. bei der Frage, ob ein Anruf an den Angerufenen durchgereicht werden soll oder nicht. Neben der Einteilung der Beziehung in Klassen wie privat und geschäftlich, kann gerade im geschäftlichen Umfeld stärker zwischen verschiedenen Klassen von Beziehungen differenziert werden. Zum einen sind Personalhierarchien relevant und können beispielsweise genutzt werden, um Anrufe von Vorgesetzten bevorzugt zu behandeln. Zum anderen können Beziehungen auch die Relation zwischen den Rollen beider Teilnehmer widerspiegeln. Solche Rollen ergeben sich beispielsweise durch die Zuordnung von Aufgaben innerhalb von Projekten. Andere Klassen von Beziehungen ergeben sich auch zu *externen* Kontakten, wie zum Beispiel zu Kunden. Zwischen der Rolle des Kundenbetreuers und den Kunden kann eine Beziehung hergestellt werden, indem das Anliegen des Kunden (soweit ermittelt) mit dem Aufgabenbereich des Kundenbetreuers abgeglichen wird.

Verfügbare Endgeräte (*Capabilities, Available Devices*)

Neben der Frage, ob ein Teilnehmer aktuell verfügbar ist und Anfragen entgegen nehmen möchte, ist es für ein optimales Kommunikationsmanagement wichtig, Aussagen über die aktuell verfügbaren Endgeräte des Teilnehmers zu erhalten. Mit dem Wissen über die nutzbaren Endgeräte und deren Fähigkeiten können neben der direkten Durchleitung einer Anfrage auch Alternativen (wie beispielsweise die Umleitung einer Anfrage auf einen IMS-Dienst) genutzt werden.

Tätigkeit (*Task*)

Die Tätigkeit des Teilnehmers beschreibt die Art der Aufgabe, mit welcher der Teilnehmer gerade beschäftigt ist. Beispiele hierfür wären *Telefonieren*, *Texte verfassen*, *Lesen* oder *Diskutieren*. Während eines eingehenden Anrufes können Informationen aus dieser Kontextdimension genutzt werden, um die Verfügbarkeit eines Teilnehmers zu ermitteln.

Verfügbarkeit (*Presence*)

Die Verfügbarkeit bezeichnet den Zustand eines Objektes, meist in Relation zu der Fragestellung, ob eine Anfrage zum aktuellen Zeitpunkt möglich oder erwünscht ist. Die Bestimmung der Verfügbarkeit, auf Teilnehmer eines Kommunikationssystems angewendet, gibt Aufschluss darüber ob der Empfänger aktuell Anfragen entgegen nehmen möchte oder nicht. In einigen Kommunikationssystemen bestimmt der Empfänger selbst, in welchem Verfügbarkeitszustand er sich aktuell befindet, indem er manuell den Zustand festlegt (dies wird von einigen IMS oder VoIP System unterstützt, wie beispielsweise Skype). Die Klassen innerhalb dieser Dimension sind meist grob unterteilt, wie beispielsweise: *Verfügbar*, *Beschäftigt*, *Kurzzeitig nicht verfügbar*, *Längerfristig abwesend*.

Anfragekontext (*Call Context*)

Der Anfragekontext basiert auf Informationen, welche sich auf den Anruf selbst beziehen. In der Regel umfasst dies alle Kontextinformationen, welche sich aus der Anfrage, die bei dem Kommunikationssystem eingeht, ermitteln lassen. Aus der Anfrage selbst lassen sich beispielsweise gewählte Endgeräte und deren Fähigkeiten, sowie die Kennungen von Empfänger und Absender ableiten. Aus den Protokollen des Kommunikationssystems lassen sich in der Regel weitere Informationen ableiten, wie beispielsweise: Vorangegangene Anrufversuche des Absenders, sowie

Häufigkeit, Zeitpunkte und Dauer der letzten Gespräche. Je nach gewählter Form und Medium der Anfrage, ist die Menge der verfügbaren Kontextinformationen zum Inhalt der Anfrage selbst verschieden. Informationen über den Anfragekontext, welche genutzt werden können, um die Relevanz einer Anfrage für den Empfänger zu bestimmen sind beispielsweise Art (Anfrage, Rückfrage, Information) und Inhalt (Privat, Geschäftlich, Projektbezogen, Thema) des Anrufes, sowie dessen Dringlichkeit.

Bevorzugte Form der Kommunikation

Wenn dem Angerufenen mehrere Varianten zur Kommunikation zur Verfügung stehen, stellt sich die Frage, welche der Varianten vom Teilnehmer bevorzugt wird. Die Erfassung dieser Kontextdimension, also die Bestimmung der Form der Kommunikation, welche der Situation und der Vorstellung des Nutzers am besten entspricht, spiegelt das Ziel dieser Arbeit wieder. Diese Entscheidung ist stark von der Verfügbarkeit des Nutzers, und der Menge der möglichen Varianten der Kommunikation abhängig.

3.1.3 Abhängigkeiten der Kontextdimensionen

Neben den bisher erläuterten Kontextdimensionen gibt es noch weitere Kontextdimensionen, welche sich je nach Anwendungsgebiet unterscheiden. Kontextdimensionen sind ein Mittel, Kontexte zu gruppieren, welche vergleichbare Eigenschaften betrachten. Sie lassen sich jedoch nicht vollständig voneinander abgrenzen. Überschneidungen von Aussagen, Abhängigkeiten zwischen den Kontextdimensionen oder die Nutzung der gleichen Kontextinformationen lassen sich oft nicht vermeiden.

In der Abbildung 3.2 werden die Abhängigkeiten zwischen den zuvor erläuterten Kontextdimensionen dieser Arbeit dargestellt. Die Pfeile deuten an, welche Kontextdimensionen wiederum als Grundlage zur Entscheidungsfindung innerhalb anderer Kontextdimensionen dienen können.

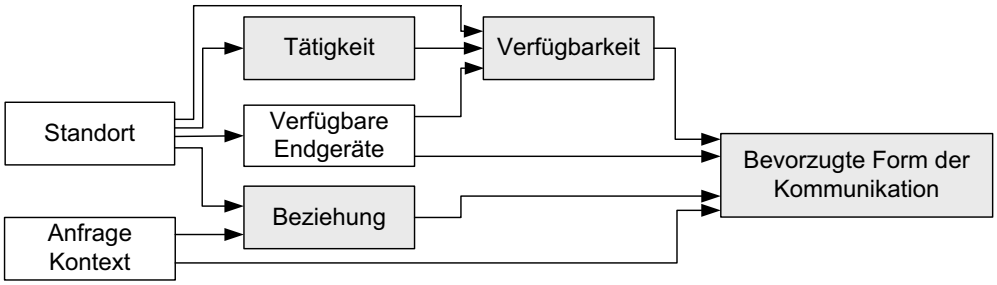


Abbildung 3.2: Abhängigkeiten der Kontextdimensionen

Die angegebenen Abhängigkeiten lassen sich wie folgt begründen:

- Der Standort eines Nutzers ermöglicht oft Rückschlüsse auf seine aktuelle Tätigkeit. Beispielsweise kann die Differenzierung des Aufenthaltsortes *Büro* von *Teeküche* bereits als wichtige Aussage über die wahrscheinliche Tätigkeit genutzt werden.
- Die Menge der verfügbaren Endgeräte ist oft abhängig von dem aktuellen Standort eines Teilnehmers.
- Durch Monitoring der Standorte (Nachbarschaftsbeziehungen/Treffen) und der Kontexte vorangegangener Kommunikationen können Beziehungsstrukturen gewonnen werden.
- Die Verfügbarkeit eines Teilnehmers hängt stark von seiner aktuellen Tätigkeit, seinem aktuellen Standort und den zur Verfügung stehenden Endgeräten ab.
- Die Entscheidung welche Kommunikationsform letztlich erwünscht ist, lässt sich gut aus den Zuständen anderer Kontextdimension ableiten. Sofern die aktuelle Verfügbarkeit, die Beziehung zum Absender und die dem Anfragekontext nach nutzbaren Endgeräte ermittelt wurden, ist diese übergeordnete Entscheidung aus diesen vorliegenden Ergebnissen ableitbar.

Oft nutzen Dienste nur einzelne Kontextdimensionen, oder sie werden innerhalb eines komplexen Dienstes an verschiedenen Stellen eines Ablaufs herangezogen. Ein weiteres Unterscheidungsmerkmal zwischen den Kontextdimensionen ist häufig die Methode, welche zur Auswertung der Kontextinformationen genutzt werden kann. Die in der Abbildung grau schattierten Kontextdimensionen sind abhängig von den jeweiligen Präferenzen eines Nutzers und müssen an den Nutzer selbst angepasst werden. Die in der Abbildung weiß dargestellten Kontextdimensionen

dagegen müssen nicht an einzelne Nutzer angepasst werden. Die dort genutzten Entscheidungsmodelle müssen an die dynamische Menge von Sensortypen beziehungsweise Kontextinformationen angepasst werden können. Danach können sie aber auf alle Nutzer gleichermaßen angewendet werden. Das bedeutet, dass für die weiß dargestellten Kontextdimensionen in der Regel ein nutzerübergreifendes Entscheidungsmodell ausreichend ist.

3.2 Gesamtarchitektur und Aufgabenbereiche

Dieser Abschnitt beschreibt die Entwicklung einer generischen Architektur für das Systems, welche sich aufgrund der Rahmenbedingungen und der benötigten Funktionalitäten ableiten lassen. Hierzu werden die notwendigen Komponenten identifiziert und deren Rollen beschrieben.

Aus dem Anwendungsszenario lassen sich folgende, für die Entwicklung der grundlegenden Architektur relevante Aufgabenbereiche ableiten:

- Die *Informationsquellen* sind verteilt verfügbar und befinden sich auf unterschiedlichen Systemen. Folglich müssen sich Teile der Architektur auf Seite der Informationsquellen befinden, um geeignete Schnittstellen zur Beschreibung und Anfrage der Informationen zur Verfügung zu stellen (siehe Abschnitt 3.3).
- Darauf aufbauend muss ein System zur *Informationsgewinnung* die Möglichkeit besitzen, diese Informationsquellen geeignet anzufragen. Je nach Kontext sind hierbei unterschiedliche Informationsquellen relevant. Diese müssen identifiziert und in angemessener Zeit abgefragt werden.
- Wie in Abschnitt 3.1.2 erläutert, hat jede Kontextdimension andere Anforderungen, was die Suche, die Auswertung und die Darstellung von Kontexten und Informationen betrifft. Für jede Kontextdimension wird daher eine entsprechend angepasste Form der *Informationsauswertung* benötigt, welches die Aufgabe der *Kontextbestimmung* erbringt.
- Je nach Szenario werden unterschiedliche *Anwendungen* auf die entsprechenden Kontextbestimmung zurückgreifen und den Kontext nutzen.
- Die Anwendung selbst oder der Nutzer dieser Anwendung kann mittels *Feedback* die Auswertung des Kontextes beeinflussen beziehungsweise steuern.

Diese Aufgabenbereiche, sowie deren Abhängigkeiten sind in Abbildung 3.3 dargestellt.

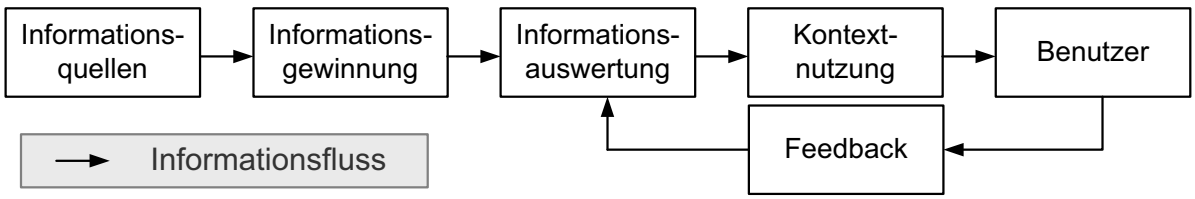


Abbildung 3.3: Überblick über die Aufgabenbereiche

Im folgenden Abschnitt werden diese Aufgabenbereiche eingeführt und behandelt, sowie grundlegende Überlegungen zur Entwicklung der Architektur erläutert. Die Arbeit orientiert sich anhand der hier dargestellten Struktur. In den einzelnen Abschnitten dieser Arbeit wird diese Struktur mehrfach aufgegriffen und verfeinert werden.

3.2.1 Informationsquellen

Von ihrer Funktion her betrachtet, sind Sensoren eine besondere Form der Dienste. Im Gegensatz zu anderen Diensten haben Sensoren hauptsächlich die Aufgabe, Informationen bereitzustellen. Als Lieferant für Informationen verfügen sie in der Regel über keine Methoden zur Verarbeitung von Daten (ausgenommen mögliche Schritte zur Vorverarbeitung ihrer eigenen Informationen). Um Informationen von den Endgeräten eines Nutzers zu erfassen, ist es notwendig, Sensoren auf den Endgeräten zu installieren. Hierdurch ergeben sich weitere Eigenschaften und Voraussetzungen:

- Ein Sensor darf den Nutzer nicht stören.
- Die Installation eines Sensors auf einem Endgerät sollte möglichst einfach durchzuführen sein.
- Das Ausführen eines Sensors sollte im Hintergrund möglich sein.
- Die auf dem Endgerät erzeugte Last sollte möglichst gering sein.

- Der Nutzer sollte Einblick in die Informationen haben, welche von seinem System erfasst werden.
- Der Benutzer selbst sollte die Möglichkeit haben, zu bestimmen, welche Informationen er bereit stellt.
- Nachträgliche Installation/Konfiguration von Sensoren sollte ohne Eingreifen seitens des Nutzers möglich sein (eventuell begleitet von einer Notifikation des Nutzers, um den vorangegangenen Aspekt zu berücksichtigen).

3.2.2 Informationsgewinnung

Der Bereich der Informationsgewinnung hat eine hohe Korrelation mit dem Aufgabenbereich der Informationsquellen. Durch die lose Kopplung zwischen den Informationsquellen und der Informationsgewinnung entstehen weitere Aufgabenbereiche. Als Grundlage zur Informationsgewinnung müssen Informationsquellen aufgefunden und verwaltet werden. Diese Aufgabe wird vom *Sensorverzeichnis* übernommen.

Um keine unnötige Redundanz bei der Verwaltung der Informationsquellen zu erhalten, soll möglichst nur ein System zur Informationsgewinnung für verschiedene Kontextdimensionen eingesetzt werden. Daher müssen Informationsquellen verschiedenster Art angebunden werden können. Für eine spätere Suche muss die Relation einer Informationsquelle zu der jeweiligen Kontextdimension bestimmt werden können. Hierbei spielt die *Beschreibung* einer Informationsquelle eine zentrale Rolle (siehe Abschnitt 3.3).

Sensorverzeichnis

Das Sensorverzeichnis bildet die Schnittstelle zwischen den Sensoren und den Diensten, welche auf die Sensordaten zugreifen möchten. Das Sensorverzeichnis registriert die Sensoren innerhalb seiner Umgebung (beispielsweise dem lokalen Netzwerk). Hierzu greift es auf Methoden zur Lokalisation von Diensten zurück (siehe 3.3).

Zu seinen zentralen Funktionalitäten gehört:

- Schnittstelle zur Registrierung von Sensoren.
- Suche nach verfügbaren Sensoren in der Umgebung.
- Bereitstellung von Referenzen auf die Schnittstellen der Sensoren.
- Erfassen/Anbieten der Beschreibungen der Sensoren.
- Überwachung der Verfügbarkeit der Sensoren.

Das Sensorverzeichnis bietet anderen Diensten die Möglichkeit, Informationsquellen schnell und ohne zusätzlichen Aufwand aufzufinden. Ein direkter Zugriff auf die Sensoren (ohne Sensorverzeichnis) ist prinzipiell möglich, setzt jedoch voraus, dass der entsprechende Sensor und dessen Schnittstellen vorab bekannt sind. Ein zentrales Management der Sensoren erspart einen zusätzlichen Funktionsumfang auf der Seite des Sensors. Somit muss ein Sensor nur über Funktionen zur Bereitstellung seiner Informationen verfügen.

Dies schließt jedoch nicht aus, dass ein Sensor durchaus auch in den Listen unterschiedlicher Sensorverzeichnisse geführt werden kann. Zur Skalierbarkeit in größeren Szenarien könnten auch mehrere Sensorverzeichnisse gekoppelt werden. Sofern jedes Sensorverzeichnis für einen speziellen Bereich (beispielsweise ein Gebäude) zuständig ist, könnten Suchanfragen an das entsprechende Sensorverzeichnis weitergeleitet werden, sobald dieser Bereich für die Anfrage relevant wird.

Suchdienst

Der Suchdienst ermöglicht es anderen Diensten, Suchanfragen an Sensoren zu stellen. Er kann dabei auf die Liste der verfügbaren Sensoren des Sensorverzeichnisses zurückgreifen. Je nach Bedarf können unterschiedliche Suchdienste mit verschiedenen Suchalgorithmen und Eigenschaften zum Einsatz kommen:

- **Semantische Suche:** Für das gegebene Szenario muss ein anfragender Dienst die Möglichkeit haben, diejenigen Sensoren anzufragen, welche zu einer bestimmten Kontextdimension und zu einem oder mehreren Kontextobjekten in Relation stehen, beziehungsweise relevante Informationen anbieten. Hierzu ist es notwendig, unter Nutzung der Schnittstellenbeschreibung und deren semantischer Beschreibungsmerkmale sowie einer Ontologie relevante Sensoren zu identifizieren.
- **Iterative Suche:** Zu Beginn einer Suchanfrage werden nur wenige Informationen verfügbar sein. In der Regel wird eine Suche mit den Angaben über die Kontextdimension und über das Kontextobjekt gestartet. Je nach verfügbaren Sensoren werden jedoch meist nur wenige Informationen direkt mit dem Kontextobjekt in Verbindung stehen. Weitaus mehr Informationen lassen sich ermittelt, wenn diese Informationen wiederum genutzt werden können, um weitere relevante Informationsquellen zu identifizieren und anzufragen.

Beispielsweise kann so in einem ersten Schritt festgestellt werden, in welchem Raum sich ein betrachtetes Objekt befindet, um dann in einem nächsten Schritt auch die Informationsquellen in dem Raum zur Beschreibung der Situation des Objekts heranzuziehen. Hier gilt es eine iterative Suche umzusetzen, welche die Menge der Suchergebnisse schrittweise durch weitere Suchanfragen über neu erhaltene Informationen erweitern kann.

Qualität der Suche und der Suchergebnisse

Die Umsetzung der Suche hat einen wesentlichen Einfluss auf die Erfüllung der Qualitätsanforderungen (siehe Abschnitt 3.4) durch das Gesamtsystem. Es müssen in kurzer Zeit viele aussagekräftige Informationen gesammelt werden, um schließlich gute Entscheidungen treffen zu können. Ein weiterer wichtiger Faktor in diesem Zusammenhang ist die Integration von Puffermechanismen zur Beschleunigung oder zur Einsparung von Suchanfragen.

3.2.3 Informationsauswertung

Wie in der Abbildung 3.3 dargestellt wurde, stellt die Informationsauswertung die Schnittstelle zwischen den Diensten, welchen einen Kontext nutzen wollen und den Diensten zur Erfassung der Informationen, dar. Die Informationsauswertung wird für eine bestimmte Kontextdimension instanziiert. Für die Nutzung des Kontextes bietet die Informationsauswertung eine Schnittstelle, welche der eines Sensors entspricht. Auf diese Weise können die Resultate einer Kontextdimension auch im Rahmen weiterer Kontextdimensionen herangezogen werden.

Wie in Abschnitt 2.4.3 beschrieben kann man zwischen verschiedenen Kontextdimensionen unterscheiden. Beispiele für Kontextdimensionen sind Verfügbarkeit, Beziehung, Tätigkeit, Standort oder verfügbare Endgeräte.

Für jede Kontextdimension sind unterschiedliche Mengen von Sensoren relevant, entsprechend werden andere Suchanfragen gestellt. Ebenso unterscheidet sich die Art der Auswertung je nach betrachteter Kontextdimension. Während sich die verfügbare Endgeräte über relativ statische Methoden bestimmen lassen, wird für die Bestimmung der Verfügbarkeit ein sehr komplexes und adaptives Verfahren benötigt.

Je nach Kontextdimension unterscheiden sich auch der Mechanismus zum Starten der Auswertung und die Vorhaltung der Kontexte. Ein Beziehungskontext zwischen zwei Personen kann erst zu dem Zeitpunkt der Anfrage gestellt werden, indem beide Personen miteinander in Kontakt treten oder es müsste jede mögliche Kombination vorab bestimmt werden. Dieser Beziehungskontext hat jedoch in der Regel eine relativ lange Gültigkeitsdauer und braucht in diesem Zeitraum nicht neu ermittelt zu werden. Auf der anderen Seite kann die Verfügbarkeit einer Person jederzeit ermittelt werden. Eine periodische Bestimmung der Verfügbarkeit bringt zudem den Vorteil, zum Zeitpunkt eines eingehenden Anrufes sofort auf den aktuell gültigen Wert zurückgreifen zu können, ohne eine zusätzliche Verzögerung in Kauf nehmen zu müssen, welche eine Suche mit sich bringen würde. Die Verfügbarkeit hat jedoch in der Regel nur eine relativ kurze Gültigkeit und muss daher in relativ kurzen Abständen neu ermittelt werden.

Auswertung

Die Auswertung analysiert den Kontext eines Kontextobjektes für eine bestimmte Kontextdimension. Das Resultat ist die in diesem Zusammenhang ermittelte Kontextklasse. Je nach Kontextdimension werden hierzu unterschiedliche Verfahren angewandt. Es gibt in dem Bereich drei grundlegende Ansätze:

- **Statische Auswertung:** Eine statische Auswertung läuft nach Regeln ab, die sich zur Laufzeit nicht ändern. Hierzu zählen auch skriptbasierte Ansätze.
- **Adaptive, unüberwachte Auswertung:** Eine adaptive Auswertung kann das Modell, welches zur Auswertung genutzt wird zur Laufzeit ändern beziehungsweise anpassen. Hierzu erhält das Verfahren jedoch nur die Informationen, welche in der Anfrage selbst enthalten sind.
- **Adaptive, überwachte Auswertung:** Eine überwachte Auswertung, erhält (im Gegensatz zur unüberwachten) neben den Anfragen selbst auch noch *Feedback* (siehe Abschnitt 3.2.4).

Insbesondere bei Kontextdimensionen, in welchen das *richtige* Resultat stark von den Wünschen und vom Empfinden eines Nutzers abhängig ist, wird eine adaptive, überwachte Auswertung benötigt. Dies gilt im Besonderen für die Dimensionen der Verfügbarkeit, welche eine zentrale Rolle im Zusammenhang mit einem System für Kommunikationsmanagement einnimmt.

Feedbackschnittstelle

Eine weitere Funktion der Informationsauswertung ist es, eine Schnittstelle für Feedback zu bieten. Feedback kann bei Kontextdimensionen, welche ein adaptives und überwachtes Verfahren zur Auswertung benötigen, benutzt werden, um den Prozess zur Adaption mit neuen Angaben zu versorgen. Hierzu kann ein Nutzer über unterschiedliche Wege Feedback geben (wie in Abschnitt 3.2.4 näher erläutert wird). Die Informationsauswertung bietet hierfür den Anknüpfungspunkt und bestimmt (je nach Instanziierung), wie das Feedback verarbeitet wird.

3.2.4 Kontextnutzung, Benutzer und Feedback

Die Anwendung, welche schließlich auf den Kontext zurück greift, wird als kontextberücksichtigende Anwendung (engl. Context-aware Application) bezeichnet. Bei den Szenarien aus Abschnitt 2.2, greift die Telefoniesoftware auf den Kontext zurück und nimmt somit die Rolle der kontextberücksichtigenden Anwendung ein. Es existieren eine Reihe weiterer Anwendungsmöglichkeiten, welche in Abschnitt 7 aufgeführt werden.

Feedback wird im Zusammenhang mit einer adaptiven, überwachten Auswertungsmethode relevant. Mit Feedback wird die Antwort beziehungsweise die Reaktion des Benutzers auf das Resultat der Kontextbestimmung bezeichnet. Das Feedback enthält neben der Anfrage, auf die sich das Feedback bezieht, auch das in diesem Zusammenhang erwünschte Resultat. Somit kann das Feedback auch als ein Ausschnitt der Vorstellungen eines Benutzers betrachtet werden. Die Problematik in diesem Bereich entsteht im Zusammenhang mit der vierten Anforderung aus Abschnitt 2.3: der Minimierung des Konfigurationsaufwandes seitens des Benutzers. Feedback vom Benutzer zu benötigen, bedeutet mehr Aufwand zu erzeugen. Da das System jedoch auf Feedback angewiesen ist, sollte dieser Aufwand so gering wie möglich gehalten werden. Daher ist es ein weiteres Ziel an dieser Stelle möglichst einfach zu nutzende Schnittstellen für den Benutzer anzubieten. Je nach Zusammenhang zwischen dem Feedback und dem Nutzerkonzept, welches vom bestehenden Modell beschrieben wird, kann zwischen mehreren Arten von Feedback unterschieden werden:

- **Negatives Feedback:** Wenn die Entscheidung des Systems nicht dem Konzept des Benutzers entspricht, kann man mittels Feedback korrigierend auf das Modell einwirken. Dieses Feedback widerspricht somit dem Nutzerkonzept, welches von dem aktuell bestehenden Modell beschrieben wird und wird somit als *negatives Feedback* bezeichnet.
- **Positives Feedback:** Entspricht das Feedback dem Nutzerkonzept, welches das bestehende Modell beschreibt, wird dies in dieser Arbeit als *positives Feedback* bezeichnet. Positives Feedback ist nicht zwangsläufig notwendig, um ein Modell anzupassen, es kann jedoch genutzt werden, um das Modell zu festigen.
- **Falsches Feedback:** Feedback, welches nicht dem eigentlichen Konzept des Benutzers entspricht, wird als *falsches Feedback* bezeichnet. Falsches Feedback kann beispielsweise durch ein Versehen seitens des Benutzers entstehen, oder ebenso wenn ein automatisierter Prozess zur Erzeugung von Feedback genutzt wird. Falsches Feedback führt möglicherweise zu einem Fehlverhalten des Modells und sollte daher weitgehend vermieden werden.

Je nach Kontextdimension und Anwendung entstehen unterschiedliche Möglichkeiten, Feedback vom Benutzer zu beziehen. In einigen Bereichen ist es von Vorteil, eine solche *Feedback-Schnittstelle* direkt in die Anwendung zu integrieren oder als zusätzliche Komponente auf das Endgerät zu installieren. Eine solche Komponente wird im Rahmen dieser Arbeit auch als *Feedbackagent* bezeichnet.

Ein wesentliches Unterscheidungsmerkmal zwischen den Ansätzen für einen Feedbackagent ist die Art und der Zeitpunkt der Generierung des Feedbacks (siehe auch Abschnitt 3.2.4). Wird das Feedback aus einer bestimmten Situation heraus gegeben (*Implizites Feedback*), können direkt die Merkmale, die diese Situation beschreiben genutzt werden. Wird das Feedback erst im Nachhinein gegeben, so bezieht sich das Feedback auf Merkmale einer zeitlich zurückliegenden Situation (*Explizites Feedback*). Für das explizite Feedback müssen die Merkmale zurückliegender Entscheidungen in einer Historie gehalten und für den Benutzer zur Entscheidungsfindung einsehbar sein. Diese Variante erfordert jedoch deutlich mehr Aufwand und eine Oberfläche, über die es dem Benutzer möglichst einfach gemacht wird, die Merkmale der jeweiligen Situation zu betrachten und selbst eine Entscheidung zu treffen.

3.3 Architekturelemente

Die Architektur des Gesamtsystems besteht aus einer Vielzahl von Komponenten. Diese Komponenten kommen unter anderem auch verteilt auf unterschiedlichen Systemen zum Einsatz. So müssen die Komponenten zur Ver-

arbeitung von Informationsquellen häufig auf derjenigen Plattform integriert werden, auf der diese Information verfügbar ist. Ein Feedbackagent sollte so umgesetzt werden, dass es dem Benutzer möglichst einfach ist, Feedback zu geben. Dies bedeutet das ein Feedbackagent idealerweise auch für Endgeräte angepasst und umgesetzt wird, welche der Benutzer mit sich führt.

Wiederum andere Komponenten, wie beispielsweise solche zur Registrierung und Suche über Informationsquellen, müssen derart integriert werden, dass sie Zugang zu den Informationsquellen haben (beispielsweise sich im gleichen lokalen Netz befinden). Komponenten zur Auswertung und Nutzung von Informationen können dagegen auf zentralen Systemen zum Einsatz kommen.

Daher muss die Architektur die Umsetzung der jeweiligen Komponenten auf den benötigten Plattformen und die Kommunikation zwischen den Komponenten ermöglichen.

Sensoren als Dienste

Es gibt zwei grundlegende Möglichkeiten externe Elemente unterschiedlicher Art in ein System zu integrieren. Entweder das System erhält die Fähigkeit mit jedem einzelnen Element zu kommunizieren, wozu möglicherweise eine Vielzahl von Varianten der Kommunikation umgesetzt werden müssen, oder beide nutzen eine einheitliche Schnittstelle zur Kommunikation. Die zweite Variante verspricht letztlich einen flexibleren Einsatz, da nachträglich auch weitere Arten von Sensoren angebunden werden können.

Der Ansatz einer einheitlichen Schnittstelle zur Anbindung eines Elements in einen Gesamtprozess, findet sich ebenfalls im Bereich der Netzdienste wieder. Sensoren können prinzipiell als Dienste betrachtet werden. Im Unterschied zu dem normalen Verständnis eines Netzdienstes, führen Sensoren jedoch im Allgemeinen keine (komplexen) Operationen durch, sondern liefern lediglich auf Anfrage Zustandsinformationen oder bilden Eingabeparameter auf Ausgabeparameter ab. Die einfache Beschaffenheit eines Sensor-Dienstes führt im Vergleich zu normalen Netzdiensten zu einer Vereinfachung der Beschreibung der Schnittstelle und deren Funktionalitäten. Durch diese Eigenschaft werden Mechanismen im Bereich der Auffindung relevanter Sensoren möglich, welche im Bereich der Suche nach relevanten Netzdiensten nicht geeignet sind (siehe Abschnitt 4.6.2). Um bereits existierende Informationsquellen zu integrieren, müssen diese gegebenenfalls gekapselt und/oder um die entsprechenden Schnittstellen und deren Beschreibung erweitert werden.

Dienstorientierte Architektur

Die einzelnen Aufgabenbereiche des angestrebten Systems lassen sich gut voneinander trennen. Eine Einteilung und Umsetzung dieser Aufgabenbereiche in Form von Diensten ermöglicht es, ein komplexes System in einzelne Komponenten mit dedizierten Aufgaben und Rollen zu unterteilen. Diese Aufteilung in einzelne Komponenten wiederum bietet ebenso den Vorteil, klare Schnittstellen zwischen den einzelnen Diensten zu erhalten.

Eine dienstorientierte Architektur (*Service-oriented-Architecture* - SoA) hat auch einen erheblichen Einfluss auf die Skalierbarkeit eines Systems. In einer solchen Architektur können Dienste auf unterschiedlichen Plattformen untergebracht oder repliziert werden, um Lasten und Aufgaben zu verteilen. Eine dienstorientierte Architektur kommt somit den Anforderungen (5) und (6) aus Abschnitt 2.3 (*Minimierung der Kosten* und *offenes System*) entgegen.

Plattform

Wie zu Beginn dieses Abschnittes erläutert, ist ein Ziel der Architektur auf möglichst unterschiedlichen Systemen zum Einsatz zu kommen. Um bei möglichst vielen Systemen auf denselben Programmcode zurückgreifen zu können, bieten sich vor allem plattformunabhängige Programmiersprachen an, welche in der Regel auf der jeweiligen Plattform durch einen Interpreter ausgeführt werden. In diesem Bereich ist Java (neben Alternativen wie beispielsweise Ruby oder Python) eine sehr weit entwickelte Programmiersprache mit vielen Funktionalitäten.

Mittels Java Anwendungen können neben Computern auch eine Reihe von einfacheren Plattformen adressiert werden. Java Code kann inzwischen auch auf vielen Routern, Smartphones oder leistungsfähigen Sensorplattformen genutzt werden. Für Plattformen auf denen die Ausführung von Java Code nicht möglichen ist, muss die Möglichkeit bestehen, ebenso an das System angebunden zu werden. Dieses Problem kann über einen Ansatz zur einheitlichen, plattformübergreifenden Form der Kommunikation adressiert werden.

Middleware

Das Gesamtsystem hat zum einen die Aufgabe eine Plattform zu bieten, die es ermöglicht, Informationsquellen aller Art auf einer einheitlichen Ebene zu adressieren und zu nutzen. Auf dieser Ebene sollen zum anderen Dienste verschiedenster Art diese Informationen auswerten und nutzen können. Eine solche Plattform kann auch als Midd-

leware bezeichnet werden. In Abbildung 3.4 ist allgemein dargestellt, was unter einer Middleware zu verstehen ist.

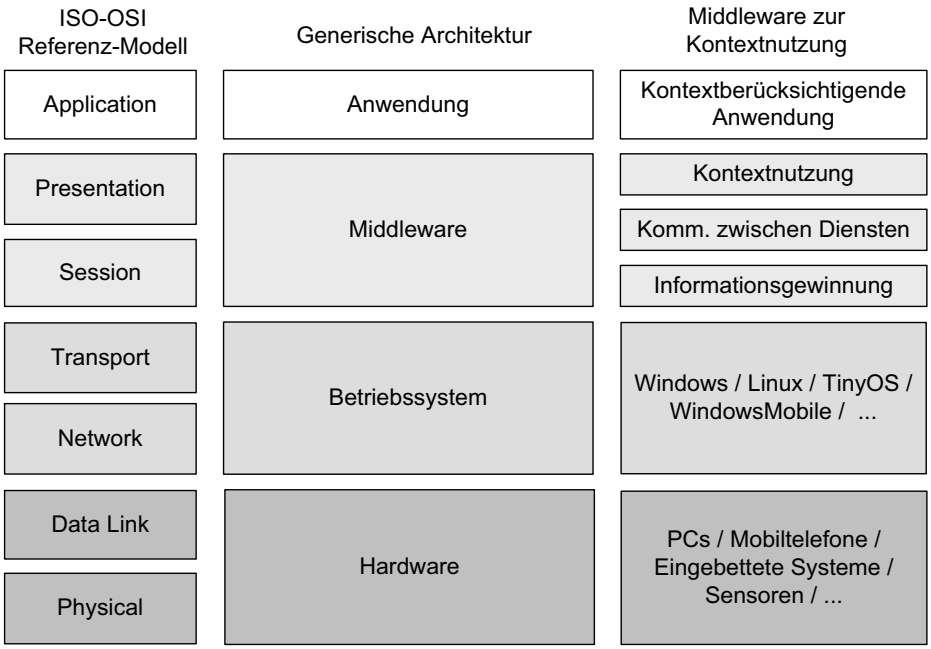


Abbildung 3.4: Middleware - Einordnung der gewählten Architektur

Auf der linken Seite der Darstellung ist das ISO-OSI Referenz-Modell aufgezeigt [SN04]. Die Form Middleware, wie sie in dieser Arbeit betrachtet wird, kann auf den Ebenen *Session* und *Presentation* des ISO-OSI Modells eingeordnet werden. Die Middleware bildet eine Ebene zwischen Anwendung und dem unterliegenden System (wie im mittleren Teil der Darstellung gezeigt wird). Somit bietet eine Middleware eine einheitliche Schnittstelle zur Kommunikation zwischen verschiedenen Anwendungen. In der Literatur werden eine Reihe von Ansätzen für Middleware aufgezeigt [MCE02]. Da der Begriff Middleware relativ vielseitig belegt ist, wird im Folgenden die in dieser Arbeit angestrebte Form der Middleware genauer beschrieben.

Je nach Ebene in welcher das Gesamtsystem betrachtet wird, kann das System mit einer der folgenden Kategorien für Middleware beschrieben werden (siehe rechte Seite der Abbildung):

- **Kommunikations-Middleware:** Auf der Ebene der *Informationsgewinnung* geht es um die Verbreitung von Zuständen von Sensoren. Hier muss eine Kommunikation über verschiedene Plattformen hinweg möglich sein. Auf dieser Ebene können auch weiter reichende Ansätze wie Nachrichten- oder Ereignisbasierte Middleware oder verteilte Datenbanken zum Einsatz kommen.
- **Objektorientierte Middleware:** Auf der Ebene der Dienste geht es um die Bereitstellung von Funktionen wie beispielsweise dem Sensorverzeichnis. Die *Kommunikation zwischen den verteilten Diensten* soll möglichst einfach und transparent möglich sein. Hierzu können entfernte Dienste als (Proxy-) Objekte im lokalen System eingebunden werden.
- **Kontextbasierte Middleware:** Auf der Ebene der *Kontextnutzung* geht es darum den Anwendungen eine allgemein nutzbare Schnittstelle zur Integration des Kontextes in den Programmablauf zu ermöglichen. In vielen dieser Ansätze wird mit dem Begriff *Kontext* jedoch meist nur die Menge erfasster oder gegebenenfalls aggregierter Kontextinformationen bezeichnet. Eine Klassifizierung in höherwertige Kontextklassen, wie sie in dieser Arbeit angestrebt wird, zählt meist nicht zu den Funktionen einer solchen Middleware.

Kommunikation

Das Anwendungsszenario (wie in Abschnitt 2.2 beschrieben) erfordert, dass die Kommunikation zwischen unterschiedlichen Diensten und Sensoren auch über heterogene Netzwerke und Systeme hinweg möglich ist. Es gibt eine Reihe von Ansätzen, um unterschiedliche Dienste miteinander zu verbinden. Hierzu zählen zum einen Ansätze, welche Kommunikations-Middleware nutzen. Zum anderen können sich die unterschiedlichen Dienste auch eine gemeinsame Sprache zum Informationsaustausch nutzen. Die zentrale Eigenschaft aller Ansätze ist, dass die Dienste eine Schnittstelle geboten bekommen, über welche sie *transparent* miteinander kommunizieren können.

Eine transparente Kommunikation zwischen Diensten bedeutet, dass Dienste nur eine abstrahierte Sicht auf die darunter liegende Kommunikation erhalten und auf entfernte Dienste wie auf lokale Objekte zugreifen können.

Konnektor

Ein Architekturelement, welches eine transparente Kommunikation zwischen Diensten ermöglicht ist ein sogenannter Konnektor. Ein Konnektor bildet eine Abstraktionsschicht zwischen dem Dienst und der Kommunikation. Hierzu bietet ein Konnektor auf der Seite der Dienste eine einheitliche Schnittstelle zur Kommunikation zwischen den Diensten.

Je nach Bedarf oder zugrunde liegendem System und Netzwerk können unterschiedliche Konnektoren eingesetzt werden. Ob ein Konnektor dabei auf eine Middleware zurück greift oder eine gemeinsame Sprache zum Informationsaustausch nutzt, ist für den Dienst, welcher ihn verwendet, nicht relevant. Details zu der Anwendung und Umsetzung von Konnektoren finden sich in Abschnitt 6.1.1.

Auffindung von Diensten

Ziel der Auffindung von Dienstes ist es, möglichst ohne konkretes Vorwissen andere Dienste zu lokalisieren. In der Regel liefert die Lokalisation eines Dienstes eine eindeutige Adresse zurück, über welche der Dienst angesprochen werden kann. Als Format für solche Adressen kommen häufig *Uniform Resource Identifier* (URI) zum Einsatz.

Es gibt in der Regel zwei grundsätzliche Varianten, wie Dienste lokalisiert werden. Entweder ein Dienst registriert sich bei einer vorab bekannten, zentralen Instanz (Vermittler, Broker, Registrar) oder es wird im Netzwerk nach dem jeweiligen Dienst gesucht. Eine solche Suche kann sich unterschiedlich gestalten. Neben einigen Ansätzen für die Suche innerhalb von Peer-to-Peer Systemen werden von der suchenden Instanz in der Regel Broad- oder Multicastnachrichten im Netzwerk versendet, worauf die betreffenden Dienste antworten.

Die Peer-to-Peer basierten Ansätze, sind zwar prinzipiell geeignet, wurden jedoch bisher, unter anderem wegen dem angestrebten Einsatz innerhalb von Unternehmen ausgelassen. Unternehmensnetze bieten meist eine in der Größe limitierte und relativ feste Infrastruktur, was den Einsatz zentralisierter Ansätze vereinfacht. Der Einsatz einer vermittlerbasierten Auffindung von Diensten eignet sich gerade in Bereichen in denen Dienste angebunden werden sollen, welche sich außerhalb der *Reichweite* von Broad- oder Multicastnachrichten befinden, wie beispielsweise außerhalb eines lokalen Netzwerkes. Andererseits bieten Systeme, welche die lokal verfügbaren Dienste auffinden, die Möglichkeit der Nutzung von Diensten, ohne das Vorwissen der Adresse eines Vermittlers. Dieses erhöht den Komfort und minimiert den Aufwand der Einrichtung eines Systems und kommt somit der Anforderungen zur Aufwandsminimierung aus Abschnitt 2.3 entgegen. Wie diese Aufgabe im Rahmen dieser Arbeit angegangen wird, ist in Abschnitt 4.2.2 beschrieben.

Beschreibung

Aus Abschnitt 2.3 geht hervor, dass bei der Informationsgewinnung auf externe Informationsquellen zurückgegriffen werden soll, ohne vorab deren Schnittstellen zu kennen. Hierdurch ist es möglich, ein offenes System zu bieten, um auch zur Laufzeit noch neue, unbekannte Sensoren hinzufügen zu können. Hierdurch entstehen aber zusätzliche Aufgaben:

- **Schnittstellen:** Methoden des Sensors müssen bestimmt werden können.
- **Relevanz:** Zugehörigkeit eines Sensors ist relevant, um zu bestimmen, für welche Anfragen ein Sensor herangezogen wird.
- **Datentyp:** Die Daten, welche der Sensor liefert müssen klassifiziert werden, um sie zu geeignet (ihrer Bedeutung nach) zu verarbeiten.
- **Metainformationen:** Weitere Informationen, beispielsweise über die Gültigkeit oder Genauigkeit des Sensorwertes können helfen, die Informationen optimal zu nutzen.

Die Beschreibung eines Dienstes oder eines Sensor nutzt hierzu folgende Komponenten:

- **Beschreibung der Schnittstellen:** Die Beschreibung der Schnittstelle ist notwendig, um dem Anfragen Dienst die Möglichkeit zu bieten zu ermitteln, über welche Funktionen und Methoden ein anderer Dienst oder Sensor verfügt, ohne ihn zu vorab zu kennen. Die Beschreibung der Schnittstellen listet die verfügbaren Methoden, sowie deren Ein- und Ausgabeparameter auf. Aus technischer Sicht reichen diese Informationen aus, um eine Anfrage an den entfernten Dienst zu stellen. Es existiert eine Reihe von Ansätzen aus dem Bereich der *Webservices*, welche sich hierfür anbieten[1].

[1] `http://www.w3.org/TR/2007/REC-wsdl20-primer-20070626/`

- **Semantische Beschreibungsmerkmale:** Die Beschreibung der Schnittstelle reicht nicht aus, um zu entscheiden, für welche Anfrage ein Sensor relevante Informationen anbietet. Ebenso geht aus einer einfachen Beschreibung der Schnittstellen nicht hervor, wie ein Rückgabeparameter zu interpretieren ist oder welche Informationen als Eingabeparameter geeignet sind. Hierzu werden zusätzliche Informationen benötigt, welche Rückschlüsse auf die Relevanz eines Sensors und die Bedeutung der Daten ermöglichen - sogenannte *semantische Beschreibungsmerkmale*.
- **Ontologien:** Ontologien ermöglichen es Beziehungen zwischen Elementen zu definieren. Setzt man sie in Kombination mit semantischen Beschreibungsmerkmalen ein, bieten sie die Möglichkeit Dienste und deren Daten zueinander in Relation zu stellen. Hierzu verweisen semantische Beschreibungsmerkmale in der Regel auf Elemente innerhalb einer Ontologie. Innerhalb einer Ontologie werden Beziehungen häufig in Form einer Subjekt-Prädikat-Objekt Beziehung dargestellt. Durch Verknüpfung vieler solcher Subjekt-Prädikat-Objekt Beziehungen lassen sich komplexe Beziehungsstrukturen abbilden. Solche Strukturen können je nach Umsetzung beispielsweise Baumstrukturen oder gerichtete Graphen ergeben.
- **Weitere Meta-Informationen:** Neben den Beschreibungselementen zur Bestimmung der Bedeutung des Sensors oder seiner Daten, werden Informationen benötigt, um die Verarbeitungsweise zu definieren. Beispielsweise kann je nach Sensor die Gültigkeit, die Genauigkeit, die Auflösung oder das Format der Daten variieren. Diese Informationen werden benötigt, um Elemente wie Puffer, Vorverarbeitung oder Konvertierung zu steuern.

3.4 Qualitätsbetrachtungen

Qualität wird definiert als:

> *Grad, in dem ein Satz inhärenter Merkmale Anforderungen erfüllt.*[2]

Aussagen über die Qualität sind an Schnittstellen zwischen Diensten notwendig. Mittels Vorgaben über die Qualität können Rahmenbedingungen für die Ausführung von Prozessen bestimmt werden. Qualität kann unter verschiedenen Gesichtspunkten betrachtet werden:

3.4.1 Dienstgüte - QoS

Die Gewährleistung von Dienstgüte wird auch als *Quality of Service* (QoS) bezeichnet [Ste00]. QoS-Merkmale beschreiben die (garantierte) Leistungsfähigkeit eines Systems. QoS wird oft als Überbegriff über eine Reihe von betrachteten Merkmalen angewendet. Die im Rahmen dieser Arbeit relevanten Merkmale sind:

- **Zeitliche Vorgaben für eine Kontextanfrage:** Ein Hauptaspekt von QoS ist die Einhaltung von Zeitvorgaben. Gerade im Bereich der Netz- und Kommunikationsdienste spielen QoS-Aspekte eine entscheidende Rolle. Ein Kommunikationsdienst muss beispielsweise eine Anfrage innerhalb kürzester Zeit verarbeiten können, um im Protokoll der Endgeräte definierten Zeitvorgaben zu erfüllen.
- **Latenzzeit der Abfrage einer Informationsquelle:** Neben der Einhaltung zeitlicher Vorgaben für das Gesamtsystem, spielt auch die Latenzzeit bei der Kommunikation zu den Informationsquellen eine entscheidende Rolle.
- **Minimierung des Aufwandes seitens des Nutzers:** Teile des Systems sollen auf Seite des Nutzers laufen, möglichst ohne ihn zu stören. Aufwand entsteht durch Nutzung der Ressourcen des Systems des Nutzers (CPU, Speicher, Netzwerk, Batterie). Sobald dieser Aufwand vom Nutzer als störend wahrgenommen wird, verringert sich der vom Nutzer empfundene Mehrwert zur Nutzung des Systems.

3.4.2 Qualität der Informationen und des Kontextes

Ein besonderer Aspekt von Qualitätsmerkmalen wird durch die Begriffe *Quality of Information* (QoI) und *Quality of Context* (QoC) beschrieben. In der Literatur finden sich einige Arbeiten, welche beiden Begriffe eine vergleichbare Bedeutung zuordnen. Im Rahmen dieser Arbeit wird der Begriff QoI genutzt, um die Informationsgrundlage, also die Sensoren und deren Eigenschaften zu beschreiben. QoC hingegen beschreibt die Informationen, die durch die Suche erfasst und zur Auswertung herangezogen werden.

[2] nach DIN EN ISO 9000:2005

QoI beschreibt die Anforderung, Informationen mit möglichst *ausreichender* Aussagekraft zur Verfügung zu stellen. QoI Anforderungen sind also dann erfüllt, wenn es prinzipiell möglich ist, auf der Basis der verfügbaren Informationen, eine zutreffende Entscheidung zu fällen.

Die Qualität der Informationen kann durch verschiedene Aspekte beeinträchtigt sein [BDPT07]. Neben der Genauigkeit, der Aktualität, der Verlässlichkeit und der Verfügbarkeit von Sensoren und deren Messdaten, spielt die Abdeckung der vom Konzept des Nutzers herangezogenen Merkmale durch Sensoren eine entscheidende Rolle.

Das Konzept des Nutzers kann als Funktion $h()$ beschrieben werden, welche aus einer Menge von Merkmalen $M = m_1, m_2, \ldots, m_n$, das gewünschte Ergebnis $k \in K$ ermittelt $k = h(M)$.

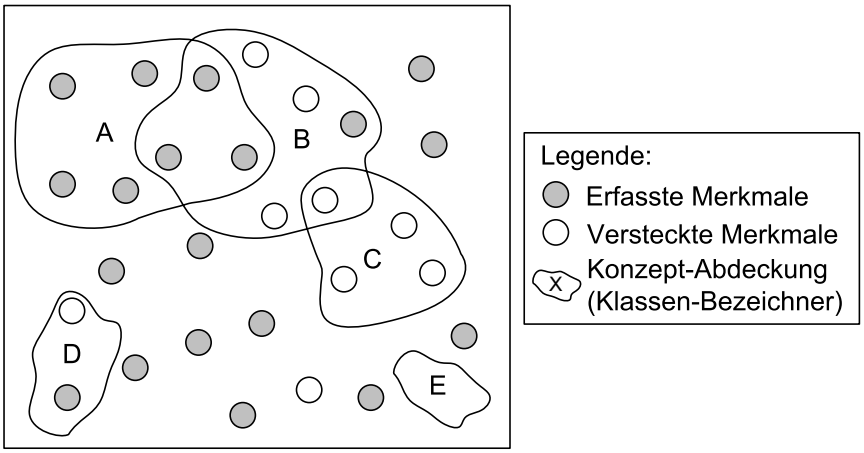


Abbildung 3.5: Qualität der Information - Nutzerkonzept und Informationen

In Abbildung 3.5 wird das Problem skizziert. Gegeben sei ein Nutzer, der in einer Kontextdimension ein Nutzerkonzept mit fünf Kontextklassen besitzt: $K = A, B, C, D, E$. Diese Klassen basieren auf Merkmalen M, welche in der Abbildung als Kreise angedeutet sind. Jedem dieser Klassen kann eine Menge von Merkmalen zugeordnet werden, was wiederum als Wolke dargestellt wird. Die Zuordnung beschreibt die Abhängigkeit zwischen der jeweiligen Klasse k und denjenigen Merkmalen, welche in der Funktion $h()$ einen Einfluss auf das Resultat haben.

Dunkel dargestellte Kreise beschreiben diejenigen Merkmale, welche durch Sensoren erfasst werden können. Die hell dargestellten Merkmale verwendet der Nutzer implizit in seinem Konzept für seine Differenzierung zwischen den Klassen. Diese *versteckten* Merkmale sind dem System jedoch nicht zugänglich und können nicht für eine Entscheidungsfindung herangezogen werden.

Bei dieser Darstellung werden die Werte der Merkmale zunächst nicht betrachtet, sondern nur die Relevanz eines Merkmals für das Konzept des Nutzers. Je nach Kontextdimension und Konzept des Nutzers kommen andere Merkmale und Sensoren in Betracht.

Jede Kontextklasse k steht zu einer bestimmten Menge von Merkmalen $M_k \subseteq M$ in Relation. Im Allgemeinen werden diese Merkmale werden jedoch nur zum Teil von Sensoren abgedeckt $S_k \subseteq M_k$.

Die erreichbare QoI bestimmt sich durch:

- Der Anzahl der Merkmale einer Kontextklasse $m = |M_k|$.
- Deren Abdeckung durch Sensoren $s = |S_k|$.
- Der versteckten Merkmale $H_k = M_k - S_k$ mit $v = |H_k|$.

Ist $m = 0$, liegt eine willkürliche Entscheidung vor, welche nicht durch Merkmale beschrieben und daher auch nicht über ein beobachtendes System ermittelt werden kann (entspricht Kontextklassen E). Je größer m ist, desto mehr Aussagen, können über einen Zustand getroffen werden. Gelingt es diese Aussagen durch Sensoren zu erfassen, so können diese genutzt werden, um Entscheidungen zu treffen. Die erreichbare QoI ist umso höher, je umfassender dies möglich ist. Bei $v = 0$ ist eine maximale erreichbare QoI gegeben (wie in Konzept A dargestellt). Dagegen würde im Falle $v = m$ der niedrigsten erreichbaren QoI entsprechen (siehe Konzept C).

Die Kontextklassen A und B beziehen sich zum Teil auf dieselben Sensoren. Gerade bei exklusiven Zuständen kann diese Menge ausschlaggebend für die Entscheidung zwischen den Zuständen sein.

Jedes Merkmal führt nur mit einer bestimmten Wahrscheinlichkeit zur korrekten Bestimmung des Zustandes. Im Allgemeinen werden mehrere Merkmale dazu genutzt, den Zustand eines Objektes zu beschreiben. Betrachtet man

die Kovarianz der Merkmale zueinander, so ist zu erwarten, dass diese häufig Korrelationen zueinander aufweisen. Je größer diese Korrelation zwischen den Merkmalen ist, desto höher ist die Redundanz r.

Der Wert r bezeichnet die durch die Korrelation der Aussagen der Merkmale hinzugewonnene Redundanz. Bei steigender Anzahl von Merkmalen, welche genutzt werden können, um einen Zustand zu beschreiben, und gleichzeitig gleich bleibender Komplexität (siehe Abschnitt 3.4.4), steigt diese Redundanz im Allgemeinen.

Die Merkmale die mit den Kontextklassen B und D in Relation stehen, werden nur zum Teil abgedeckt. In dem Fall, dass die Kontextklassen B und D ähnliche Komplexität aufweisen, jedoch B durch eine größere Redundanz r in den Merkmalen die zur Verfügung stehen aufweist, kann davon ausgegangen werden, dass dort eine höhere Wahrscheinlichkeit besteht, *ausreichend* viele Informationen zu sammeln, um eine zutreffende Entscheidungen treffen, als für Zustand D.

3.4.3 Qualität des Kontextes

Nach der Definition aus Abschnitt 2.4.3, umfasst der Kontext eines Objektes eine Menge von Merkmalen, welche den Zustand des Objektes, beziehungsweise die Situation, in welcher es sich befindet, beschreiben. Die Qualität des Kontextes (QoC) beschreibt die Qualität der Menge der von der Suche erfassten Merkmale. Die Qualität dieser Beschreibung hängt maßgeblich von folgenden Faktoren ab [Zim06]:

- Genauigkeit, Abstraktionsgrad, Alter der Daten (*Aussagekraft*).
- Bezug der Daten zum Kontextobjekt (*Relevanz*).
- Auswirkungen der Durchführung der Suche (Systematische oder statistische Fehler, welche im sich im Endeffekt auf Aussagekraft und Relevanz der gesammelten Daten auswirken).

Die QoC ist genau dann ausreichend, wenn die gesammelten Informationen es ermöglichen, sicher zu anderen möglichen Kontextklassen innerhalb der Kontextdimension zu differenzieren. Dies ist jedoch sowohl stark von der Komplexität des Konzepts, als auch davon abhängig, ob diejenigen Merkmale als Informationen vorliegen, welche ausschlaggebend für eine Differenzierung zu anderen Resultaten sind.

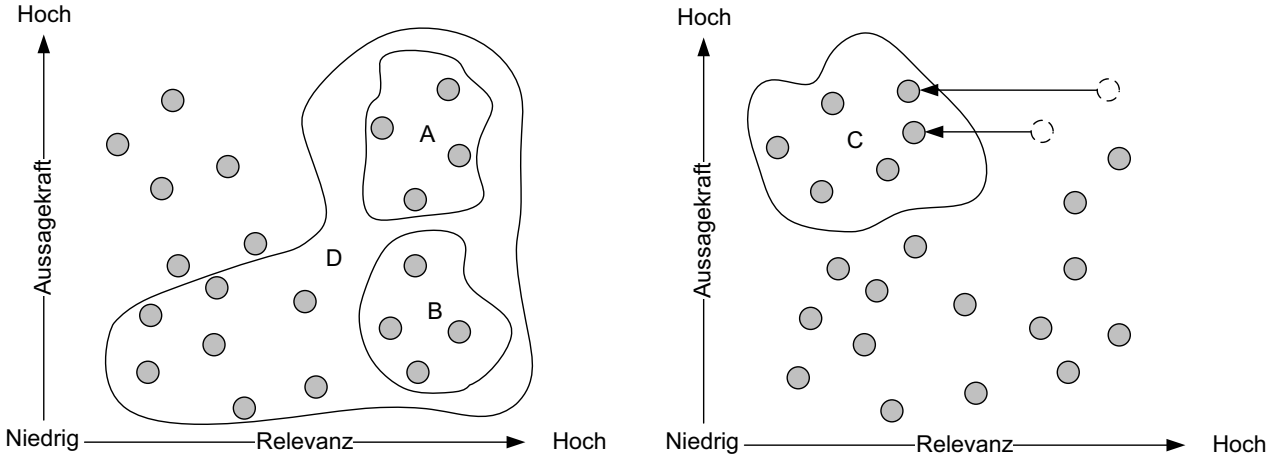


Abbildung 3.6: Qualität des Kontextes

In Abbildung 3.6 wird das Problem beispielhaft beschrieben. Die Informationen werden als Punkte und das Ergebnis der Suche als Wolke dargestellt. Die Informationen sind in der Abbildung in zwei Dimensionen angeordnet:

- **Relevanz:** Auf der horizontalen Ebene wird zwischen der Relevanz von Informationen unterschieden. Als Basis für die Relevanz einer Information steht deren Korrelation mit dem Ergebnis. Die Aussagekraft bestimmt sich jedoch auch durch die Anwendbarkeit auf das Modell, welches die gesammelten Informationen auswertet. Eine Information ist in der Regel vergleichbar mit einem Indiz. Es gibt schwache und starke Indizien. Reichen die Indizien in ihrer Summe aus, um mit dem Modell eine sichere Entscheidung zu treffen, war die QoC ausreichend.
  Wenn Informationen herangezogen werden, die keinen Bezug zu dem Konzept des Nutzers haben, können falsche Entscheidungen getroffen werden. Aufgabe der Suche ist es, potentiell relevante Sensoren mit einem Bezug zu dem Kontextobjekt zu bestimmen.

- **Aussagekraft:** Die vertikale Dimension beschreibt die Aussagekraft einer Information. Nicht jede Information kann direkt auf ein Modell angewandt werden oder sie erzielt bedingt durch ihre Darstellung nur geringe Aussagekraft innerhalb des Modells. Manche Informationen müssen geeignet dargestellt (z.B. Verlauf über die Zeit) oder vorverarbeitet (z.B. Objekterkennung auf Bilddaten oder Stichwortsuche in Texten) werden, um sinnvoll ausgewertet werden zu können. Durch Vorverarbeitung der Sensordaten (siehe Abschnitt 5.2.5) kann die Aussagekraft von Sensoren verbessert werden.

Eine irrelevante Information *sollte* im Bezug auf das Modell keine hohe Relevanz erreichen. Da Informationen im Modell abhängig von ihrem (semantischen) Typ verarbeitet werden, kann eine *suboptimal* durchgeführte Suche jedoch dazu führen, dass irrelevante Sensoren anstelle von vergleichbaren Typen an dem Modell angewandt werden, welche in dem Modell eine hohe Relevanz haben.

Eine QoC-Anforderung an eine Suche könnte sein, möglichst aussagekräftige *und* relevante Sensoren zu identifizieren und zu erfassen. Die Suche A in dem linken Teil der Darstellung würde diesem Ziel entsprechen. Die Suche B hingegen würde zwar relevante Sensoren finden, deren Aussagekraft würde jedoch eventuell nicht ausreichen, um mit dem Modell eine sichere Entscheidung zu treffen. In der rechten Darstellung ist die Suche C dargestellt. Suche C erfasst Informationen über ein anderes Kontextobjekt als Suche A. Daher werden in diesem Beispiel zwei Sensoren, die für die Suche A relevant waren, für die Suche C irrelevant. Bei der Suche C würden Informationen zur Auswertung des Modells genutzt werden, welche nicht im Bezug zu dem Kontextobjekt stehen. Folglich wären nicht zutreffende Ergebnisse zu erwarten. Bei der Suche D würden zwar alle wesentlichen Sensoren gefunden werden, es würden jedoch auch irrelevante Sensoren zur Auswertung herangezogen werden, welche (vergleichbar mit der Suche C) das Ergebnis negativ beeinflussen könnten.

Bei einer adaptiven, überwachten Auswertung (siehe Abschnitt 5.1.1) können Informationen durch Feedback und Modelladaption an Relevanz gewinnen. Innerhalb einer *Trainingsphase* kann eine Suche neben der Auswertung auch gegebenenfalls eine Modelladaption zum Ziel haben. Eine solche Suche hätte als eine QoI-Anforderung, möglichst viele der relevanten Informationen zu erhalten. Eine Kombination der Suchergebnisse von Suche A und B würde diesem Ziel entsprechen. Die Bestimmung der Relevanz einzelner Informationen, kann durch eine Analyse der Korrelation zwischen den Informationen und dem Verhalten des Nutzers (siehe Abschnitt 5.3.1) oder durch genetische Ansätze wie in [Zim06] beschrieben erfolgen.

3.4.4 Qualität des Modells

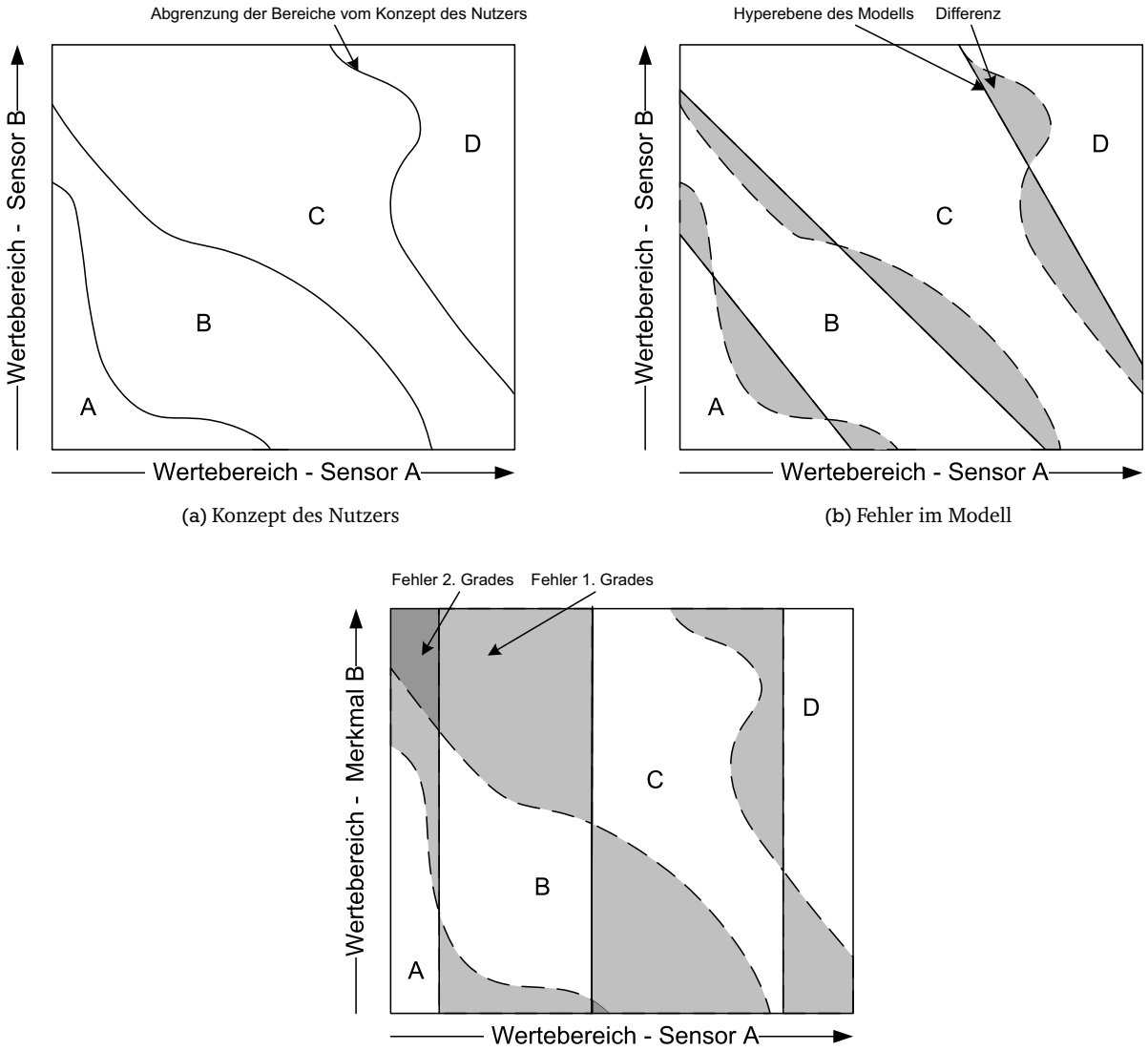
Zur Entscheidungsfindung werden die gesammelten Informationen ausgewertet. Innerhalb des Prozesses zur Entscheidungsfindung wird ein Modell zur Auswertung genutzt. Im Idealfall entspricht dieses Modell genau den Anforderungen des Nutzers, beschreibt also genau dasselbe Entscheidungsmuster (Konzept), wie der Nutzer es hat:

$$m() = h()$$

In Abbildung 3.7 a) wird das Problem beispielhaft beschrieben. Gegeben sei ein Nutzerkonzept, welches auf zwei Merkmalen basiert, welche durch die Sensoren A und B erfasst werden. In dem Beispiel unterscheidet das Konzept des Nutzers anhand der Zustände dieser Merkmale (dargestellt durch die gestrichelte Linie) exklusiv zwischen den Kontextklassen A, B, C und D.

Ein Modell kann das Nutzerkonzept nur näherungsweise beschreiben. Je mehr sich das Nutzerkonzept mit dem Modell deckt, desto höher ist die Qualität des Modells (QoM). Zum einen hängt die QoM von der Qualität der Informationen und dem Feedback ab, welches zur Erstellung des Modells genutzt wird. Zum anderen basiert die QoM jedoch auch auf den Darstellung des Modells und den damit verbundenen Möglichkeiten, das Nutzerkonzept zu beschreiben. Ein Modell welches beispielsweise auf der Basis linearer Regression arbeitet, würde das Modell durch Hyperebenen beschreiben, mit denen es den Wertebereich der Sensoren unterteilt, wie es in der Abbildung 3.7 b) durch die durchgehende Linie dargestellt wird. Bei der Generierung des Modells würden die Fehler (grau dargestellten Bereiche) minimiert werden. die Mittel die dem Modell zur Verfügung stehen, würden jedoch auch im optimalen Fall keine vollständige Abdeckung des Nutzerkonzeptes erlauben.

Ein Modell kann jedoch nur mit den Merkmalen arbeiten, welche durch Informationsquellen erfasst, über die Suche gefunden und in geeigneter Form dargestellt wurden. Somit steht dem Modell zur Entscheidungsfindung in der Regel nur eine Untermenge der Merkmale zur Verfügung, welche bei der Auswertung des Nutzerkonzeptes angewendet werden.



(a) Konzept des Nutzers

(b) Fehler im Modell

(c) Versteckte Merkmale

Abbildung 3.7: Qualität des Modells

In der Grafik 3.7 c) wird das Problem der *versteckten* Merkmale im Bezug auf das Modell dargestellt. Angenommen ein Nutzerkonzept beruht auf zwei Merkmalen, von denen nur eines über einen Sensor erfasst wird. Ein Verfahren zur Erstellung eines Modells wie aus dem vorherigen Beispiel würde zwar anhand der gesammelten Informationen des Sensors A versuchen eine möglichst gute Näherung an das Konzept des Nutzers zu finden. Da jedoch weder zur Generierung des Modells noch zur Auswertung des Modells Informationen bezüglich des Merkmals B vorliegen, entstehen größere Abweichungen des Modells zu dem Konzept des Nutzers. In der Darstellung sind Fehler erster Ordnung (*Abstand* zwischen Resultat und Nutzerkonzept) durch hellgraue Flächen dargestellt, sowie Fehler zweiter Ordnung durch dunkelgraue.

3.4.5 Qualität des Feedbacks

Wie im nachfolgenden Kapitel genauer erläutert werden wird, wird Feedback benötigt, um das Modell zur Entscheidungsfindung an das Konzept des Nutzers anzupassen. Ein Nutzer kann mittels Feedback das Modell verändern. Die Menge und Qualität des erhaltenen Feedbacks (siehe Abschnitt 3.2.4), kann die Qualität des Modells maßgeblich beeinflussen.

Für die Qualität des Feedbacks (QoF) sind hierbei mehrere Faktoren entscheidend:

- **Komplexität des Nutzerkonzeptes:** Je komplexer das Nutzerkonzept, umso mehr *aussagekräftiges* Feedback wird benötigt.
- **Aussagekraft:** Feedback ist genau dann aussagekräftig, wenn es ermöglicht, zwischen verschiedenen Ergebnismengen innerhalb des Nutzerkonzeptes zu differenzieren.
- **Korrektheit:** Feedback, welches nicht dem Nutzerkonzept entspricht, liefert falsche Aussagen.

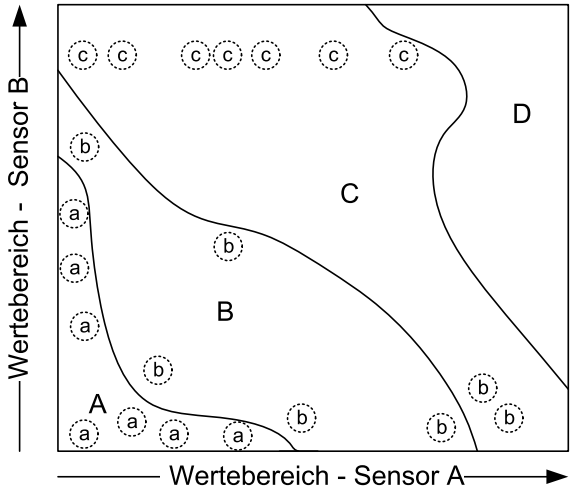


Abbildung 3.8: Qualität des Feedbacks

Diese Faktoren werden in Abbildung 3.8 erläutert. In dem Beispiel wird das Nutzerkonzept aus dem vorherigen Abschnitt 3.4.4 aufgegriffen. Die Punkte in der Darstellung entsprechen dem Feedback des Nutzers, wobei die Position eines Punktes den zum Feedback gesammelten Informationen entspricht und der Bezeichner des Punktes dem durch das Feedback übermittelten und vom Nutzer gewünschten Zustand. Der Raum, welcher die Kontextklasse A des Nutzerkonzeptes beschreibt wird gut durch Feedback abgedeckt (hohes QoF). Das Feedback, welches die Kontextklasse B beschreiben soll, verteilt sich zwar gut über den Raum, es existieren jedoch einzelne Elemente, welche nicht dem Nutzerkonzept entsprechen (unten rechts dargestellt). Solche (eventuell versehentlich getätigten) Falsch-Aussagen reduzieren die Qualität des Feedbacks und können das resultierende Modell negativ beeinflussen.

Die Komplexität des dargestellten Nutzerkonzeptes benötigt Feedback über beide Sensoren, um die Kontextklassen umfassend zu beschreiben. Fällt ein Sensor weg oder wird der Raum, welcher von dem Zustand des Nutzerkonzeptes beschrieben wird, nicht durch Feedback abgedeckt (wie für den Zustand C dargestellt), ist die Qualität des Feedbacks reduziert. Für die Beschreibung des Raumes von Zustand C reicht das gegebene Feedback nicht aus. Zu dem Zustand D wurde kein Feedback gegeben. In einem solchen Fall ist die Bestimmung dieses Zustandes (über ein, durch das Feedback generierte Modell) nicht möglich.

3.4.6 Qualität der Entscheidung

Die Qualität der Entscheidung (QoD) entspricht der Anzahl der korrekten Entscheidungen des Systems im Bezug auf das Konzept des Nutzers. Eine zutreffende Entscheidung liegt genau dann vor, wenn das Ergebnis der Auswertung des Modells $m()$ gleich ist mit der Auswertung des Konzeptes des Nutzers $h()$. Eine korrekte Entscheidung liegt genau dann vor, wenn $h(M) = m(f(M))$. Als Funktion $f()$ werden hierbei alle Effekte bezeichnet, welche Einfluss auf die Menge der Informationen haben, allen voran die Umsetzung der Suche.

3.4.7 Abhängigkeiten der Qualitätsmerkmale

Die Qualitätsmerkmale weisen Abhängigkeiten auf. QoC beschreibt die Aussagekraft der gesammelten Informationen. Die Menge der gesammelten Informationen ist abhängig von der Umsetzung der Suche, der Verfügbarkeit aussagekräftiger Sensoren und vom Aufwand, welcher in die Suche investiert wurde. Somit ist die QoC vor allem

abhängig von der bestehenden QoS innerhalb der Suche nach Informationen und der Abdeckung der Merkmale durch zur Verfügung stehenden Sensoren (QoI):

$$QoC \widehat{=} f_{Suche}(QoS, QoI)$$

Das Modell kann mittels Feedback angepasst werden. Feedback bezieht sich immer auf eine Entscheidung des Systems und damit auf die jeweilige Situation und die Informationen, welche die Grundlage für die Entscheidung bildeten. Somit ist die QoM abhängig von der QoF und der QoC:

$$QoM \widehat{=} f_{Adaption}(QoF, QoC)$$

Die Qualität der Entscheidung (QoD), welche von dem Gesamtsystem getroffen wird, ist abhängig von der Qualität des Modells und der Qualität der gesammelten Informationen:

$$QoD \widehat{=} f_{Evaluation}(QoM, QoC)$$

Das Gesamtresultat hängt von vielen Faktoren ab. Die Basis bildet die Umsetzung des Systems, also der Funktionen für die Suche, sowie der Adaption und der Evaluation des Modells. In der Phase der Nutzung des Systems können schließlich noch folgende Faktoren maßgeblich das Resultat beeinflussen:

$$QoD \widehat{=} f_{Evaluation}(f_{Adaption}(QoF, f_{Suche}(QoS, QoI)), f_{Suche}(QoS, QoI))$$

Ziel dieser Arbeit ist es, unter den gegebenen Faktoren, eine möglichst hohe QoD zu erzielen.

3.5 Fazit

In diesem Kapitel wurde das Konzept des Gesamtsystems erläutert. Dieses Grundkonzept sieht eine Aufteilung des Problems in eine Informationsgewinnung und eine Informationsauswertung vor. Diese Komponenten werden in den folgenden Kapiteln detailliert behandelt, wobei auf den Entscheidungen dieses Kapitels für die Architektur aufgebaut wird. Dies beinhaltet eine dienstorientierte Architektur, sowohl für die Sensoren, als auch für die Komponenten, welche die Informationen verarbeiten. Eine plattformübergreifende Form der Kommunikation über Konnektoren zwischen den Diensten bildet eine Middleware. Diese Middleware erlaubt eine transparente Anbindung von neuen Sensoren und weiteren Diensten in das Gesamtsystem. Die Auffindung und Adressierung neuer Sensoren und Dienste wird über Mechanismen innerhalb der Konnektoren abgedeckt. Die Integration von unbekannten Sensoren in den Gesamtprozess wird über Beschreibungen der Schnittstellen und der Semantik der Sensoren gewährleistet.

Des Weiteren wurden die Faktoren für die Qualität des Gesamtsystems aufgezeigt. Um letztendlich zu guten Entscheidungen im Bezug auf die Wahl der geeigneten Verarbeitung von Kommunikationsanfragen zu gelangen, müssen eine Reihe von Einflüssen auf die Qualität der Entscheidung berücksichtigt werden. Hierzu zählen vor allem die Abdeckung der Merkmale des Nutzerkonzepts durch Sensoren, die Erfassung relevanter Sensoren durch die Suche, die Leistung des Verfahrens zur Auswertung der Daten und die Aussagekraft des Feedbacks des Nutzers.

4 Informationsgewinnung

Ein Schwerpunkt dieser Arbeit liegt im Bereich der Informationsgewinnung. Die Aufgaben innerhalb dieses Schwerpunktes beziehen sich auf die Erkennung der zur Verfügung stehenden Sensoren, Erfassung und Nutzung der Beschreibungen der Sensoren und schließlich der Aggregation von Informationen.

Vorgehen

Zu Beginn werden in Abschnitt 4.1 die Voraussetzungen für die Informationsgewinnung von der allgemeinen Anforderungsanalyse aus Abschnitt 2.3 abgeleitet. Eine grundlegende Herausforderung stellt sich in der plattformübergreifenden Kommunikation. Hier gilt es, eine möglichst flexible und ressourcensparende Form der Kommunikation zu finden, um sowohl komplexere Methodenaufrufe, als auch Abfragen von Informationsquellen auf eingebetteten Systemen zu ermöglichen. Diese Aspekte werden in Abschnitt 4.2 behandelt.

Das Ziel, ein offenes System zur Informationsgewinnung aufzubauen, stellt eine weitere Herausforderung dar. Zum einen muss eine allgemeine Schnittstelle zur Auffindung und Anbindung von Sensoren angeboten werden, zum anderen muss diese Schnittstelle weitere Funktionalitäten zur Abdeckung der sensorspezifischen Fähigkeiten erweitert werden können. Die für diese Anforderungen entwickelte Architektur wird in Abschnitt 4.3 erläutert.

Erweiterbare Schnittstellen machen eine Beschreibung der Schnittstelle erforderlich. Da eine der Herausforderungen in der Minimierung der Gesamtdauer der Informationsgewinnung unter Maximierung der Qualität der gesammelten Informationen liegt, ist es notwendig, die Anfragen auf relevante Sensoren und wiederum relevante Methoden der Schnittstelle des Sensors zu reduzieren. Um im Voraus die Relevanz zu bestimmen ist eine semantische Beschreibung des Sensors und dessen Schnittstellen erforderlich. In welcher Form und in welchem Umfang eine solche Beschreibung angewandt wird, ist in Abschnitt 4.4 beschrieben.

Die Informationsgewinnung soll ermöglichen, dass einerseits dynamisch Sensoren angebunden werden und andererseits verschiedene Dienste flexibel und effizient Anfragen über die aktuell verfügbaren Informationsquellen stellen können. Somit ist zum Zeitpunkt einer Anfrage eine Liste der verfügbaren Sensoren und deren Fähigkeiten notwendig. Würde diese Aufgabe in die einzelnen anfragenden Dienste ausgelagert, entstünde ein erhöhter Aufwand im Vergleich zu einer dedizierten Instanz, welche diese Aufgabe übernimmt. Diese Aufgabe wird vom *Sensorverzeichnis* übernommen. Zur Optimierung der Informationsgewinnung können vom Sensorverzeichnis weitere Funktionalitäten wie Vorbereitung der Suche, Zwischenspeicherung der Daten, Datenvorhaltung und Datenaufbereitung angeboten werden. Die detaillierte Architektur des Sensorverzeichnisses und die Möglichkeiten zur Optimierung der Informationsgewinnung werden in Abschnitt 4.5 dargestellt.

Eine Informationsgewinnung soll möglichst umfassend alle Informationen sammeln, welche sich im Rahmen der Kontextdimension auf ein bestimmtes Kontextobjekt beziehen. Einige Informationsquellen beziehen sich direkt auf ein Kontextobjekt. Beispielsweise können Informationsquellen, welche sich auf dem Mobiltelefon eines Nutzers befinden, direkt auf den jeweiligen Nutzer bezogen werden. Manche Beziehungen zwischen einer Informationsquelle und einem Kontextobjekt lassen sich jedoch nicht direkt bestimmen. So werden beispielsweise Informationsquellen welche sich auf einen Raum beziehen lassen (wie z.B. ein Bewegungsmelder oder die Informationsquellen benachbarter Personen) erst dann relevant, wenn der Standort eines Nutzers bekannt ist. Die Informationsgewinnung muss daher in der Lage sein, eine Suche über diese semantischen Strukturen hinweg zu ermöglichen. Diese Herausforderung wird in Abschnitt 4.6 behandelt.

Die Laufzeit der Suche hängt maßgeblich von der Menge der Sensoren, welche im Verlauf der Suche angefragt werden müssen ab. In Abschnitt 4.7 werden Ansätze behandelt, die es erlauben die Menge anzufragender Sensoren zu reduzieren ohne die Qualität der Informationen zu beeinträchtigen.

4.1 Ziele und Komponenten

Die Anforderungen für diesen Bereich, abgeleitet von den allgemeinen Rahmenbedingungen, ergeben sich wie folgt:

1. **Dynamische Menge von Sensoren:** Die Menge aktuell zur Verfügung stehender Sensoren kann starken Schwankungen unterworfen sein. Damit muss das System unter anderem robust gegenüber Ausfall von Informationsquellen und skalierbar in der Anzahl adressierbarer Sensoren sein.

2. **Offenes System:** Neue, unbekannte Sensoren sollen jederzeit angebunden und in die Informationsgewinnung integriert werden. Das System muss ohne vordefinierte Typen von Sensoren agieren können.
3. **Heterogene Sensoren:** Es sollen Sensoren aller Art integriert werden können, unabhängig von Plattform, Betriebssystem, Programmiersprache oder Kommunikationsform.
4. **Kosten, Aufwand, Mehrwert:** Zur Vergrößerung des Anwendungsbereiches und zur Erhöhung des Mehrwertes durch das Verringern von Kosten sollen auch bestehenden Standardkomponenten als Informationsquellen genutzt werden können. Hinzu kommt das Ziel der Reduzierung des Aufwandes zur Integration von Standardkomponenten, sowie neuer unbekannter Informationsquellen.
5. **Dienstgüte:** Auf der einen Seite bringen heterogene Sensoren eine Vielzahl von unterschiedlichen Eigenschaften im Rahmen von Qualitätsmerkmalen und -anforderungen mit sich. Auf der anderen Seite muss es möglich sein, unter Einhaltung von Qualitätsanforderungen seitens der Suche auf diese Informationsquellen zurückzugreifen. Hierzu gehört auch die Skalierbarkeit in der Anzahl der Anfragen beziehungsweise anfragenden Systeme.
6. **Informationsgüte:** Wie in Abschnitt 3.4.2 beschrieben, besteht eine der Hauptaufgaben dieses Bereiches darin relevante Informationsquellen aus einer Vielzahl von möglichen Informationsquellen zu identifizieren und unter Einhaltung von Dienstgüte-Anforderungen zu erfassen.

4.1.1 Überblick der Komponenten

Das folgende Bild 4.1 gibt einen Überblick über Komponenten welche innerhalb dieses Abschnittes behandelt werden. Die einzelnen Komponenten arbeiten auf unterschiedlichen Ebenen. Die Anordnung der Komponenten beschreibt die Abhängigkeit der Komponenten. Die Abhängigkeiten zwischen den Komponenten sind jeweils durch vertikale Zuordnungen angedeutet.

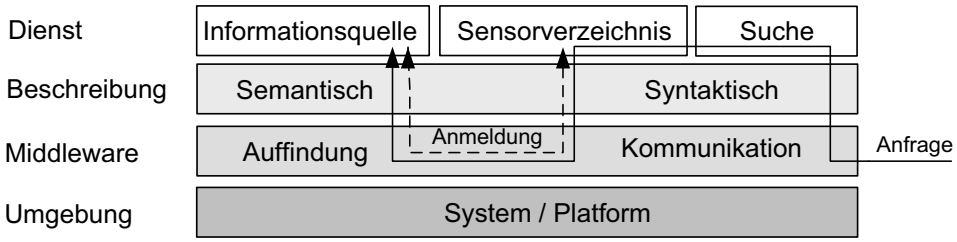


Abbildung 4.1: Überblick über die Komponenten

In diesem Kapitel werden die in der Grafik dargestellten Komponenten wie folgt behandelt:

- **Informationsquellen:** Aus den Rahmenbedingungen geht hervor, offen für Informationsquellen aller Art zu sein. Daher werden Informationsquellen als solche zunächst nur abstrakt behandelt. Eine Informationsquelle kann jede beliebige Komponente sein, an der Informationen abgegriffen werden können. Welche Informationsquellen konkret umgesetzt wurden, wird in Kapitel 6 genauer erläutert. Zur Abfrage der Beschreibungselemente eines Sensors, erfolgt die Anbindung über eine einheitliche Spezifikation für diese Grundfunktionalitäten der Sensor-Schnittstellen. Diese Spezifikation wird in Abschnitt 4.3 behandelt.
- **Sensorverzeichnis:** Zur Einhaltung von QoS-Anforderungen sollten zur Laufzeit einer Suche, Informationen über die verfügbaren Sensoren bereits vorliegen. Das Sensorverzeichnis bietet eine Schnittstelle zur Registrierung aufgefundener Sensoren (dargestellt als gestrichelter Pfeil in Abbildung 4.1). Das Sensorverzeichnis zählt daher zu den zentralen Elementen dieser Arbeit und wird in Abschnitt 4.5 eingehend behandelt.
- **Suche:** Die Suche nach Informationen ist der Dienst, welcher von der Informationsgewinnung nach außen angeboten wird. Über eine Schnittstelle können Anfragen an die Suche gestellt werden. Eine Suche wird dann mittels der Informationen des Sensorverzeichnisses durchgeführt und die entsprechenden Informationsquellen werden abgefragt (dargestellt als Pfeil mit durchgehender Linie in Abbildung 4.1). Diese Suche und deren Schnittstelle werden in Abschnitt 4.6 beschrieben.
- **Beschreibung:** Über Beschreibungselemente ist es möglich auf unbekannte Schnittstellen eines Dienstes zuzugreifen. Eine weitere Aufgabe der Beschreibung, welche für diese Arbeit eine hohe Relevanz hat, ist die Beschreibung der Bedeutung eines Sensors mittels semantischer Beschreibungselemente. Dieser Aufgabenbereich wird in Abschnitt 4.4 detailliert erläutert.

- **Kommunikation:** Zur Kommunikation der verschiedenen Dienste über unterschiedliche Plattformen hinweg, ist es notwendig, eine gemeinsame Ebene der Kommunikation zu ermöglichen. Neben der Kommunikation ist die Auffindung der Dienste über Plattform und Netzwerkgrenzen hinweg eine wichtige Aufgabe. Die Ansätze, welche hierfür in Frage kommen, eine Auswahl möglicher Ansätze und deren Vergleich werden in Abschnitt 4.2 beschrieben.

4.2 Kommunikation

Wie in Abbildung 4.1 dargestellt wird eine Schicht zur Kommunikation genutzt, welche von der darunter liegenden Plattform abstrahiert und somit eine transparente Kommunikation zwischen Diensten auf unterschiedlichen Plattformen ermöglicht. Sogenannte *Konnektoren* bieten auf dieser Ebene eine einheitliche Schnittstelle zur Kommunikation.

Je nach Bedarf werden unterschiedliche Konnektoren benötigt. In diesem Abschnitt wird der Bereich der Kommunikation über Konnektoren genauer erläutert und es wird analysiert, welche Technologien zur Umsetzung der Konnektoren in Frage kommen.

4.2.1 Ablauf der Kommunikation über Konnektoren

Für diese Arbeit wurde der Ansatz der *Konnektoren* entwickelt und umgesetzt, welcher es ermöglicht entfernte Dienste unabhängig von der genutzten Technologie aufzurufen. Hierzu wird eine einheitliche Schnittstelle angeboten, welche durch den Konnektor implementiert werden muss. Ein Konnektor muss dazu die folgenden grundlegenden Funktionalitäten erbringen:

- Lokalisation von Diensten.
- Aufruf von Diensten.
- Datenrepräsentation und -transport.

Wie in den vorangestellten Abschnitten dargestellt wurde, existiert eine Reihe von Technologien, welche einzelne dieser Funktionen erfüllen. Zur Umsetzung eines Konnektors mit vollständigem Funktionsumfang bestehen daher folgende Varianten:

- Mehrere Technologien aus den jeweiligen Bereichen kombinieren.
- Ein Rahmenwerk nutzen, welche diese Technologien kombiniert.

Je nach Umsetzung eines Konnektors kann der Ablauf der Kommunikation im Detail unterschiedlich verlaufen. Im Allgemeinen lässt sich jedoch die Kommunikation durch den folgenden (vereinfachten) Ablauf abbilden:

1. Registrierung
Ein Dienst A soll für einen anderen Dienst B verfügbar gemacht werden. Hierzu bietet der entfernte Dienst A eine Schnittstelle (in der Regel ein *Interface* seines Programmcodes) nach außen an, indem er die Schnittstelle an den Konnektor anmeldet.

2. Suche
Der Dienst B, welcher auf den Dienst A zugreifen möchte, muss diesen zuerst auffinden beziehungsweise lokalisieren. Über einen Mechanismus zur Dienst-Lokalisation (im Anhang unter A.2) kann über Parameter (beispielsweise Beschreibung des Dienstes, URI oder Package-Bezeichner) ein verfügbarer und geeigneter Dienst bestimmt werden.

3. Bindung an Proxy / Stub
Wurde bei der Suche (mindestens) ein geeigneter Dienst bestimmt, kann ein Proxy-Objekt oder *Stub* erstellt werden. Hierzu ist entweder die Schnittstelle des Dienstes bekannt oder es muss eine Beschreibung der Schnittstelle des entfernten Dienstes verwendet werden (siehe Abschnitt 4.4 zur syntaktischen Beschreibung von Diensten). Ein solcher Stub besitzt dieselben Schnittstellen wie die entfernte Schnittstelle, kann jedoch wie eine lokale Klasse angesprochen werden.

4. Aufruf, Ergebnis

In den Schritten 4.1 und 4.2 werden alle Aufrufe auf den Stub über den Konnektor zum entfernten Dienst geleitet, um dort die Schnittstelle des entfernten Dienstes aufzurufen. Hierzu werden alle Parameter eines Aufrufes in einer Nachricht eingebettet (engl. *Marshalling*) und über das unterliegende Netzwerk zur Plattform des entfernten Dienstes transportiert. Das Ergebnis des Aufrufs wird analog zum Aufruf in den Schritten 4.3 und 4.4 zurück an den aufrufenden Dienst geleitet.

Dieses sehr generisch gehaltene Schema ist als Ablaufdiagramm in Abbildung 4.2 dargestellt. Die ersten drei Schritte bilden die Initialisierungsphase, bei der der Dienst mit dem Konnektor interagieren muss. Durch die Bindung des Stubs an ein lokales (Proxy-)Objekt erhält die Anwendung die Möglichkeit, transparent auf Schnittstellen der entfernten Dienste zurückzugreifen. Das bedeutet in der Phase der Nutzung des vom Konnektor generierten Objektes unterscheidet sich der Aufruf von einem lokalen Methodenaufruf höchstens noch durch die entstehende Verzögerung und in der eventuell notwendigen Behandlung von Fehlerzuständen, welche durch Probleme in der Kommunikation hervorgerufen werden können.

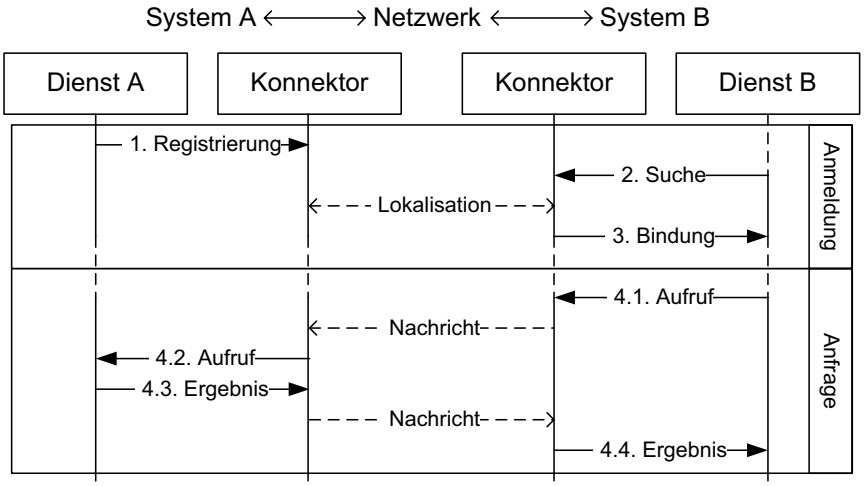


Abbildung 4.2: Konnektor: Generisches Ablaufschema für die Auffindung und Nutzung von Diensten

4.2.2 Basistechnologien

Wie in Abschnitt 4.2.1 beschrieben, besteht die Aufgabe der Konnektoren in der Umsetzung der Kommunikation zwischen Diensten. Diese Aufgabe lässt sich wiederum in die Bereiche Datenrepräsentation, Datentransport, Anfragespezifikation, sowie Dienst-Lokalisation aufteilen.

Je nach zur Verfügung stehenden Konnektoren können unterschiedliche Plattformen und damit zusätzliche Informationsquellen an das System angebunden werden.

Die Wahl der Technologie im Bereich Transport und Datenrepräsentation kann die Geschwindigkeit bei der Anfrage einer Informationsquelle maßgeblich beeinflussen. Die Schwierigkeit in diesem Bereich liegt vor allem darin unter diesen Bedingungen einen möglichst effizienten, plattformunabhängigen Ansatz zu wählen:

- Plattformunabhängige Ansätze nutzen in der Regel ein Meta-Format zur Beschreibung komplexer Objekte. Die Umsetzung hin zu diesem Meta-Format, als auch die Übersetzung zurück erzeugen zusätzlichen Aufwand.
- Effiziente Ansätze nutzen häufig eine direkte Umwandlung des Objektes in einen serialisierten, binären Datenstrom. Dieser Datenstrom kann jedoch meist nur von Systemen eingelesen werden, welche mit Objekten des entsprechenden Typs umgehen können, was im Allgemeinen nur auf derselben Plattform möglich ist.

Technologien zur Spezifikation der Anfrage können teilweise nur auf bestimmten Plattformen eingesetzt werden und haben oftmals feste Abhängigkeiten zu anderen Technologien im Bereich Datenrepräsentation oder Transport. Die Tabelle 4.1 vergleicht die im Anhang in Abschnitt A.2 erläuterten Technologien im Bezug auf deren Einschränkungen auf bestimmte Plattformen, sowie deren Abhängigkeiten zu anderen Technologien.

Datenrepräsentation

Bei dem Transport der Nachrichten ist es erforderlich, alle für den entfernten Aufruf benötigten Daten und Parameter in Nachrichten gekapselt zu transportieren. Da ein Aufruf möglicherweise auch den Transport komplexer Objekte (deren Struktur vorab unbekannt ist) erforderlich macht, muss eine geeignete Form der Repräsentation dieser Daten gewählt werden. In den meisten Ansätzen kann man zwischen den folgenden Formen der Repräsentation von Daten differenzieren:

- **Objekt:** Systemspezifische Serialisierung von Objekten. Diese Technik wird beispielsweise in Ansätzen wie Remote Method Invocation (RMI) verwendet (welcher im folgenden Teil dieses Abschnittes erläutert wird).
- **Binärcodiert:** Systemübergreifende Binärcodierung (wie z.B. Abstract Syntax Notation One (ASN.1) oder wie in Common Object Request Broker Architecture (CORBA) (siehe Abschnitt 4.2.3) durch Interface Definition Language (IDL) in Kombination mit Common Data Representation (CDR)).
- **Extensible Markup Language (XML):** Strukturierte textuelle Darstellung.

Details zu den einzelnen Verfahren und weitere Ansätze finden sich im Anhang in Abschnitt A.2.

Datentransport

Bei dem Bereich des Transports der Daten geht es um die Wahl der Technologie zur Kommunikation zwischen den Diensten. Die Kommunikation zwischen den Diensten kann im Allgemeinen durch den Aufruf einer oder mehrerer Methoden auf einem entfernten Dienst beschrieben werden (engl. Remote Procedurce Call (RPC)). Im Allgemeinen wird dabei auf folgende Ansätze zurückgegriffen:

- **Socket:** Transport von Rohdaten über ein Netzwerk.
- **Hypertext Transfer Protocol (HTTP):** Protokoll zum Austausch von Daten über Sockets.
- **Nachrichten:** Transport von (meist asynchronen) Anfragen über mehrere Systeme hinweg, wie beispielsweise bei Java Message Service (JMS), Session Initiation Protocol (SIP) oder Extensible Messaging and Presence Protocol (XMPP)).

Diese und weitere Ansätze werden im Anhang in Abschnitt A.2 detailliert erläutert.

Spezifikation von Anfragen

Die Anfrage eines entfernten Dienstes erfordert in der Regel mindestens die Nennung der aufzurufenden Methode und die Bereitstellung der zum Aufruf benötigten Parameter. Hierzu wird eine gemeinsame Spezifikation für die Art der Anfrage und das Nachrichtenformat auf Seiten des anfragenden und des angefragten Systems erforderlich. Für das Absetzen einer Anfrage an einen entfernten Dienst existiert eine Reihe von Ansätzen, welche im Rahmen dieser Arbeit betrachtet wurden:

- **Representational State Transfer (REST):** Einfaches Konzept um auf Basis von HTTP Dienste aufzurufen.
- **Internet Inter-ORB Protocol (IIOP):** Im Rahmen von CORBA entwickeltes Protokoll zum Aufrufen von Diensten.
- **Remote Method Invocation (RMI):** Von Java bereitgestelltes System zum Aufruf anderer, entfernter Java Methoden.
- **Remote-OSGI (R-OSGI):** Eine Spezifikation zur systemübergreifenden Kommunikation zwischen OSGI Diensten[1].
- **XML-RPC:** Auf XML basierende Spezifikation zum Aufruf entfernter Dienste.
- **Jabber-RPC:** Ein Ansatz, um über das IMS-System Jabber[2] entfernte Dienste zu adressieren.
- **SOAP:** Auf XML basierender Standard[3] zum Aufruf entfernter Dienste und zur Repräsentation der Daten.
- **Simple Network Management Protocol (SNMP):** Protokoll zur Abfrage und Steuerung von Geräten im Netzwerk.

Eine detaillierte Beschreibung dieser Ansätze befindet sich im Anhang in Abschnitt A.2. Die Ansätze zur Spezifikation von Anfragen basieren auf den zuvor genannten Technologien zur Datenrepräsentation und zum Transport von Daten. Die Tabelle 4.1 zeigt einen Überblick der betrachteten Ansätze und vergleicht deren Abhängigkeiten.

[1] `http://www.osgi.org/`
[2] `http://www.jabber.org/`
[3] `http://www.w3.org/TR/soap/`

	REST	IIOP	RMI	R-OSGI	XML-RPC	Jabber-RPC	SOAP	SNMP
Plattformen	*	*	JAVA	JAVA, OSGI	*	*	*	*
Transport	(HTTP)	Socket	Socket	Socket	(HTTP)	XMPP	(HTTP)	Socket
Repräsentation	(XML, SOAP)	IDL, CDR	Objekt	Objekt	XML(+)	XML(+)	XML	ASN.1, BER

* uneingeschränkte Anwendbarkeit einer Technologie im Bezug auf unterschiedliche Plattformen.
(...) die jeweilige Spezifikation macht keine Vorgaben in diesem Bereich.
Oft kommt jedoch die Technologie zum Einsatz kommt, welche in den Klammern genannt wird.
(+) weist auf eine Beschränkung der Datentypen auf primitive Typen hin.

Tabelle 4.1: Technologien für die Spezifikation von Anfragen

Lokalisierung von Diensten

Es existieren eine Reihe von Ansätzen zur Lokalisierung von Diensten:

- **Broadcast:** Einfache Ansätze lassen die Dienste regelmäßig Broadcast-Nachrichten schicken, um von anderen erkannt zu werden.
- **Zentralisiert:** Viele Ansätze nutzen einen zentralen Instanzen (wie beispielsweise *Registrare*, *Broker* oder *Namensdienste*) zur Lokalisation von Diensten. Diese müssen systemweit bekannt sein und dienen der Registrierung und Abfrage von verfügbaren Diensten.
- **Protokoll:** Viele der Ansätze, welche auf Broadcasts oder zentralen Verzeichnisdiensten basieren, sind nur für die Nutzung des Ansatzes innerhalb des jeweiligen Systems konzipiert (wie beispielsweise PDP, welches in JXTA[4] Verwendung findet). Es existieren jedoch Protokolle zur systemübergreifenden Lokalisierung von Diensten. Hierzu zählen das Service Location Protocol (SLP) oder das Simple Service Discovery Protocol (SSDP). Diese Protokolle bieten meist die Möglichkeit mit oder ohne Verzeichnisdienst zu arbeiten. Manche von Ihnen greifen im Bedarfsfall auf den Broadcast Ansatz zurück, um den zentralen Verzeichnisdienst zu lokalisieren oder um ohne Verzeichnisdienst genutzt werden zu können. Eine Protokoll-Variante, welche in dem Peer-to-Peer (P2P) System JXTA zum Einsatz kommt ist das Peer Discovery Protocol (PDP).

Zentralisierte Ansätze benötigen meist ein dediziertes System im Netzwerk, welches den Dienst erbringt und dessen Adresse systemweit bekannt ist, beziehungsweise bei den teilnehmenden Systemen manuell eingestellt werden muss. Aus den Anforderungen fünf und sechs aus Abschnitt 2.3 ist abzuleiten, dass solche manuellen Eingriffe oder notwendiges Vorwissen vermieden werden sollten. In diesem Abschnitt werden daher Technologien betrachtet, welche die Lokalisation von Diensten ohne Vorwissen ermöglichen.

Protokolle zur Lokalisation von Diensten ermöglichen meist eine Beschreibung der Dienste, sowie Suchmechanismen, welche Dienste mit bestimmten Beschreibungselementen filtern. Diese Verfahren zur Beschreibung von Diensten und den darauf basierenden Suchmechanismen sind jedoch in der Regel relativ einfach (beispielsweise basierend auf Schlüsselworten). Ein guter Vergleich existierender Varianten wird in der Arbeit von Bettstetter und Renner aufgezeigt [BR00a].

Details zu den betrachteten Protokollen zur Lokalisierung von Diensten finden sich im Anhang im Abschnitt A.2.

4.2.3 Überblick über Rahmenwerke zur Kommunikation

Ein Konnektor muss die Funktionalitäten erbringen, einen Dienst zu lokalisieren, eine Methode des Dienstes aufzurufen und den Transport der Daten zu gewährleisten.

Es existieren eine Reihe von Technologien, welche einzelne oder mehrere dieser Aspekte adressieren:

[4] https://jxta.dev.java.net/

	CORBA	JINI, JavaSp.	JADE	UPnP	WCF	J2EE	DPWS	Mundo-Core	JXTA
Plattformen	Java, C++	Java	* (JAVA)	*	* (.NET)	* (Java)	* (Java, C)	* (Java, C++)	*
Netzwerke	* (IP, BT)	IP	* (IP)	IP	IP	IP	IP	* (IP, BT)	IP, P2P
Anfrage	IIOP (RMI)	RMI	* (RMI, IIOP, HTTP)	SOAP-RPC	* (REST, SOAP-RPC, IPC)	* (SOAP)	SOAP	RMI-, SOAP-ähnlich	Socket
Lokalisation	Namens-dienst	Namens-dienst	* (Multi-cast)	SSDP	Namens-dienst	Namens-dienst	Multicast	Broadcast	PDP
Daten-repräsentation	IDL, CDR	Objekt	* (Objekt, IDL, CDR)	XML, SOAP	* (SOAP)	* (SOAP)	SOAP	Objekt, XML	XML

* Es bestehen für das jeweilige System keine Einschränkungen für den entsprechenden Bereich.
* (...) Es bestehen keine Einschränkungen, jedoch finden einzelne Technologien vorrangig Verwendung oder die Menge verfügbarer Implementierungen ist beschränkt.

Tabelle 4.2: Technologien zur Kommunikation zwischen Diensten

- **CORBA:** Plattformübergreifende, objektorientierte Middleware[5].
- **JINI, JavaSpaces:** Auf Java basierendes Rahmenwerk zur Programmierung verteilter Anwendungen oder im Falle von JavaSpaces zur verteilten Datenhaltung.
- **Java Agent DEvelpment Framework (JADE):** Agentenbasierte Middleware für Java.
- **Universal Plug and Play (UPnP):** Protokoll zur Auffindung, Abfrage und Steuerung von Geräten im Netz.
- **Windows Communication Foundation (WCF):** System zur Kommunikation zwischen Diensten, hauptsächlich im Umfeld von Microsoft .NET Anwendungen.
- **Java Platform, Enterprise Edition (J2EE):** Rahmenwerk zur Ausführung von Java Diensten und zur Kommunikation zwischen den Diensten.
- **Devices Profile for Web Services (DPWS):** Spezifikation für Netzdienste auf mobilen Endgeräten
- **MundoCore:** Leichtgewichtige Kommunikations-Middleware für Dienste in heterogenen Umgebungen.
- **JXTA:** Ein offenes, auf Java basierendes Peer-to-Peer System.

Eine detaillierte Beschreibung der einzelnen Rahmenwerke findet sich im Anhang in Abschnitt A.3. In Tabelle 4.2 werden die betrachteten Rahmenwerke und Technologien aufgezeigt und im Vergleich dargestellt. In der Tabelle werden die in den jeweiligen Rahmenwerken verwendeten Technologien hervorgehoben. Es werden sowohl die Plattformabhängigkeiten und die adressierten Arten von Netwerken dargestellt. Manche Systeme erlauben auch den Einsatz außerhalb von IP-basierten Netzen, beispielsweise innerhalb von Bluetooth (BT). Die Kategorien innerhalb der Tabelle beziehen sich unter anderem auf die zuvor aufgezeigten Bereiche Spezifikation der Anfrage, Lokalisation und Datenrepräsentation (siehe hierzu im Anhang in Abschnitt A.2).

4.2.4 Ergebnisse des Vergleichs

Für den Einsatz in dem angestrebten Szenario lassen sich folgende Aussagen aus der Betrachtung und aus dem Vergleich der verschiedenen Technologien ableiten:

[5] http://www.corba.org/

Anwendbare Technologien

Es existiert eine Vielzahl von Technologien, welche grundsätzlich angewendet werden können.

- Die dargestellten Rahmenwerke decken unterschiedliche Anwendungsgebiete ab.
- Rahmenwerke wie UPnP oder JavaSpaces sind von Konzept her geeignet, Daten wie beispielsweise Sensorwerte zu transportieren. Sie wurden jedoch ursprünglich für andere Anwendungsbereiche konzipiert. Eine wie in dieser Arbeit vorgesehene, dienstorientierte Kommunikation zu den Sensoren wäre in dem Rahmen möglicherweise aufwendig umzusetzen, nicht möglich oder mit unerwünschten Nebeneffekten verbunden.
- Rahmenwerke wie beispielsweise WCF oder J2EE sind relativ schwergewichtig. Eine direkte Nutzung eines solchen Rahmenwerkes ist ungeeignet für Plattformen mit eingeschränkten Ressourcen.
- Die Menge der verwendbaren Plattformen wird oft nicht durch Protokoll, sondern durch verfügbare Implementierungen eingeschränkt. Ansätze wie beispielsweise CORBA, MundoCore könnten prinzipiell auf einer Vielzahl sehr unterschiedlicher Plattformen zum Einsatz kommen. Es existieren jedoch teilweise nur einzelne ausgereifte Implementierungen für bestimmte Plattformen.
- Manche Ansätze wie beispielsweise CORBA oder JINI sind relativ alte Ansätze. Sie werden häufig in der Literatur genannt, werden jedoch in der Praxis nur noch selten angewendet, da mittlerweile weiter entwickelte Ansätze existieren, welche beispielsweise in Umgebungen wie .NET integriert worden sind und daher oft einfacher angewandt werden können.
- Viele Rahmenwerke nutzen einen eigenen Ansatz zur Lokalisierung von Diensten. Meist wird hierzu ein dedizierter Verzeichnisdienst oder Namensdienst eingesetzt. Zudem muss in vielen Fällen entweder der Namensdienst bekannt sein oder es wird zusätzliche Komponente zur Lokalisation unbekannter Dienste notwendig.
- Die Spezifikation DPWS sieht Funktionen vor, welche relativ gut zu den Anforderungen passen. Die Nutzung von SOAP bzw. XML zum Transport der Daten wirkt sich allerdings negativ auf den erzeugten Aufwand aus [Hil09].

Plattformübergreifende Anwendung

Viele der verwendeten Protokolle erlauben eine plattform- beziehungsweise rahmenwerksübergreifende Anwendung.

- Je nach Spezifikation der Anfrage und der Datenrepräsentation können unterschiedliche Rahmenwerke prinzipiell miteinander kommunizieren.
- Oft bestehen Unterschiede im Detail und es existieren viele unterschiedliche Spezifikationen für Anfragespezifikation und Datenrepräsentation. Die Verbindung verschiedener Rahmenwerke ist oft nicht trivial.
- Viele der Rahmenwerke setzen auf Basistechnologien wie SOAP oder RMI auf.
- Webservice Spezifikationen wie beispielsweise SOAP in Kombination mit Web Services Description Language (WSDL) sind relativ weit verbreitet und einen bieten einen rahmenwerksübergreifenden Ansatz.
- REST wird häufig als *kleinster gemeinsamer Nenner* angewendet. Es beinhaltet jedoch keine Funktionalitäten oder Vorgaben zur Repräsentation komplexer Daten.
- Nutzung und Einsatz großer Rahmenwerke (wie beispielsweise WCF oder J2EE) bietet meist eine Vielzahl von Funktionalitäten, welche über die Anforderungen der Kommunikation zwischen Diensten hinaus gehen. Auf der anderen Seite eignen sich diese teilweise recht schwergewichtigen Ansätze nicht für den Einsatz auf mobilen oder ressourcenbeschränkten Endgeräten.
- Zusätzlicher Aufwand entsteht häufig in der Bereitstellung oder Nutzung von Schnittstellenbeschreibungen. Gerade bei Ansätzen wie ASN.1, IDL, CDR, MundoCore werden hierzu zusätzliche Schritte benötigt.

4.3 Sensoren

Wie in Abschnitt 3.3 erläutert, wird für das Gesamtkonzept eine dienstorientierte Architektur angestrebt. Somit liegt es nahe, die Sensoren ebenfalls als (den anderen Diensten gleichgestellten) Dienst zu betrachten. Dies hätte zum Vorteil, dass eine Kommunikation zu den Sensoren auf gleicher Ebene wie die Kommunikation zwischen den übrigen Diensten erfolgen kann. Somit wäre der Zugriff auf die Sensoren prinzipiell von jedem Dienst innerhalb dieses Systems aus möglich. Die Möglichkeit auf die Sensoren direkt zuzugreifen vereinfacht die Umsetzung neuer Dienste unter Gewährleistung des Zugriffs auf die Sensoren.

In der Architektur, wie sie in Abbildung 4.3 dargestellt wird, kapselt ein Sensor mindestens eine Informationsquelle und erweitert sie um eine Schnittstelle zur Anbindung an das Rahmenwerk. Daher werden Informations-

quellen als solche zunächst nur abstrakt behandelt. Vielmehr geht es um die Architektur eines Sensor und eine Spezifikation der Schnittstelle.

Die Funktionalitäten der Schnittstelle kann in zwei Bereiche aufgeteilt werden:

- **Grundfunktionalitäten:** Diese dienen der Anmeldung, Registrierung und Anbindung eines Sensors. Diese Funktionen werden von allen Sensoren gleichermaßen benötigt.
- **Erweiterte Funktionalitäten:** Diese Elemente der Schnittstelle spiegeln die eigentlichen Methoden zur Abfrage der Informationsquelle wieder. Die Beschreibung bezieht sich auf den Sensor und die Methoden, die durch die Erfassung gegeben sind.

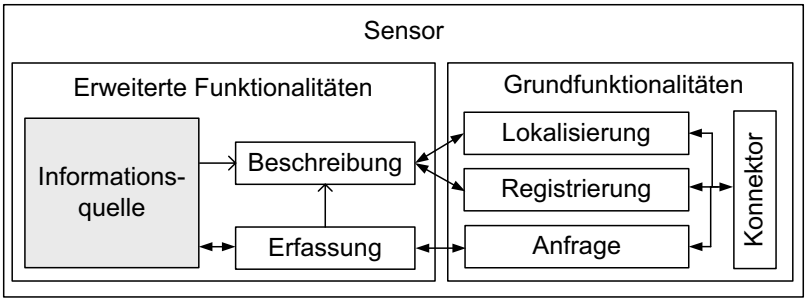


Abbildung 4.3: Schema: Kapselung eines Sensors und der Grundfunktionalitäten

4.3.1 Funktionalitäten der Schnittstelle

Ein Sensor benötigt eine Reihe von grundlegenden Funktionalitäten. Diese Elemente der Schnittstelle sind systemweit bekannt und für Sensoren fest vorgegeben. Sie dienen der Registrierung eines Sensors und der Übermittlung der Beschreibungselemente (siehe Abschnitt 4.4).

Prinzipiell wäre die Abfrage einer Informationsquelle über eine einzige, allumfassende Methode möglich. Dies würde jedoch zu erhöhtem Aufwand führen, da sonst auch nicht benötigte Informationen ermittelt und übertragen werden, welches somit den Rahmenbedingungen (3) Heterogenität, (4) Kosten und (5) Dienstgüte aus Abschnitt 4.1 widerspricht.

Zur Abfrage von Informationsquellen werden daher im Allgemeinen unterschiedliche Arten und Mengen von Methoden benötigt, um den Funktionsumfang der Informationsquelle abzudecken und um eine Anfrage in einzelne Methodenaufrufe aufzuteilen, welche sich entsprechend an den angebotenen Informationen der jeweiligen Methode orientieren. Diese erweiterten Funktionalitäten der Schnittstelle spiegeln die eigentlichen Methoden zur Abfrage der Informationsquelle wieder.

Die Problematik, die sich durch die zweite Rahmenbedingung aus Abschnitt 4.1 (offenes System) ergibt, besteht darin, dass vorab nicht bekannt ist, über welche Funktionen ein Sensor verfügt. Somit ist vorab nicht bekannt, über welche erweiterten Funktionalitäten die Schnittstelle eines Sensors verfügt.

Diese Beschreibungselemente, welche über die Grundfunktionalitäten der Schnittstelle abgerufen werden, dienen auch der Adressierung dieser erweiterten Funktionalitäten (siehe Abschnitt 4.4).

4.4 Beschreibung von Sensoren und Sensordaten

Wie in Abschnitt 4.3 erläutert, ist es notwendig die Schnittstelle eines Sensors geeignet zu beschreiben, um die erweiterten Funktionalitäten zu adressieren. Dort wird auch die Umsetzung eines Sensors als Dienst vorgeschlagen, welcher an ein Netzwerk angebunden und über eine Schnittstelle adressierbar ist. Damit ist ein Sensor vergleichbar mit einem Netzdienst im herkömmlichen Sinne.

Eigenschaften von Beschreibungsmethoden

Eine grundlegende Aufgabe bei der Integration von Beschreibungsmethoden besteht darin, eine gemeinsame Form der Beschreibung zu finden, welche alle Anforderungen gerecht wird.

Eine Beschreibungsmethode hat hierbei folgende Eigenschaften:

- **Datenaustauschformat.**
- **Auszeichnungssprache.**
- **Ausdrucksschema.**
- **Vokabular.**

Es existiert eine Reihe von Technologien als *Datenaustauschformat* und als Auszeichnungssprache, wie beispielsweise *JavaScript Object Notation* (JSON)[6] oder *Abstract Syntax Notation One* (ASN.1)[7].

Eine *Auszeichnungssprache* (engl. Markup Language) dient zur Beschreibung der Daten. Eine Beschreibung von Objekten oder Diensten besitzt in der Regel variable Inhalte, daher muss die Beschreibungsmethode möglichst ohne vordefinierte Felder oder festes Schema auskommen. Als erweiterbare Auszeichnungssprache wird in vielen Ansätzen die *Extensible Markup Language* (XML) genutzt. Im Gegensatz zu anderen Ansätzen ist XML textueller Natur. XML ist für Menschen lesbar, es benötigt in der Regel jedoch mehr Speicherplatz und muss von Maschinen über einen Parser eingelesen und interpretiert werden.

Da XML erweiterbar ist, ist das Schema der Darstellung von Inhalten (Form, Reihenfolge, Hierarchie, etc.) variabel. Daher müssen weitere Vorgaben zur Strukturierung angewandt werden, um die Inhalte einer Beschreibung geeignet zu speichern und später interpretieren zu können. Um Beschreibungselemente anbieten und verarbeiten zu können muss die Struktur der Aussagen innerhalb der Beschreibung allen Systemen bekannt sein. Hierzu ist es notwendig, ein gemeinsames *Ausdrucksschema* anzuwenden.

Als letztes Element gilt es, sich auf ein gemeinsames *Vokabular* für die Elemente innerhalb der Beschreibung zu einigen, um die Aussagen verarbeiten zu können, sie also beispielsweise zueinander vergleichbar oder aufeinander anwendbar zu machen. Hierzu dient in der Regel eine Ontologie.

Inhalt und Ziel der Beschreibung

Die Beschreibung von Diensten kann syntaktisch und/oder semantisch erfolgen:

- **Syntaktische Beschreibung:** Methoden zur syntaktischen Beschreibung beschreiben ein Objekt rein funktional. Im Fokus liegt hier die Beschreibung der Eigenschaften einer Methode, welche für einen Methodenaufruf benötigt werden. Somit können beispielsweise Datentypen für Ein- und Ausgabeparameter festgelegt werden. Welche Bedeutung oder Format ein solcher Ausgabeparameter hat bzw. ein Eingabeparameter haben muss, wird dabei nicht erfasst. Bei einer Abfrage an einem Temperatursensor ist beispielsweise notwendig zu ermitteln, ob es sich bei einem numerischen Ausgabeparameter einer Methode um einen Wert vom Typ Celsius oder Fahrenheit handelt.
- **Semantische Beschreibung:** Wie bereits in Abschnitt 2.4.7 eingeführt, dient eine semantische Beschreibung, der Bestimmung der Bedeutung eines Dienstes. Eine semantische Beschreibung eines Dienstes kann beispielsweise dafür genutzt werden, um zu bestimmen, ob ein Dienst für eine Anfrage überhaupt in Frage kommt. Dies kann beispielsweise erst der Fall sein, wenn der Dienst die erwünschte Funktionalität aufweist und wenn alle benötigten Informationen vorliegen, welche benötigt werden, um den Dienst aufzurufen. Ein weiterer Punkt ist die Bedeutung einer Information klar darzustellen. Wie schon Dey in einer Arbeit zeigte [DMA+02], kann eine rein syntaktische Beschreibung einer Information beispielsweise durch Mehrdeutigkeiten zu Fehlern beim der Nutzung der Information führen.

Fazit: Kombination aus syntaktischer und semantischer Beschreibung

Um die zweite Anforderung aus Abschnitt 4.1 (offenes System) zu adressieren, muss es möglich sein, einen zuvor unbekannten Sensor in das System zu integrieren. Hierzu ist es sowohl notwendig zu bestimmen, ob ein Sensor für eine Suchanfrage relevant und geeignet ist, als auch anschließend die entsprechenden Methoden des Sensors aufzurufen. Eine rein syntaktische oder eine rein semantische Form der Beschreibung reicht hierfür nicht aus. Es müssen also entweder zwei Methoden kombiniert werden, oder es muss eine Methode gewählt werden, die beides ermöglicht.

In dieser Arbeit wird daher von einer Beschreibung ausgegangen, welche syntaktische und semantische Elemente kombiniert anwendet. Von der Informationsgewinnung bis hin zur Informationsauswertung werden die Informationen immer im Bezug auf ihre Beschreibung verarbeitet. Wenn im folgenden Teil dieser Arbeit Begriffe wie Parameter, Informationen oder Daten genutzt werden, so kann im Allgemeinen davon ausgegangen werden, dass auch deren syntaktische und semantische Zuordnung bekannt ist.

[6] http://www.json.org/
[7] http://www.asn1.org/

Vergleichbarer Ansatz - Semantisches Netz

Semantische Beschreibungsmethoden bilden die Grundlage für ein Semantisches Netz[8].

Mit *Semantic Web Services* werden Netzdienste bezeichnet, deren Beschreibung neben der syntaktischen Beschreibung der Schnittstelle auch Elemente zur semantischen Beschreibung des Dienstes beinhalten.

Sensoren können als eine reduzierte Variante eines Netzdienstes betrachtet werden. Im Gegensatz zu Netzdiensten liegt jedoch nicht die Verarbeitung von Daten sondern die Bereitstellung von Informationen im Vordergrund. Bei der semantischen Beschreibung von Netzdiensten wird vor allem auch die Funktion des Dienstes selbst beschrieben. Sensoren führen im Allgemeinen keine Verarbeitung durch oder benötigen die Parameter lediglich zur Adressierung der gewünschten Information (beispielsweise der Name zu Auswahl einer Person im Adressbuch). Betrachtet man Sensoren als Netzdienst, so ist es vor allem wichtig die Informationen zu beschreiben, welche vom Sensor bereitgestellt werden.

Im Zusammenhang mit semantischen Netzen oder Semantic Web Services werden häufig auch Begriffe wie *Web 2.0* oder *Next Generation Internet* genannt. Hierzu zählen meist auch Ansätze zur automatischen Vernetzung von Diensten. Ähnlich wie in dem innerhalb dieser Arbeit angestrebten Szenario besteht die Aufgabe darin, einen zur Anfrage relevanten Dienst zu finden. Semantische Annotationen von Diensten können dabei genutzt werden, um einen geeigneten Dienst automatisch zu ermitteln und zu nutzen.

Bestehende Technologien zur Definition von Ausdrucksschemata

Bei der Suche nach geeigneten Methoden zu Beschreibung von Sensoren liegt es nahe sich an bestehenden Methoden zur Beschreibung von Netzdiensten zu orientieren, um bestehende Technologien und Werkzeuge wiederverwenden zu können.

Ein wesentliches Unterscheidungsmerkmal der betrachteten Sensoren im Vergleich zu Netzdiensten besteht in ihrer Einfachheit. Ein Sensor bietet keine komplexe Funktionalität an, sondern im Allgemeinen lediglich Methoden zur Abfrage von Zuständen, welche teilweise mit Parametern zur Auswahl konkreter Daten versehen sind. Daher geht es weniger darum, komplexe Interaktionen aufeinander abzustimmen, als darum, zu bestimmen, ob ein Sensor für eine bestimmte Suchanfrage relevant ist und ob die gesammelten Daten ausreichen, um eventuell benötigte Parameter geeignet mit Werten zu belegen.

Da Sensoren jedoch eine spezielle Form von Netzdiensten darstellen ist es prinzipiell möglich, die Schnittstellen eines Sensors mit denselben Technologien zu beschreiben, wie sie bei Netzdiensten eingesetzt werden. Dies ermöglicht sowohl die Integration von Sensoren in bestehende dienstorientierte Systeme, als auch die Wiederverwendung der Technologien und Werkzeuge aus dem Bereich der Netzdienste.

Es existieren nur einige wenige Standards zur semantischen Beschreibung von Netzdiensten:

- **Semantic Annotations for WSDL (SAWSDL):** Dieser Ansatz stellt eine Erweiterung des etablierten Standards WSDL zur syntaktischen Beschreibung von Schnittstellen dar. Die Erweiterung sieht die Möglichkeit vor, Referenzen auf Objekte in einer Ontologie in der Schnittstellenbeschreibung zu definieren.
- **Web Service Modeling Ontology (WSMO):** Basiert auf dem *Web Service Modelling Framework* (WSMF) und nutzt *Mediatoren* um zwischen verschiedenen Systemen zu vermitteln.
- **Semantic Markup for Web Services (OWL-S):** Kombiniert die Technologien RDF und OWL zur semantischen Beschreibung von Schnittstellen.

Die Ziele, die durch diese Technologien verfolgt werden, sind eng miteinander verwandt (für eine detaillierte Beschreibung der aufgezeigten Technologien, siehe Abschnitt A.4). Die grundlegenden Funktionalitäten sind oft aufeinander abbildbar. Eine Gemeinsamkeit aller Ansätze besteht unter anderem darin, dass letztlich eine Ontologie benötigt wird, um ein gemeinsames Vokabular zu definieren und Aussagen zueinander vergleichbar beziehungsweise aufeinander anwendbar zu machen. Während WSMO und OWL-S ihre jeweilige Form der Ontologie als integralen Bestandteil mitbringen, wird bei SAWSDL nur die Möglichkeit geboten auf eine Ontologie zu verweisen. Um die semantischen Beschreibungen von SAWSDL zu nutzen und um daraus letztlich einen Mehrwert zu erhalten, müssen daher weitere Technologien zur Auswertung und zur Definition eines gemeinsamen Vokabulars

[8] In dem Artikel *The Semantic Web* [BLHL] wird der Begriff Semantik (engl. semantics) im Zusammenhang mit Computernetzwerken eingeführt. Die Vision bestand darin, vernetzte Geräte und ihre Fähigkeiten mit Beschreibungen zu versehen, welche ihrerseits von anderen Geräten genutzt werden können. Die Voraussetzung für diese Aufgabe besteht darin, dass die Geräte die Beschreibungselemente auslesen, interpretieren und nutzen können. Viele Informationen im Internet sind für Menschen ausgelegt und für Maschinen nicht direkt nutzbar. Die Idee des semantischen Netzes besteht darin, Informationen (parallel zu der von Menschen lesbaren Form) in einer Art und Weise bereit zu stellen, in welcher sie auch von Maschinen verarbeitet werden können. Hierdurch können semantische Suchen ermöglicht werden, welche das Potenzial haben, über die Funktionalitäten einer klassischen Stichwortsuche hinaus, neue Möglichkeiten zu bieten und daher eine potentiell bessere Zielführung einer Anfrage versprechen.

	SAWSDL	WSMO	OWL-S
Matching	- (nur Referenz)	* (Capability)	* (Profile)
Anbindung	- (BPEL4WS, WS-CDL)	* (Orchestration)	* (Model)
Nutzung	* (WSDL, SOAP)	- (Choreography)	- (Grounding)

- Kein Bestandteil der jeweiligen Technologie.
* Wird von der jeweiligen Technologie bereitgestellt.
(...) In den Klammern wird die zum Einsatz kommende Technologie oder dessen Teilbereich genannt.

Tabelle 4.3: Semantische Beschreibungsmethoden

genutzt werden (wie es beispielsweise in den Systemen FUSION und SAWSDL-MX umgesetzt wurde [KK08]). In der Regel wird SAWSDL hierzu mit Web Ontology Language (OWL), WSMO oder mit Ansätzen aus diesen Technologien kombiniert.

WSMO bringt durch Web Service Modeling Language (WSML) eine Möglichkeit der Auswertung mittels logischer Ausdrücke mit sich. In Kombination mit OWL wird die hierzu die Semantic Web Rule Language Semantic Web Rule Language (SWRL)[9] angeboten.

WSMO liefert durch die Mediatoren einen Ansatz, Heterogenität zwischen den Diensten zu behandeln [BFK05]. Dieser Aspekt wird zwar von den anderen Systemen nicht betrachtet, er spielt jedoch auch in dem angestrebten System eine untergeordnete Rolle, da Informationen, wie sie im entwerfenden System zu erwarten sind, in der Regel durch einfache Datentypen dargestellt werden können.

In allen Systemen ist es möglich den Aufruf des Dienstes durch WSDL in Kombination mit SOAP durchzuführen und somit bestehende Systeme zu integrieren. SAWSDL setzt auf WSDL auf und ist damit fest verbunden, wohingegen bei den anderen Technologien an dieser Stelle auch andere Verfahren genutzt werden können.

Grundlegend erfüllen alle Ansätze die Anforderungen, die für die Beschreibung von Sensoren notwendig sind. SAWSDL muss zwar mit anderen Technologien kombiniert angewandt werden, würde jedoch einen eher leichtgewichtigen Ansatz darstellen. Das Ziel mit dem WSMO und OWL-S entworfen wurde ist sehr ähnlich. Daher sind die Unterschiede zwischen WSMO und WSDL zum Teil erst im Detail erkennbar und für Anwendung in dem angestrebten System und die geringen Anforderungen im Bereich Komposition von Diensten eher vernachlässigbar. WSMO bietet ein solides theoretisches Konzept, wohingegen OWL-S an einigen Stellen, wie bei der Beschreibung des Profils eines Dienstes und dem Grounding ausgereifter ist als WSMO [LPL+05].

Alle Technologien werden dazu verwendet drei Aufgaben zu erfüllen:

- **Matching:** Es muss festgestellt werden können, welche Funktionalität ein Dienst erbringt, um bestimmen zu können, ob er für den jeweiligen Anwendungsfall relevant ist.
- **Anbindung:** Wenn ein Dienst relevant ist, muss festgestellt werden, wie er in den Gesamtablauf integriert werden kann (beispielsweise welche Parameter er benötigt oder in welcher Reihenfolge die Methoden des Dienstes aufzurufen sind).
- **Nutzung:** Soll die Methode eines Dienstes letztendlich aufgerufen werden, so muss der Aufruf des Dienstes auf eine syntaktische Ebene heruntergebrochen werden.

In Tabelle 4.3[10] werden die im Anhang im Abschnitt A.4 detailliert beschriebenen Technologien zusammengefasst und verglichen.

4.5 Sensorverzeichnis

Das *Sensorverzeichnis* (oder auch *Service Directory*) ist ein zentrales Element in der Gesamtarchitektur. Es bildet die Schnittstelle zwischen den Informationsquellen auf der einen Seite und den Diensten, welche auf die Informationsquellen zurückgreifen möchten, auf der anderen Seite. Es stellt eine Reihe von Funktionen zur Verfügung, welche für die Informationsgewinnung wichtig sind:

[9] http://www.w3.org/TR/owl-features/
[10] Angepasst von http://kmi.tugraz.at/blogs/wissenstechnologie/files/2007/11/wissenstechnologie-2007-viii.pdf

- Registrierung von verfügbaren Informationsquellen.
- Verarbeitung von Sensorbeschreibungen.
- Suchen über verfügbare Sensoren.
- Bereitstellung der Ontologie.
- Vorverarbeitung und Caching von Suchresultaten.

Diese Funktionalitäten werden in den folgenden Abschnitt weiter erläutert. Durch die Bereitstellung dieser Funktionalitäten als eigenständigen Dienst im Netz wird deren Wiederverwendbarkeit innerhalb anderer Dienste ermöglicht.

4.5.1 Registrierung der Informationsquellen

Zur Abfrage von Sensoren unter Einhaltung von zeitlichen Vorgaben ist es von Vorteil, alle Aufgaben vor der eigentlichen Abfrage durchzuführen, sofern dies möglich ist. Zur späteren Suche nach relevanten Sensoren ist es daher notwendig schon im Vorfeld die Menge der verfügbaren Sensoren zu ermitteln und aktuell zu halten. Das Sensorverzeichnis erfasst und registriert die Informationsquellen aus der Umgebung. Im Gegensatz zu den Namensdiensten anderer Technologien muss das Sensorverzeichnis nicht manuell eingestellt werden oder bekannt sein, da zur Erkennung neuer Sensoren die Mechanismen des Konnektors zur Dienste-Lokalisation genutzt werden. Ebenso können über dieses Konzept mehrere Service Directories parallel betrieben und genutzt werden. Über Änderungen in der Liste verfügbarer Sensoren werden sowohl die Suchverfahren als auch die Caching-Mechanismen informiert. Abhängig vom Suchverfahren kann diese Liste genutzt werden, um beispielsweise einen Suchbaum vorbereitend aufzustellen (siehe hierzu Abschnitt 4.6). Aktive Caching-Mechanismen (siehe Abschnitt 4.7) benötigen diese Liste, um neue Sensoren in die Verarbeitung aufzunehmen oder herauszunehmen.

Auffinden neuer Sensoren

Die Registrierung von Informationsquellen greift unter anderem auf die Funktionalitäten des Konnektors zum Lokalisieren von Diensten zurück (siehe Abschnitt 4.2.2). Mit diesen Funktionen kann ein Sensor aufgefunden und grundlegend von anderen Diensten unterschieden werden.

Wie in dort ebenfalls beschrieben wurde, ermöglichen diese Technologien zwar eine (meist einfache) Beschreibung von Diensten und eine Suche über diese Beschreibung, diese bietet jedoch in der Regel nicht die Möglichkeiten für eine Suche, wie sie in dieser Arbeit angestrebt wird. Daher werden in für die Umsetzung des angestrebten Gesamtkonzeptes weitere Schritte zur umfassenden Beschreibung, sowie zur Auswertung dieser Beschreibung durchgeführt (siehe Abschnitt 4.5.2).

Erkennung nicht mehr verfügbarer Sensoren

Ein Aufruf auf einen nicht mehr verfügbaren Sensor würde bis zu dem Zeitlimit der Suche oder des Aufrufs am Konnektor verzögert werden. Je nach Technologie, welche im Konnektor eingesetzt wird, existieren unterschiedliche Ansätze Abbrüche der Verbindung zu erkennen. Manche Technologien beinhalten eine solche Erkennung, bei anderen kann sie im Konnektor als zusätzliche Funktionalität ergänzt werden und wieder andere erkennen Verbindungsabbrüche erst bei dem Versuch eines Aufrufs einer entfernten Methode (beispielsweise wenn keine Statusnachrichten im Protokoll vorgesehen sind, um Energie zu sparen).

Die Liste der registrierten Sensoren muss daher zeitnah mit den am Konnektor verfügbaren Diensten abgeglichen werden, um nicht mehr verfügbare Sensoren zu erkennen und aus der Liste zu entfernen (die Umsetzung dieser Funktionalität wird in Abschnitt 6.3 beschrieben).

4.5.2 Verarbeitung der Sensorbeschreibungen

Die Beschreibung eines Sensors umfasst folgende Elemente:

- Adresse.
- Relationen des Sensors (*Individual*).
- Beschreibung des Sensor Typs.

Adresse

Die Adresse eines Sensors entspricht dem Ergebnis der Dienst-Lokalisation. Die Adresse ist eindeutig und kann genutzt werden, um über den Konnektor den Sensor zu adressieren. In dem angestrebten System werden hierzu

Adressen im URL-Format genutzt. Die Adresse eines Sensors bezeichnet neben dem Konnektor über den er angesprochen werden kann und der Netzwerk-Adresse des Systems auf welchem er betrieben wird auch den Typ des Sensors und einen zur Laufzeit gültigen Identifier (für Umsetzungsdetails siehe Abschnitt 6.1.1).

Anhand der Adresse vom Sensor kann festgestellt werden, ob bereits dem System andere Sensoren vom gleichen Typ bekannt sind. Bei gleichen Sensor Typen kann von gleichen Schnittstellen ausgegangen werden. Wenn also der Sensor bereits bekannt ist, kann auf eine komplette Verarbeitung der Beschreibung verzichtet werden.

Relationen, Individuals

Dieser Teil der Beschreibung beinhaltet die Elemente, welche den Sensor individuell beschreiben (die *Individuals*). Hierzu gehören Inhalte wie die Zugehörigkeit beziehungsweise Zuordnung des Sensors zu (Kontext-) Objekten. Ein Sensor kann auf diese Weise anderen Objekten zugeordnet werden, beispielsweise:

- Einer Person.
- Einem Ort.
- Einem Gerät.

Diese Zuordnung ist relevant für die spätere Suche. Diese Tags stellen Referenzen auf semantische Elemente dar. Diese semantischen Elemente finden sich in der Beschreibung von Suchergebnissen oder in der Ontologie wieder.

Beschreibung des Sensor Typs

Wie in Abschnitt 4.4 erläutert wurde, gehören zur Beschreibung des Sensor Typs sowohl die syntaktische Beschreibung der Funktionalitäten der Schnittstelle, als auch die semantischen Zusatzinformationen. Sofern der Sensor Typ dem System unbekannt war, wird zudem überprüft, ob eine Anpassung der Ontologie notwendig ist und gegebenenfalls durchgeführt (siehe Abschnitt 4.5.4).

4.5.3 Suchdienst

Der Suchdienst nimmt Suchanfragen entgegen, bestimmt darauf hin relevante Informationsquellen und erfasst schließlich deren Daten. Die Abbildung 4.4 zeigt die Komponenten, welche im Zusammenspiel mit dem Suchdienst zur Informationsgewinnung benötigt werden.

Der Suchdienst bietet eine Schnittstelle für andere Dienste an, über die Suchanfragen gestellt werden können. Die Suche arbeitet direkt auf der Liste der aktuell verfügbaren Sensoren des Sensorverzeichnisses und der aktuell gültigen Ontologie. Die Liste aktuell verfügbarer Sensoren und die Ontologie werden durch das Sensorverzeichnis gepflegt (dargestellt als Pfeile mit gestrichelten Linien). Da die Verzögerung bei der Durchführung der Suche ein wesentlicher Faktor bei der Einhaltung von zeitlichen Rahmenbedingungen ist, ist es sinnvoll die Suche direkt mit dem Sensorverzeichnis zu koppeln.

Die Entwicklung des Verfahrens zur Suche von Informationen, welche mit Hilfe von Sensoren bereitgestellt werden, ist ein zentraler Punkt dieser Arbeit und wird daher ausführlich in Abschnitt 4.6 erläutert.

4.5.4 Ontologie

Semantische Begriffe werden in Sensor-Beschreibungen (Parameter, Individuals), Sensordaten (Ergebnisse) und der Suche (Startwissen, Suchparameter, Suchziele) verwendet. Um zu bestimmen welche Sensoren für eine bestimmte Suchanfrage relevant sind, müssen die Begriffe innerhalb der Beschreibungen der Sensoren und Begriffe innerhalb der Suchanfrage miteinander in Verbindung gebracht werden.

Analog zu XML und der Notwendigkeit von XML-Schemas existiert auch bei der Nutzung von semantischen Begriffen das Problem, dass diese Begriffe (relativ) frei gewählt werden können. Das bedeutet, zur Verarbeitung von semantischen Begriffen muss ein gemeinsames Verständnis für die verwendeten Begriffe definiert werden.

Die Ontologie dient vor allem dazu:

- Einen gemeinsamen Namensraum für Begriffe zu bilden.
- Unterschiedliche Begriffe in Zusammenhang zu bringen.

Eine Ontologie bietet demnach der Suche die Möglichkeit Aussagen über den Zusammenhang zwischen Suchanfrage, Sensorbeschreibung und erhaltenen Informationen zu treffen.

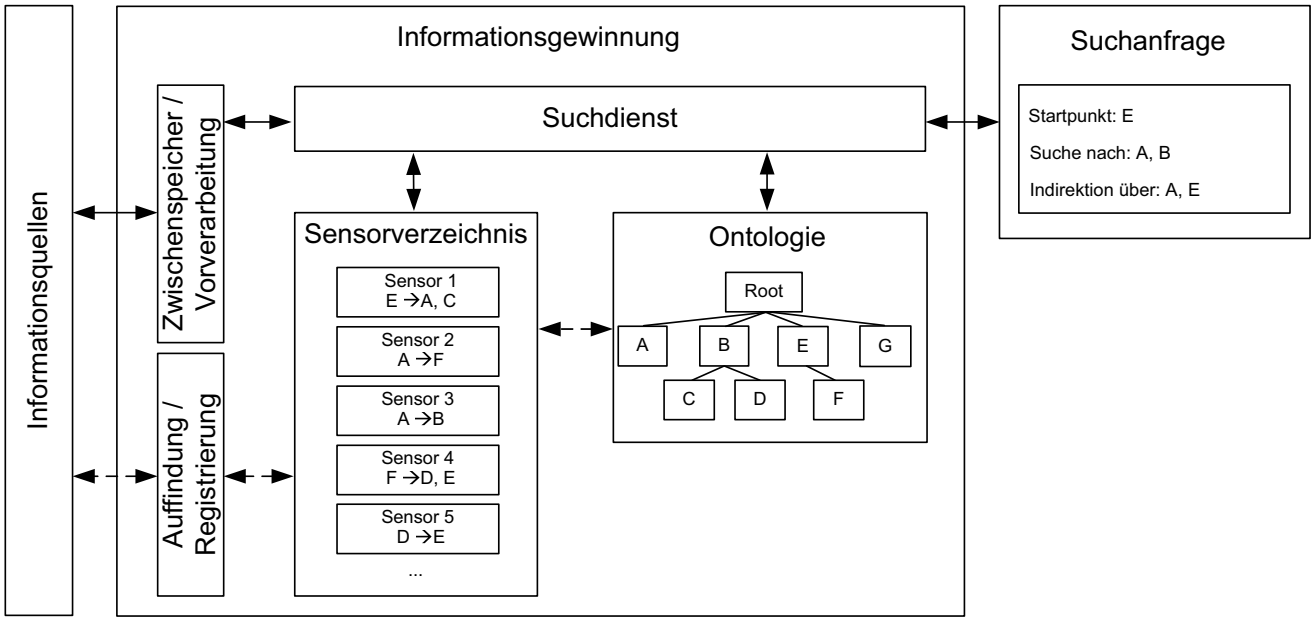


Abbildung 4.4: Schema: Suche unter Einbeziehung des Sensorverzeichnisses und einer Ontologie

In der Regel lässt sich eine Ontologie auf eine Menge von RDF (siehe Abschnitt A.4) Ausdrücken abbilden. Das bedeutet eine Ontologie ist eine Menge von Ausdrücken in der Form:

$$Subjekt \longrightarrow Prädikat \longrightarrow Objekt$$

Subjekte und Objekte nutzen denselben Namensraum, wie er auch von semantischen Begriffen der Beschreibungen genutzt wird. Das Prädikat entspricht dem Zusammenhang zwischen Subjekt und Objekt. Für diese Zusammenhänge ist es ebenfalls wichtig ein gemeinsames Verständnis und einen gemeinsamen Namensraum zu definieren.

Ontologien werden eingesetzt um diese Aufgaben zu erfüllen. Die Umsetzung von Ontologien kann jedoch im Detail unterschiedlich erfolgen.

Die *Web Ontology Language* (OWL) ist ein de-fakto Standard für diesen Einsatzbereich. Sie wird sowohl innerhalb dieser Arbeit eingesetzt, als auch bei fast allen verwandten Projekten.

Im Namensraum der Prädikate in OWL [Die03] (siehe hierzu auch Abschnitt A.4) wird der Basis Typ *is_a* („ist ein") vordefiniert. Mit Hilfe dieses Prädikates lassen sich Elemente hierarchisch in Form einer Baumstruktur zuordnen, bei der das Subjekt die Eigenschaften des Objektes übernimmt und gegebenenfalls erweitert (vergleichbar zur Vererbung).

Anwendungsbeispiel

Ein fest angebrachter Sensor befindet sich in einem Raum mit der Bezeichnung *345a*. Der Sensor enthält in seiner Beschreibung als Individual den Bezeichner des Raums als Teil seiner Beschreibung. Dieser Wert erhält als semantische Bezeichnung *Büro*. Zusammengenommen wäre dies ein Eintrag in der Form *Büro=345a*.

In Abbildung 4.5 ist ein Teil einer Beispiel Ontologie abgebildet. Der Begriff Büro ist in der Ontologie als Untergruppe von Raum eingetragen, welche wiederum eine Untergruppe von Lokation darstellt.

Somit wurde nun erreicht, dass der Begriff *Büro* mit dem Begriff *Lokation* in Verbindung gebracht wurde. Eine Suche nach Sensoren über Lokationen hinweg könnte nun direkt in Verbindung mit dem angeführten Sensor gebracht werden.

Mittels Ontologien lassen sich auch weitere Zusammenhänge darstellen. Ein Beispiel hierfür wäre die Abbildbarkeit von Einträgen aufeinander. So könnten Einträge wie GPS, Bezeichner von Räumen und Adressen (in zum Teil eingeschränkter Genauigkeit) aufeinander abgebildet werden. Über solche Zusammenhänge kann bestimmt werden, welche Ergebnisse einer Sensoranfrage als Eingabeparameter für andere Sensoranfragen in Frage kommen, sofern ein entsprechender Mechanismus zum Transformieren oder Abbilden der Daten zur Verfügung steht.

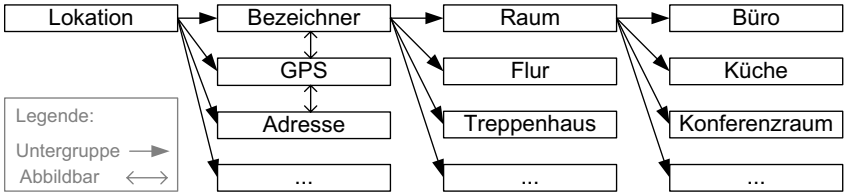


Abbildung 4.5: Ontologie - Beispiel

Ontologie - Anpassung

Es gibt verschiedene Ansätze, Ontologien zu nutzen. Häufig werden sie jedoch als statische Komponente im System genutzt. Dies bedeutet, dass die Ontologie im Voraus den Namensraum definiert und sich alle anderen Systeme beziehungsweise deren Beschreibungen daran orientieren müssen. Dies widerspricht jedoch der zweiten Vorgabe aus Abschnitt 4.1. Nach dieser Vorgabe muss es möglich sein auch im Nachhinein noch unbekannte Sensoren in das System zu integrieren.

Daher wird bei der Verwendung einer Ontologie eine Funktion benötigt, welche es ermöglicht die Ontologie an neue, unbekannte Sensor Typen anzupassen. Dies ist notwendig, um auch eventuell unbekannte Begriffe innerhalb der Beschreibung neuer Sensoren in Zusammenhang mit anderen semantischen Begriffen in der Ontologie zu bringen. Indem neue Begriffe in die Ontologie integriert werden, wird der entsprechende Sensor in Zusammenhang mit bereits bestehenden Sensoren, Parametern und Suchen gebracht.

Herausforderung - Integration von Begriffen

Die Herausforderung liegt in dieser Integration von Begriffen in bestehende Ontologien. Um in diesem Aspekt ebenfalls den Rahmenanforderungen (siehe Abschnitt 4.1 - vierte Anforderung, Kosten der Integration) zu genügen, ist es notwendig diese Aufgabe zu automatisieren. In dem Bereich der automatisierten Anreicherung oder der Verknüpfung von Ontologien gibt es zwar verwandte Arbeiten, welche jedoch meist aus anderen Anwendungsgebieten stammen [Faa04, PNL02]. In diesem Fall besteht die Aufgabe darin, eine Beschreibung eines Sensors so zu gestalten, dass neue Begriffe in Relation zu bestehenden Begriffen innerhalb der Ontologie gebracht werden können.

Sofern die bestehende Ontologie öffentlich zugänglich ist beziehungsweise eingesehen werden kann, kann ein Entwickler oder Administrator auf bestehende Begriffe der Ontologie zugreifen und diese für die Beschreibung des neuen Sensors nutzen.

Ansatz - Anker

Der Ansatz, welcher im Rahmen dieser Arbeit entwickelt und umgesetzt wurde, besteht darin die Elemente innerhalb der Beschreibung eines Sensors mit *Ankern* zu versehen. Diese Anker beschreiben mögliche Zusammenhänge zu anderen Begriffen. Über diese Anker ist es dem System möglich, sinnvolle Anknüpfungspunkte für den neuen Begriff zu bestimmen, um so den neuen Begriff automatisch in Zusammenhang mit bestehenden Begriffen zu bringen (oder gegebenenfalls eine neue Domäne/Kategorie für den Begriff zu etablieren).

Auf diese Weise kann das System zum Startzeitpunkt, mit einer relativ einfachen Basis Ontologie ausgestattet, genutzt werden. Diese Ontologie muss nur die Grundstruktur beinhalten. Die Begriffe der Sensoren füllen schrittweise die Ontologie und bauen Zusammenhänge auf indem sie neue Elemente mit bestehenden vernetzen.

Semi-automatische Begriffsbildung

Das Problem dieses Ansatzes besteht jedoch wieder in der gemeinsamen Begriffsbildung und in dem gemeinsamen Verständnis über die Bedeutung der Begriffe. So ist es nur schwer möglich, eine Suche auf bestimmte Begriffe zu definieren, wenn diese Begriffe (noch) nicht in der Ontologie existieren.

Ein Ansatz besteht in der Vorgabe einer Grundstruktur, welche die Basis für die Anbindung von Sensoren und die Definition von Suchanfragen bildet. Diese Grundstruktur wird zur Laufzeit iterativ durch die Integration unbekannter Sensoren erweitert, was wiederum die Verfeierung von Suchanfragen erlaubt. Dieser semi-automatische Ansatz, bei dem die Designer von Sensoren und Suchanfragen Einsicht in die aktuelle Ontologie haben, bildet die Grundlage für die Nutzung der Ontologie in dieser Arbeit.

Andere Möglichkeiten zur freien Wahl von Begriffen erfordern weitere Maßnahmen, diese Begriffe automatisch in Verbindung zu bringen. Diese Aufgabe ist jedoch sehr komplex und Inhalt vieler anderer aktueller Arbeiten aus dem Bereich Wissensmanagement (Knowledge Management).

Anwendungsbeispiel

Angenommen der Entwickler eines Skype Sensors kennt die aktuell bestehende Ontologie. Die Abbildung 4.6 zeigt den relevanten Teil der Ontologie und die Beschreibung des Sensors. Er beschreibt seinen Sensor und führt den Begriff SkypeID ein, welcher den Ausgabeparameter seines Sensors genauer beschreibt. An die neuen Begriffe setzt er Referenzpunkte (Anker) auf bestehende Elemente der Ontologie, welche den neuen Begriffen am ehesten entsprechen. In diesem Fall nutzt er den Begriff IP-Telefonie als Anker, welcher Adressen im Bereich IP-Telefonie beschreibt. IP-Telefonie bildet somit einen geeigneten Anknüpfungspunkt, da er die Obergruppe von SkypeIDs bildet.

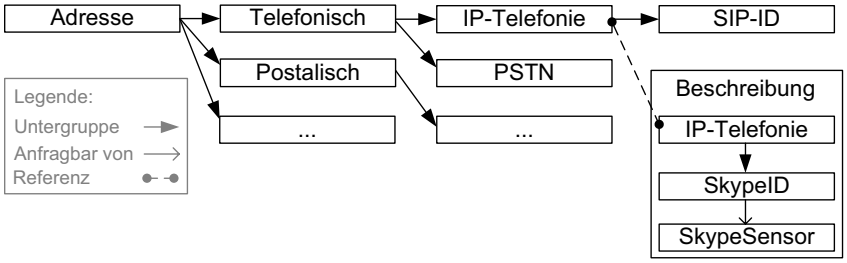


Abbildung 4.6: Anwendungsbeispiel: Anpassung der Ontologie

Der neue Sensor meldet sich an dem System an und überträgt seine Beschreibung. Die Beschreibung wird verarbeitet und der neue Begriff anhand der Referenzen auf existierende Objekte in die Ontologie integriert. Bestehende Suchen, welche beispielsweise nach Telefon-Nummern allgemein oder spezieller nach IP-Telefonie Adressen suchen, nutzen automatisch den neuen Sensor (außer die Suchanfrage schließt die Beziehung *Untergruppe* aus). Neue Suchanfragen können schließlich spezifisch auf den neuen Begriff oder Sensortyp angewendet werden.

Fazit

Alle Sensoren innerhalb einer statischen Ontologie im Vorfeld zu definieren erfordert viel Aufwand und führt zu einem geschlossenen System, welches keine neuen Sensoren aufnehmen kann. Man kann annehmen, dass der Entwickler eines Sensors am besten die Funktionen seines Sensors beschreiben kann. Ontologien können (je nach Umsetzung) innerhalb existierender Editoren oder innerhalb von Browsern betrachtet oder auch editiert werden. Dies ermöglicht dem Entwickler eines Sensors auf bestehende Ontologien zurückzugreifen und sich auf Inhalte darauf zu beziehen.

Das Einfügen von Elementen in die Ontologie kann jedoch durchaus ein komplexeres Verfahren benötigen, als das Anwendungsbeispiel vermuten lässt. Sobald die Ontologie gewachsen und damit weitaus komplexer als in dem Beispiel dargestellt ist, ist das Einfügen neuer Inhalte in komplexe Strukturen nicht mehr trivial. Außerdem werden Fehler innerhalb der Beschreibung eines Sensors möglicherweise auf die Ontologie übertragen. Je nach Anwendungsfall müssten Inhalte umstrukturiert werden, um neue Inhalte aufzunehmen, aussagekräftiger darzustellen oder zu gruppieren. Dies kann einerseits durch aussagekräftigere Beschreibungen der Anker, sowie weiterer Verfahren für deren Verarbeitung geschehen. Andererseits kann hier im Bedarfsfall auch das Eingreifen eines Administrators erfolgen, um die Darstellung der Inhalte und deren Struktur zu optimieren.

4.5.5 Zwischenspeicherung und Vorverarbeitung

Der Suchdienst bestimmt die Sensoren, welche für die jeweilige Suchanfrage relevant sind und angefragt werden sollten. Die Dauer der gesamten Suche hängt maßgeblich von der Anzahl der anzufragenden Sensoren und der Zeitvorgabe zur Durchführung der Suche ab. Je nach Sensor gelten unterschiedliche Zeiträume für die ein Sensorwert gültig ist. Innerhalb dieses Zeitraumes kann ein zuvor angefragter Sensorwert für andere Anfragen wiederverwendet werden. Dies spart Anfragen ins Netz und somit wird die gesamte Dauer der Suche reduziert bzw. es können mehr Sensoren innerhalb eines beschränkten Zeitraumes ermittelt werden.

Die Gültigkeitsdauer kann anhand der Sensorbeschreibung ermittelt werden. Zudem kann anhand von Messung und statistischen Aussagen über das Anfrageverhalten einzelner Sensoren ermittelt werden, welche Sensoren aktiv angefragt und vorgehalten (Prefetching - siehe Abschnitt 4.7) werden sollten. Das Verfahren zum Vorhalten von Sensordaten arbeitet eng mit der eigentlichen Suche und dem Sensorverzeichnis zusammen. Eine Integration in Kombination mit dem Sensorverzeichnis ist daher ebenfalls sinnvoll.

Das Verfahren zum Vorhalten von Daten baut auf den Grundlagen der Suche auf und wird in Abschnitt 4.7 genauer erläutert.

Vorverarbeitung

Es gibt zwei Stellen, an denen eine Vorverarbeitung durchgeführt werden kann. Während der Suche sind Vorverarbeitungsschritte möglich, die auf einzelnen Werten oder nur auf den bisher gesammelten Daten durchgeführt werden können. Beispielsweise können komplexe Daten wie Bilder oder Audiodaten ausgewertet werden, um höherwertige Aussagen, wie die Aktivität innerhalb eines Raumes zu bestimmen. Dies hat den Vorteil, dass Aussagen, welche durch die Vorverarbeitung hinzukommen, wieder in die Suche einfließen können. Diese Form der Vorverarbeitung kann direkt bei der Erfassung bestimmter Datentypen durchgeführt werden, sie kann jedoch auch in Form *virtueller Sensoren* abgebildet werden. Ein solcher virtueller Sensor bietet die Vorverarbeitung in Form von Methoden an, welche die komplexen Datentypen als Eingabeparameter erfordern, und die höherwertigen Aussagen als Resultat anbieten. Dieser Ansatz hat den Vorteil, dass diese Vorverarbeitungsschritte dynamisch in die Suche integriert und bedarfsweise über die Suche adressiert werden können. Zudem können diese möglicherweise aufwendigen Verarbeitungsschritte auf verschiedene Systeme verteilt werden, um so eine Verteilung der Lasten zu erhalten.

Eine weitere Stelle an der eine Vorverarbeitung sinnvoll eingesetzt werden kann, ist nach der Suche. An diesem Punkt liegen alle Sensordaten vor und die Vorverarbeitungsschritte können auch die Aussagen mehrerer Sensordaten und deren Beziehungen untereinander nutzen. Ist zudem die Kontextdimension bekannt, so können die Vorverarbeitungsschritte zusätzlich auch darauf angepasst werden. Daher ist es sinnvoll, diese Form der Vorverarbeitung im Zusammenhang mit der Kontextauswertung durchzuführen. Dieser Aspekt wird in Abschnitt 5.2.5 näher erläutert.

4.6 Suche

Korrekte Entscheidungen können nur dann getroffen werden, wenn (mindestens) eine relevante bzw. ausschlaggebende Informationsquelle verfügbar ist. Dazu ist es hilfreich, auf möglichst viele Informationsquellen Zugriff zu haben. Es ist notwendig, dass die Informationsquellen Meta-Informationen anbieten, um zu bestimmen, welche Informationen relevant sind (d.h. welche in Relation mit dem Nutzer stehen) und wie sie verwendet werden können. Gerade in dynamisch wechselnden Umgebungen ist die Suche nach relevanten Sensoren keine triviale Aufgabe.

Ansätze aus der Literatur

Die Abhängigkeiten zwischen den Sensoren ergeben sich dynamisch. Beispielsweise werden Sensoren in einem Raum erst dann relevant, wenn über einen Lokationssensor zuvor bestimmt wurde, dass sich das Objekt, über welches Informationen gesammelt werden soll, dort befindet. Hierdurch ergeben sich Abfolgen in denen Sensoren angefragt werden müssen. Die Bestimmung der Abfolgen von Aktionen wird in der Literatur als *Automated Planning* bezeichnet. In Ansätzen wie *Partial Order Planning* [Kam95] oder *Hierarchical Task Networks* [EHN94] werden Start- und Zielvorgaben, sowie eine Menge von Regeln zur Definition möglicher Abfolgen von Aktionen definiert. Das Problem dieser Suche kann durch vergleichbare Regeln beschrieben werden, welche die Abfolgen der Abfragen der Sensoren definieren (siehe Abschnitt 4.6.1). Die Startvorgaben beinhalten das Objekt über das gesucht werden soll und die Zielvorgaben entsprechen der Menge von Informationen, welche zur Bestimmung des Kontextes relevant sind. Die Arbeiten in den Bereichen Partial Order Planning oder Hierarchical Task Networks haben häufig zum Ziel, zu bestimmen, ob eine Lösung des Problems, also das Erreichen des Ziels, möglich ist oder sie versuchen den Weg dorthin zu optimieren. Allerdings kann in dieser Arbeit das Ziel, also die Menge relevanter Sensoren im Voraus nicht definiert werden.

Die Eingrenzung relevanter Sensoren kann aufgrund ihrer Semantik erfolgen. Das Vorgehen hierbei ist vergleichbar zur Komposition von Netzdiensten [PF02]. Jeder Sensor kann als einfacher Netzdienst betrachtet werden, mit der Angabe der semantischen Eingabe- und Ausgabeparametern. Wenn der semantische Ausgabetyp eines Netzdienstes als Eingabeparameter auf den nächsten Netzdienst angewandt werden kann und der Ausgabetyp des nächsten Netzdienstes von Interesse ist, wird dieser angefragt. Im Gegensatz zu Netzdiensten haben Sensoren je-

doch Relationen zu Objekten (welche im den Individual-Beschreibung definiert werden). Diese Relation bestimmt maßgeblich die Relevanz eines Sensors und muss vor der Anfrage berücksichtigt werden.

In der Arbeit [WZL06] wird der Datentyp an dem eine Anwendung interessiert ist innerhalb einer Query definiert. Anhand von zuvor definierten Regeln, welche die Abhängigkeiten zwischen den Datentypen beschreiben, kann dann ein Suchbaum aufgebaut werden, welcher zudem nach Latenz oder Kosten optimiert werden kann. Auf diesem Wege werden jedoch nur der Datentyp und die Eigenschaften des Sensors zur Bestimmung der Relevanz berücksichtigt.

Im Gegensatz zu vielen Arbeiten in diesem Bereich, können die Abfolgen von Sensorabfragen jedoch nicht im Voraus bestimmt werden, da zur Bestimmung der nachfolgenden Sensoren, das Ergebnis der vorangehenden Sensorabfrage abgewartet werden muss.

4.6.1 Herausforderungen der Suche

Ziel der Suche ist es

- möglichst alle relevanten Sensoren anzufragen,
- dabei jedoch möglichst wenige Anfragen durchzuführen.

Die Herausforderungen in der Suche bestehen darin, dass die Abfolge der möglichen Anfragen, abhängig von den gesammelten Informationen und den benötigten Eingabeparametern beziehungsweise den Individuals eines Sensors ist.

Das bedeutet ein Sensor kommt für eine Anfrage erst dann in Betracht, wenn folgende Bedingungen erfüllt sind:

1. Ein Sensor ist relevant für die Suchanfrage:
   a) Der Sensor liefert relevante Informationen.
   b) Es besteht eine Beziehung zwischen dem Sensor und dem Objekt, über welches gesucht wird.
2. Alle benötigten Parameter für einen Methodenaufruf wurden gesammelt (jedoch nur wenn der Sensor Parameter benötigt).

Eine Herausforderung in der Durchführung der Suche besteht in der Bestimmung der 1. Aussage. Hierzu müssen Elemente aus der Beschreibung eines Sensors mit der Suchanfrage in Bezug gebracht werden.

Bedingung 1.a - Relevante Informationen

Anhand der Beschreibung des Sensortyps kann ausgewertet werden, ob diese Bedingung erfüllt wird. Nur wenn ein Sensor Informationen anbietet, welche relevant für die Suche sind, kommt er in die engere Auswahl. Diese Bestimmung der Relevanz kann anhand der Ontologie durchgeführt werden. Referenzieren Suche und Sensoren Elemente aus der Ontologie, welche entweder gleich oder direkt verwandt sind oder in einer Beziehung zueinander stehen, welche von der Suche *erlaubt* wird, so sind sie relevant für die Suche.

Bedingung 1.b - Bezug zu Such-Objekten

Die Erfüllung der Bedingung 1.b kann anhand des Individuals (siehe Abschnitt 4.5.2) bestimmt werden. Es gibt Sensoren, welche keinen direkten Bezug zu Objekten haben und daher keine Individuals benötigen. Hierzu gehören Sensoren welche sich nicht an einzelne Objekte binden lassen, wie beispielsweise globale Adressbücher oder Datenbanken.

Sobald ein Sensor jedoch einen direkten Bezug zu einen *Such-Objekt* hat, erlaubt dies eine Bindung des Sensors an dieses Objekt. Da das Individual Teil der Beschreibung ist, liegen diese Informationen über die Bindung eines Sensors zu Such-Objekten direkt bei der Suche vor. Dies wiederum ermöglicht vorab eine Bestimmung der Relevanz des Sensors zu Suchanfragen. Erst wenn die Suche an einen Punkt gekommen ist, an dem eine Beziehung zwischen dem Objekt, auf welches sich der Sensor bezieht, und der Suche hergestellt werden kann, kommt dieser Sensor in Betracht.

Bedingung 2 - Parameter

Ein Sensor, welcher Eingabeparameter benötigt, kann erst dann angefragt werden, wenn diese Parameter vorliegen. Kommt die Suche an einen Punkt, an dem alle Parameter vorliegen, welche benötigt werden, kann dieser Sensor angefragt werden.

Problembeschreibung

Im Bild 4.7 wird das Problem der Suche beschrieben. Diese Darstellung ist bewusst abstrakt gehalten und dient nur dazu exemplarisch die verschiedenen Aufgaben der Suche beschreiben. Es bezieht sich dabei auf die Ontologie und die Suchanfrage aus Abbildung 4.4.

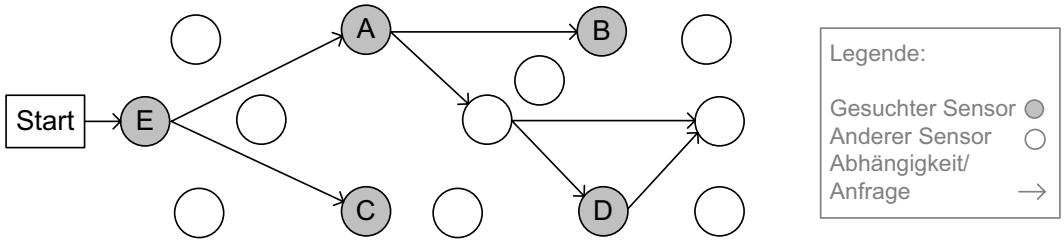


Abbildung 4.7: Suche: Problembeschreibung

Die Suchanfrage aus dem Beispiel startet mit dem Wert E als Startpunkt und sucht nach Werten des Typs A und B. Sofern die Ontologie herangezogen wird, kann festgestellt werden, dass Typ C und D ebenfalls zum Typ B gehören.

Die Suchanfrage ist links als Startpunkt der Suche dargestellt. Eine Suchanfrage liefert die Basis für die Suche. Neben den Informationen, welche genutzt werden, um die Relevanz von Sensoren zu bestimmen, können Informationen bereitgestellt werden, welche als Startpunkt der Suche genutzt werden. Meist läuft eine Suche über ein bestimmtes Such-Objekt. Beispielsweise werden meist Suchen gestartet, welche alle Sensoren in einem Raum suchen oder bestimmte Informationen über eine Person sammeln. Der Bezeichner dieses Raumes oder der Person kann als Startpunkt der Suche genutzt werden.

In der Darstellung ist eine Menge von Sensoren angedeutet, welche als Kreise dargestellt sind. Gegeben sei eine Suche, zu der alle dunkel markierten Sensoren relevant sind (nach Bedingung 1.a). Je nach Suchanfrage werden andere Mengen von Sensoren relevant.

Die Abhängigkeiten zwischen den Sensoren sind durch Pfeile beschrieben. Ein Pfeil bedeutet, dass zuerst der Sensor im Ursprung des Pfeils angefragt werden muss, um eine Beziehung zu dem Sensor aufzubauen, auf den der Pfeil deutet. Das bedeutet, das Ergebnis des Sensors im Ursprung des Pfeils wird benötigt, um die Bedingungen zur Anfrage des Sensors zu erfüllen, auf welchen der Pfeil deutet. Dies kann entweder ein Parameter sein, welcher den Bezug zum Individual des darauf folgenden Sensors herstellt (siehe Bedingung 1.b), oder welcher als Eingabeparameter benötigt wird (siehe Bedingung 2).

Zur besseren Lesbarkeit wurden die Sensoren, nach ihren Abhängigkeiten sortiert, von links nach rechts dargestellt. Durch Abarbeiten der Abhängigkeiten entspricht die Darstellung auch ungefähr der zu erwarteten Reihenfolge der Anfragen. Durch Verzögerungen einzelner Anfragen, kann die Reihenfolge jedoch auch davon abweichen.

Ziel der Suche

Ein naiver Ansatz würde alle Sensoren anfragen. Die Sensoren, welche ohne Bezeichner und ohne Pfeile dargestellt wurden, sind weder relevant für die Suche, noch besitzen sie Abhängigkeiten zu den anderen Sensoren. In reellen Szenarien ist eine Vielzahl solcher Sensoren zu erwarten. Ab einer gewissen Größe des Szenarios wäre jedoch der Aufwand für eine Anfrage aller Sensoren entsprechend groß, was zur Folge hätte, dass QoS-Anforderungen nicht eingehalten werden könnten. Eine Suche muss daher selektiv aus der Vielzahl vorhandener Sensoren diejenigen bestimmen, welche relevant sind.

Zu Beginn des Abschnitts 4.6.1 wurde daher das Ziel der Suche festgelegt, möglichst alle relevanten Sensoren mit möglichst wenig Anfragen zu erfassen. Auf die exemplarische Darstellung der Suche bezogen ergibt dies eine Anfragemuster, wie es in Abbildung 4.8 dargestellt wurde.

In diesem Fall wäre eine *optimale* Suche mit fünf Anfragen möglich. Optimal bedeutet in dem Fall, dass die Ziele der Suche bestmöglich erfüllt werden. Weniger Anfragen sind nicht möglich, da alle Abhängigkeiten aufgelöst werden müssen, um die Bedingung zur Anfrage des nachfolgenden Sensors zu erfüllen oder weil sonst nicht alle relevanten Informationen gefunden werden. Mehr Anfragen sind prinzipiell nicht notwendig, sondern verursachen eine höhere Verzögerung die eventuell QoS Anforderungen verletzt.

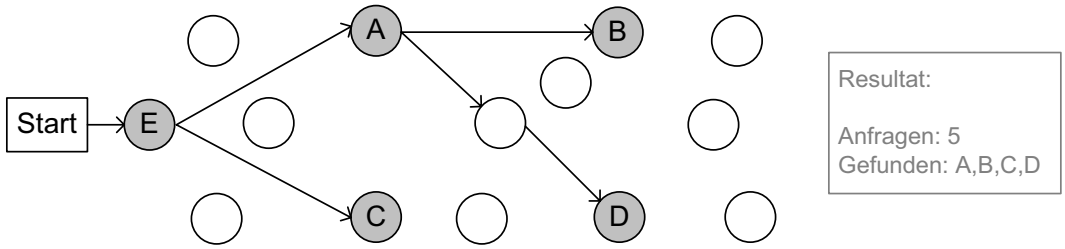


Abbildung 4.8: Ziel der Suche

4.6.2 Ansätze für Suchverfahren

In diesem Abschnitt wird eine geeignete Suche anhand möglicher Ansätze entwickelt. Hierbei werden schrittweise die Aufgaben und Herausforderungen eines jeweiligen Ansatzes beschrieben und im darauf folgenden Ansatz aufgenommen.

Direkte Suche

Eine herkömmliche Suchanfrage beschreibt in der Regel eine *direkte Suche*. Ohne Domänenwissen, kann die Instanz, welche eine Sucheanfrage formuliert nur diejenigen Aussagen nutzen, die ihr zur selbst Verfügung stehen, also zum einen das Objekt über das Gesucht werden soll und zum anderen die Angabe der gesuchten Datentypen.

Bei der *direkten Suche*, werden direkt anhand der Suchanfrage diejenigen Sensoren bestimmt, welche (anhand der Bedingungen 1.b und 2) relevant und abfragbar sind. Dies wären Sensoren welche entweder keine Abhängigkeiten besitzen oder deren Abhängigkeiten durch die Such-Objekte aufgelöst werden können, welche als Startpunkte in der Suchanfrage enthalten sind.

Je nachdem, ob der betrachtete Ansatz der direkten Suche eine Ontologie heranzieht oder nicht, kann weiterhin festgestellt werden, dass der Typ C ebenfalls zum Typ B zählt. Wie in Abbildung 4.9 dargestellt, wären in dem Beispiel die Sensoren A und C in der Ergebnismenge einer direkten Suche unter Ausnutzung einer Ontologie.

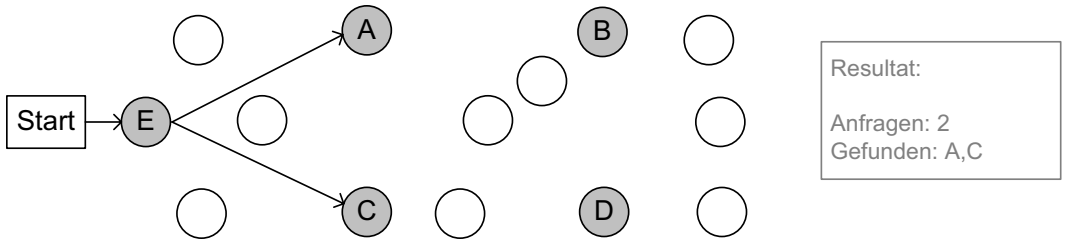


Abbildung 4.9: 1. Ansatz: Direkte Suche

Eine solche Suche würde nur wenige Anfragen erzeugen, dabei jedoch nicht alle relevanten Informationen finden. Je nach Ergebnis einer Anfrage kommen neue Informationen hinzu, welche genutzt werden können, um die Bedingungen 1.b und 2 erfüllen.

Eine Herausforderung liegt darin, dass nicht jede Sensor-Anfrage zu vorhersagbaren Ergebnissen führt. Die Beschreibung eines Sensors liefert lediglich den Rahmen, in dem der Sensor agieren kann. Dies bedeutet die Beschreibung eines Sensors beinhaltet auch Informationen als Ausgabeparameter (im Bezug auf Bedingung 1.a), welche zum Zeitpunkt der Anfrage, nur bedingt ermittelt werden können. Beispielsweise beinhaltet ein Sensor zur Lokalisation von Personen in einem Raum eine Liste von Personen-Bezeichnern als Ausgabeparameter. Befindet sich zum Anfragezeitpunkt keine Person im Raum, so ist folglich das Ergebnis leer. Manche Sensoren beinhalten auf diese Weise eine Reihe von Ausgabeparametern in der Beschreibung, welche sich jedoch nur teilweise in den Ergebnissen wieder finden. Somit lässt sich im Voraus keine feste Reihenfolge der Aufrufe vorgeben.

Iterative Suche

Die *iterative Suche* greift das Problem der direkten Suche auf, indem es die Resultate einer Suchanfrage in einem nächsten Iterationsschritt nutzt, um die Menge der Sensoren zu erweitern, welche die Bedingungen 1.b und 2 erfüllen.

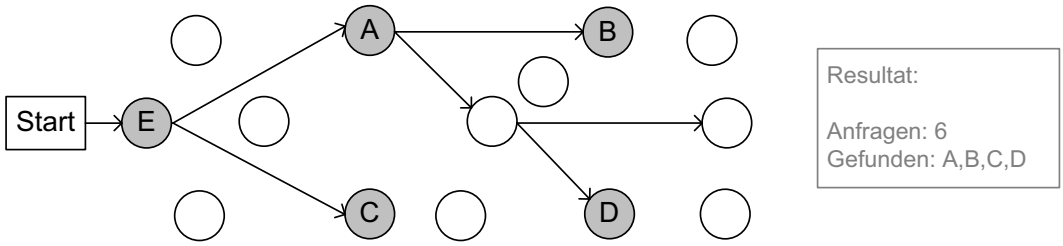


Abbildung 4.10: 2. Ansatz: Iterative Suche

Diese Suche kann wie die direkte Suche parallele Anfragen auf die Sensoren durchführen. Hierbei entsteht nun das Problem der unterschiedlichen Anfragedauer. Die Resultate treffen unabhängig voneinander zu unterschiedlichen Zeitpunkten ein, müssen jedoch einer gemeinsamen *Wissensbasis* zugeführt werden. Diese Wissensbasis bildet die Menge aller gesammelten Informationen und wird genutzt, die Relevanz weiterer Sensoren zu bestimmen.

Anhand der Wissensbasis kann ebenso bestimmt werden, welche Sensoren bereits angefragt wurden und welche Parameter dabei gegebenenfalls verwendet wurden. Ein grundlegender Endpunkt bei einer iterativen Suche ist beispielsweise: Kein Sensor soll mit denselben Parametern mehrmals angefragt werden. Dies bedeutet, dass Schleifen wie zum Beispiel:

$$Raum \overset{lokalisiert}{\rightarrow} Person\ A \overset{kennt}{\rightarrow} Person\ B \overset{kennt}{\rightarrow} Person\ A \overset{kennt}{\rightarrow} \ldots$$

vermieden werden, indem nach der ersten Iteration über den Typ *Person* abgebrochen wird.

Eine iterative Suche würde jedoch vom Startpunkt aus möglicherweise alle Informationen finden, die in irgendeinem Zusammenhang zum Startpunkt stehen würden. In Abbildung 4.10 wäre dies exemplarisch der rechts dargestellte Sensor. In der Realität könnten jedoch eine größere Menge solcher Sensoren existieren. Dies würde zu einer Vielzahl von Anfragen und Informationen führen, die nicht relevant für die Suche sind.

Ontologiebasierte Suche

Ausgehend vom Ansatz der iterativen Suche, geht es um die Einschränkung der Breite der Suche auf relevante Sensoren (nach Bedingung 1.a). Die Ontologie kann genutzt werden, um die Beziehung zwischen einem Sensor und der Suchanfrage zu bestimmen. Ein Sensor kommt nur dann in Betracht, wenn der Sensor Informationen liefert, die in bestimmten Relationen zur Suchanfrage stehen. Wenn ein Sensor relevant ist (Bedingung 1.a und 1.b), jedoch noch nicht alle Parameter zum Aufruf des Sensors vorliegen (Bedingung 2), so werden die benötigten Parameter temporär als relevant markiert (nach Bedingung 1.a). Das bedeutet, sobald ein die Relevanz eines Sensor festgestellt werden kann, versucht die Suche diesen Sensor anzufragen, was wiederum der Zielvorgabe entspricht.

Wie in Abbildung 4.11 dargestellt wird, kann die Suche auf die dunkel eingefärbten Sensoren eingeschränkt werden.

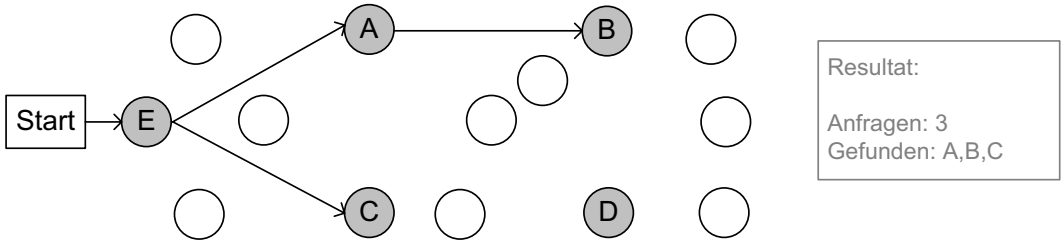


Abbildung 4.11: 3. Ansatz: Ontologiebasierte Suche

Eine Suche, welche auf einer Ontologie basiert, kann zudem erweitert werden, um Sensoren, die nicht in einer direkten Abhängigkeit zu anderen Sensoren stehen, zu adressieren. Hierzu kann die Suchanfrage auch Elemente aus der Ontologie referenzieren, die zwar nicht direkt für das Suchergebnis relevant sind, über die jedoch iteriert werden soll. Hiermit können auch Sensoren wie der in Abbildung 4.11 rechts dargestellte Sensor D gefunden werden.

4.6.3 Verborgene Sensoren in Abhängigkeits-Ketten

Eine weitere Herausforderung sind Sensoren, deren Abhängigkeit über Sensoren führt, die nicht relevant für die Suche sind. Ein solcher Fall ist in der Abbildung 4.11 beim rechten dunkel eingefärbten Sensor D dargestellt. Das Problem solche *verborgenen* Sensoren automatisch zu finden beruht auf mehreren Aspekten:

- Eine solche Kette von Abhängigkeiten kann möglicherweise sehr lang sein.
- Die Kette kann komplexe Abhängigkeiten aufweisen.
- Von den Sensoren in der Kette ist nur der (mögliche) Typ des Ergebnisses bekannt.
- Der Wert des Ergebnisses ist jedoch nicht vorab bekannt, kann also nur bei Anfrage ermittelt werden.
- Je nach Wert des Ergebnisses entstehen andere Zusammenhänge oder Ketten.

Daher können zwar mögliche Abhängigkeits-Ketten anhand der Sensorbeschreibungen ermittelt werden (beispielsweise über graphentheoretische Verfahren), jedoch wäre die Menge der gefunden möglichen Ketten in der Regel ungleich größer, als die Menge, die letztlich zielführend wäre. Letztlich müsst jede der möglichen Abhängigkeits-Ketten auf deren Zielführung hin getestet werden (durch Anfragen der Sensoren in der Kette), um alle verborgenen Sensoren zu bestimmen. Je nach Struktur der Ontologie und der Suchanfrage, könnte sich der Aufwand einer solche Verarbeitung aller möglichen Abhängigkeits-Ketten an den Aufwand der iterativen Suche annähern und somit den Vorteil der ontologiebasierten Suche zunichte machen.

Die Menge der gefundenen, möglichen Ketten muss also auf eine Menge *sinnvoller* Ketten reduziert werden. Dies kann über eine Heuristik durchgeführt werden, die Parameter wie Pfadlänge, Anfragegeschwindigkeit und Ergebniswerte aus der Vergangenheit zur Auswertung heranzieht. Auf diese Weise könnten zumindest Teile der verborgenen Sensoren aufgefunden werden.

4.6.4 Zusammenfassung des Ansatzes

Um QoI-Anforderungen zu erfüllen, muss die Suche aus einer potentiell großen Anzahl von Sensoren diejenigen identifizieren, die für die Suchanfrage relevant sind. Durch QoS-Anforderungen ist in den meisten Fällen jedoch nur eine begrenzte Anzahl von Anfragen möglich. Daher ist eine gezielte Suche nach relevanten Sensoren notwendig. Zur Bestimmung der Relevanz in einem offenen, dynamischen System ist Hintergrundwissen in Form von beispielsweise semantischen Beschreibungselementen und Ontologien notwendig. Die Relevanz von Sensoren lässt sich häufig erst nach der Anfrage anderer Sensoren bestimmen. *Verborgene* Sensoren können nur bedingt ermittelt werden. Unter begrenzter Anzahl von Anfragen alle relevanten Sensoren zu bestimmen ist nicht in jedem Fall möglich.

Bei dem ontologiebasierten Ansatz ist die Suche einerseits darauf ausgelegt, solange in die Tiefe zu gehen, wie weitere relevante Sensoren aufgefunden werden können und andererseits ist die Suche auf die Sensoren beschränkt, welche relevante Informationen liefern. Bei den angeführten Beispielen auf dem vorherigen Abschnitt ist zu erkennen, dass der ontologiebasierte Ansatz mit wenigen Anfragen umfassend die relevanten Sensoren erfassen kann, welche nicht verborgen sind. Der ontologiebasierte iterative Ansatz bietet damit eine Annäherung an die optimale Suche.

4.7 Aktiver Zwischenspeicher zur Einhaltung von Zeitbeschränkungen

Wie in Abbildung 4.4 in Abschnitt 4.5.3 dargestellt, wurde der Zwischenspeicher als Ebene zwischen der Suche und den Sensoren integriert. Alle Anfragen der Suche durchlaufen den Zwischenspeicher. Die grundlegende Funktionalität des Zwischenspeichers ist passiv. Das bedeutet, die Anfrage an den jeweiligen Sensor wird nur durchgeführt, wenn keine Informationen vorliegen oder die Qualität nicht ausreicht. Das Ergebnis der Anfrage wird als Resultat zurückgegeben und für zukünftige Anfragen im Zwischenspeicher hinterlegt.

Eine Herausforderung besteht darin, ebenfalls Sensoren, welche nicht innerhalb der Zeitvorgabe der Suche Angefragt werden können, zu adressieren. Gerade bei mobilen oder ressourcenbeschränkten Endgeräten kann eine Abfrage relativ lange Verzögerung bis zum Eintreffen des Ergebnisses erzeugen. Durch den iterativen Ansatz der Suche addieren sich solche Verzögerungen, so dass potentiell relevante Informationen möglicherweise nicht zur Auswertung zur Verfügung stehen. Um dieser Herausforderung zu begegnen, wurde im Rahmen dieser Arbeit ein Ansatz zum aktiven Datenvorhalten entwickelt und umgesetzt. Diese Erweiterung des passiven Ansatzes ermöglicht die Bestimmung *kritischer* Sensoren und deren aktive Anfrage zur Datenvorhaltung. So ist eine Einhaltung der

Qualitätsanforderung auch bei vielen Iterationen und längeren Verzögerungen (bei der Verarbeitung von Anfragen an die einzelnen Sensoren) möglich.

Ohne das Wissen über die für die Suche verfügbaren Sensoren und deren Verzögerung ist eine Bestimmung *kritischer* Sensoren nicht möglich. Daher wird erst zur Laufzeit ermittelt, welche Sensoren vorgehalten werden sollten, da erst dann Aussagen über die Latenz und die Relevanz eines Sensors gemacht werden können. Ist es nicht möglich den Suchbaum in der geforderten Zeit zu durchlaufen, so entsteht eine Überschreitung der Zeitvorgabe. In einem solchen Fall werden diejenigen Sensoren ausgewählt, welche den größten Einfluss auf die Einhaltung der Anforderungen haben.

Prinzipiell haben diejenigen Sensoren den größten Einfluss auf das Endergebnis, welche:

- als Eingabeparameter für andere Sensoren benötigt werden. Je mehr Sensoren von einem Ausgabeparameter abhängig sind, desto wichtiger ist dessen zeitnahe Bestimmung.
- eine große Verzögerung aufweisen.

In Abbildung 4.11 wurde das Beispiel eines Suchbaumes dargestellt. In der praktischen Anwendung kommen jedoch auch komplexere Zusammenhänge vor. Durch mehrere Eingabeparameter entstehen Knoten mit mehreren Eingangsknoten, so dass sich auch *Wälder* bilden (ein Graph ohne Zyklus). Ein solcher Suchwald ist in Abbildung 4.12 dargestellt.

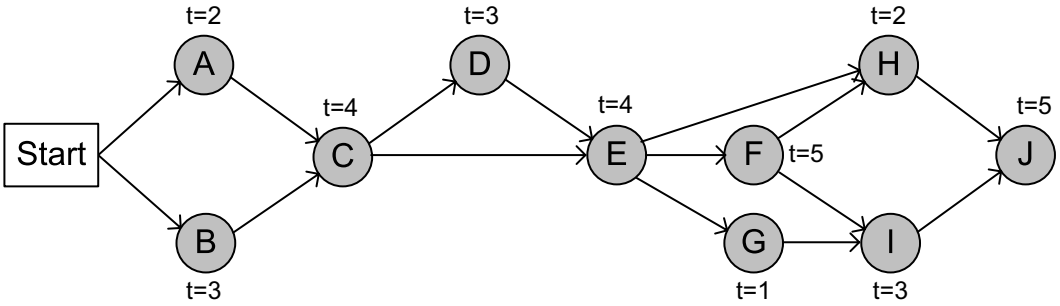


Abbildung 4.12: Die Abhängigkeiten der Suche beschreiben einen *Suchwald*

Er beschreibt die Abhängigkeiten der Sensoren zueinander, wie sie bei der Durchführung der Suche bestanden haben. Ein Sensor, welcher in der Abbildung als Knoten dargestellt wurde, kann erst angefragt werden wenn die Bedingungen aus Abschnitt 4.6.1 erfüllt worden sind. Angenommen dies ist der Fall wenn alle Knoten angefragt wurden, die in der Abbildung als Vaterknoten dargestellt worden sind. Die Darstellung zeigt zudem den dabei gemessenen Zeitaufwand t. Die Bestimmung der kritischen Sensoren erfolgt, wie es in der Tabelle 4.4 dargestellt wurde, über eine Gewichtung des Zeitaufwandes mit der Anzahl der Kindknoten.

Sensor	t	w	$t * w$
A	2	1	2
B	3	1	3
C	4	2	8
D	3	1	3
E	4	3	12
F	5	2	10
G	1	1	1
h	2	1	2
I	3	1	3
J	5	1	5

Tabelle 4.4: Bestimmung kritischer Sensoren

Im Rahmen des aktiven Zwischenspeichers wurde ein Ansatz zu Bestimmung *kritischer* Sensoren entwickelt. Der Ablauf der Auswahl kritischer Sensoren erfolgt in den folgenden Schritten:

1. Bestimme alle Gewichte.
2. Markiere denjenigen Sensor mit dem höchsten Gewicht als *kritisch*.
3. Setze dessen Verzögerung auf 0.
4. Berechne die Gesamtlaufzeit.
5. Ist die Gesamtlaufzeit größer als die Zeitvorgabe, starte bei Schritt 1.

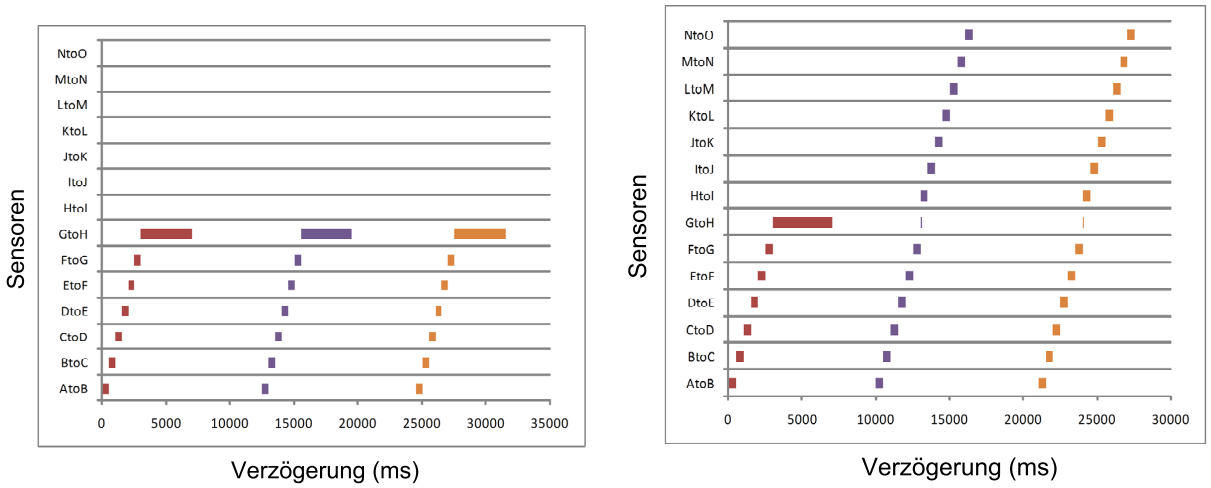
Wird der Wald des Beispiels durchlaufen und summiert man dabei alle Verzögerungen (unter Berücksichtigung möglicher paralleler Anfragen) auf, so erfordert eine Suche $t = 3+4+3+4+5+3+5 = 27$. Ausgehend von einer Zeitvorgabe von $t_{max} = 20$, überschreitet die Suchanfrage diese Zeitvorgabe. In diesem Beispiel wird zu Beginn der Sensor E als kritisch markiert, was zu einer Gesamtdauer von $t = 3+4+3+5+3+5 = 23$ führt. Da dies noch immer oberhalb des Limits liegt, wird als nächstes auch Sensor F als kritisch markiert. Dies führt schließlich zu einer Gesamtdauer von $t = 3+4+3+1+3+5 = 19$, womit die Anforderung für eine gleichartige, zukünftige Anfrage erfüllt wäre.

Kritische Sensoren werden aktiv angefragt, um jederzeit einen Wert mit ausreichender Qualität bereitzustellen. Hierzu wird das Intervall entsprechend an der Qualitätsanforderung des Interesses eingestellt. Wird ein Sensor aufgrund mehrerer Interessen als kritisch markiert, so wird das Intervall entsprechend dem Interesse mit der höchsten Anforderung ausgerichtet.

Es wird regelmäßig überprüft, ob ein kritischer Sensor weiterhin regelmäßig angefragt werden muss. Wird ein kritischer Sensor über einen längeren Zeitraum nicht benötigt oder ist die Gesamtdauer der dazugehörigen Suche im Durchschnitt deutlich unterhalb der Zeitvorgabe, wird ein Sensor wieder als normal eingestuft.

4.7.1 Diskussion des Ansatzes

In den Abbildungen 4.13 wurde ein Testszenario zur Verdeutlichung der Auswirkung aufgebaut. Hierzu wurde eine Reihe von Sensoren mit semantischen Abhängigkeiten zueinander definiert. In diesem Fall bilden die Abhängigkeiten eine einfache Folge, in der jeder Sensor den Datentyp (*A* bis *O*) liefert, welchen der nächste Sensor zur Ausführung benötigt. Zudem wurde bei dem Sensor, welcher den Datentyp *G* auf *H* abbildet, eine erhöhte Verzögerung in die Bearbeitung einer Anfrage künstlich integriert.



(a) Ohne aktive Zwischenspeicherung

(b) Mit aktiver Zwischenspeicherung markierter Sensoren

Abbildung 4.13: Auswirkungen von aktivem Zwischenspeicher

Unter der Annahme, dass die *expirationTime* geringer als die Anfrageperiode ist, kann anhand der Abbildung 4.13 a) abgelesen werden, dass jede der drei periodisch stattfindenden Durchführungen einer Suchanfrage, durch eine Überschreitung der Zeitvorgabe von $t_{max} = 7500ms$, an der Abfrage der verbliebenen Sensoren gehindert wird.

Wird das aktive Zwischenspeichern genutzt, so lässt sich das Verhalten anhand der Darstellung 4.13 b) erkennen. Nach der ersten Überschreitung der Zeitvorgabe wird der Sensor *GtoH* aktiv angefragt, so dass der Wert zum

Zeitpunkt der nächsten Anfrage bereits vorliegt. Dies hat zur Folge, dass nun auch die restlichen Sensoren innerhalb der Zeitvorgabe angefragt werden können.

Sensoren, die kritisch sind, jedoch regelmäßig benötigt werden, können mit Hilfe des aktiven Zwischenspeichers aktiv angefragt und so jederzeit mit ausreichender Aktualität bereitgestellt werden. Das Problem liegt weiterhin in der Dynamik der Abhängigkeiten. Sensoren, die nur kurzzeitig relevant werden, sind zwar für zukünftige Suchen als aktiv markiert, haben jedoch keinen Einfluss auf das Ergebnis der Suche, welche initial zu der Zeitüberschreitung geführt hatte. Zudem werden solche Anfragen nur auf den jeweiligen Eingabeparameter bezogen durchgeführt, was zu keinen Vorteilen bei Anfragen über andere Eingabeparameter führt. Für viele Anwendungsszenarien ist zu erwarten, dass die Anfragen in regelmäßigen Abständen durchgeführt werden und somit häufiger die gleichen Eingabeparameter in den Suchanfragen genutzt werden.

Ein Sensor, welcher statische Informationen, statische Abbildungen oder Informationen mit langer Gültigkeit aufweist, könnte im Voraus alle seine Informationen in den Cache transferieren, wodurch eine Anfrage zur Laufzeit eingespart wird.

4.8 Fazit

In diesem Kapitel wurde auf die Informationsgewinnung eingegangen. Die Informationsgewinnung hat zum Ziel, eine dynamische Menge heterogener Sensoren anzubinden und anderen Komponenten den Zugriff, auf die Informationen der Sensoren zur weiteren Auswertung und Nutzung, zu ermöglichen. Zur plattformübergreifenden Kommunikation können Konnektoren an das System angebunden werden, welche geeignete Mechanismen zur Datenrepräsentation, zum Datentransport, zur Anfragespezifikation und zur Dienstelokalisation bereitstellen. Es wurden hierzu jeweils eine Reihe von Technologien vorgestellt und verglichen. Zur Integration unbekannter Sensoren müssen diese lediglich über eine Schnittstelle zur Abfrage weiterer Beschreibungen verfügen. Anhand dieser Beschreibung können spezifische Funktionen der Sensoren adressiert und genutzt werden. Eine weitere Aufgabe der Beschreibung eines Sensors liegt in der Bestimmung seiner Relevanz im Bezug zu einer Suchanfrage. Anhand der Beschreibung können Aussagen über die Relationen zwischen der angebotenen Information eines Sensors und der Suchanfrage, sowie zwischen dem Sensor und anderen Objekten hergestellt werden. Die entwickelte Suche ist in der Lage, relevante Sensoren anhand Ihrer Beschreibung und einer erweiterbaren Ontologie zu identifizieren. Des Weiteren kann diese Suche über die erhaltenen Suchergebnisse iterieren, um auch Sensoren zu adressieren, welche nur indirekt mit dem Suchobjekt in Verbindung gebracht werden können. Diese Vorgehensweise ermöglicht die Integration neuer Sensoren zur Laufzeit und reduziert das notwendige Domänenwissen zur Formulierung von Suchanfragen. Zur Einhaltung von Vorgaben in Form von Zeit- und Qualitätsanforderungen wurde ein aktiver Ansatz zur Datenvorhaltung entwickelt, welcher es ermöglicht Sensoren zu identifizieren, deren proaktive Anfrage maßgeblich zur Einhaltung dieser Vorgaben beiträgt.

5 Informationsauswertung

Dieses Kapitel behandelt die Auswertung der gesammelten Informationen zur Bestimmung des Kontextes. Die Nutzung von Kontext im Bereich der Kommunikation ermöglicht die Entwicklung neuartiger Dienste und Funktionen. Diese kontextberücksichtigenden Kommunikationsdienste können Aufgaben im Bereich der Steuerung und Verarbeitung von Kommunikationsanfragen an die Wünsche und Situation der Nutzer anpassen [SHS07]. Durch Einbeziehung der Zustände beider Kommunikationsteilnehmer kann die der Situation angemessenste Form der Kommunikation gewählt werden. Durch diese automatische Verarbeitung von Anfragen kann der Nutzer entlastet und Anfragen mit erhöhter Zielführung verarbeitet werden.

Die Bestimmung des Kontextes von Nutzern und Geräten ist die Basis für kontextberücksichtigende Anwendungen. Ein kontextberücksichtigendes System beinhaltet also eine Komponente zur Informationsauswertung, welche die Funktion hat, den aktuellen Zustand eines Nutzers zu bestimmen (im Folgenden als *Kontextdienst* bezeichnet), sowie darüber liegende Komponenten, welche den Kontext nutzen. In diesem Kapitel geht es darum, einen möglichst generischen Ansatz für einen Kontextdienst zu entwickeln.

Vorgehen

Im ersten Abschnitt 5.1 werden die Ziele und Rahmenbedingungen für die Informationsauswertung bestimmt. In diesem Zusammenhang werden bestehende Ansätze zur Auswertung von Informationen aufgezeigt und verglichen. Aus diesem Vergleich heraus begründet sich wird die Wahl des adaptive Ansatzes. Anschließend werden die Anforderungen aufgezeigt, welche im Rahmen eines adaptiven Ansatzes zusätzlich entstehen.

Im nächsten Abschnitt 5.2 wird die Architektur des Dienstes zur Auswertung der Informationen im Rahmen der Gesamtarchitektur dargestellt, um die Abhängigkeiten zu anderen Komponenten des Systems aufzuzeigen. Zudem werden die Aufgaben und notwendigen Funktionalitäten zur Integration adaptiver Verfahren erläutert.

Im Abschnitt 5.3 wird genauer auf die Lernverfahren eingegangen. Aus dem gegebenen Anwendungsszenario heraus, ergibt sich eine Reihe von Effekten, welche die Lernverfahren vor verschiedene Herausforderungen stellt. Diese Effekte werden erläutert und bestehende Ansätze zur Behandlung aufgezeigt. Die existierenden Lernverfahren wurden für verschiedene Problemstellungen konzipiert und gehen die Aufgaben unterschiedlich an. Anhand dieser Vorgehensweisen ergeben sich verschiedene Eigenschaften für die Verfahren, welche deren Eignung im Bezug auf die verschiedenen Anwendungsszenarien maßgeblich beeinflussen. In diesem Zusammenhang werden daher die existierenden Lernverfahren kategorisiert, auf ihre Anwendbarkeit hin untersucht und ausgewählt.

Der Abschnitt 5.4 beinhaltet die Entwicklung eines Testaufbaus, welcher sich zur Analyse der Verfahren eignet. Die Ergebnisse einer Vielzahl von Tests werden dann zur Bewertung der Verfahren im Bezug auf die verschiedenen Rahmenbedingungen genutzt.

Anhand der Ergebnisse der Analyse können kritische Eigenschaften der existierenden Lernverfahren deutlich gemacht werden. In den Abschnitten 5.5 und 5.6 werden Ansätze bestimmt und verglichen, welche als Grundlage zur Behandlung dieser Eigenschaften geeignet erscheinen. Als nächsten Schritt werden ausgewählte Ansätze erweitert und auf die Rahmenbedingungen angepasst. Diese erweiterten Ansätze wurden anschließend umgesetzt und ebenfalls im Rahmen des vorgestellten Testaufbaus analysiert.

5.1 Ziele und Rahmenbedingungen

Wie in den Anforderungen zu Beginn festgelegt (siehe Abschnitt 5.1.2), spielt der Mehrwert seitens des Anwenders eine Schlüsselrolle. Um einen Mehrwert zu erhalten muss der Aufwand für den Nutzer geringer sein als der Nutzen, den er daraus zieht.

Wie in Abbildung 5.1 dargestellt, besteht ein wesentliches Ziel darin eine lose Kopplung zwischen der Einspeisung von Informationen und der Nutzung des Kontextes zu ermöglichen. Aus Sicht des Nutzers soll das System dabei ähnlich einer *Black Box* funktionieren. Auf der einen Seite sollen Informationen an das System angebunden werden können. Auf der anderen Seite soll der Nutzer einen Mehrwert aus dem System ziehen und beispielsweise durch selbsttätig getroffene Entscheidungen des Systems unterstützt werden. Die grundlegende Kontrolle behält der Nutzer dabei durch die Möglichkeit, dem System mittels Feedback die getroffene Entscheidung zu bewerten oder die gewünschte Antwort vorzugeben.

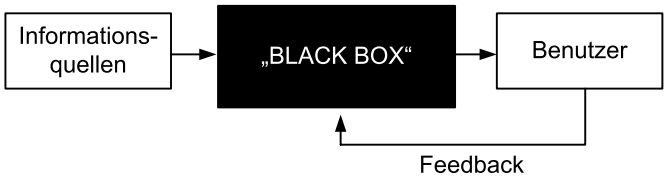


Abbildung 5.1: Ziel: Lose Kopplung zwischen Informationsquellen und Informationsnutzung

Durch die zuvor in Kapitel 4 beschriebenen Mechanismen ist es möglich Informationsquellen lose gekoppelt anzubinden. Durch die semantische Suche werden jeweils die zu einer bestimmten Situation relevanten Informationsquellen erfasst und aggregiert.

Die Herausforderung, die in diesem Kapitel adressiert wird, ist nun, diese Menge an Informationen geeignet auszuwerten. Hierzu wurde in Abschnitt 3.2, sowie in Abbildung 3.3 der *Kontextdienst* vorgesehen. Der Kontextdienst wertet die beobachtete Situation beziehungsweise die dabei zugrunde liegenden Informationen mit Hilfe eines *Entscheidungsmodells* aus. Im Rahmen des angestrebten Systems steht der Kontextdienst folgenden Herausforderungen gegenüber:

- Hohe Anzahl von Informationsquellen.
- Dynamische Menge verfügbarer und relevanter Informationen.
- Reduzierte Visibilität verfügbarer Informationsquellen.
- Komplexe Nutzerkonzepte.
- Änderungen der Nutzerkonzepte.

Daher muss der Kontextdienst die Fähigkeit erhalten, das Modell auf Basis von Beobachtungen der Situation, beziehungsweise über die dabei zugänglichen Informationen, automatisch anzupassen [WF01].

5.1.1 Bestehende Ansätze

Es existieren verschiedene Ansätze zur Auswertung von Informationen. Diese werden im folgenden Teil dieses Abschnittes kategorisiert, erläutert und verglichen.

Statisch - Regelbasierte Ansätze

Existierende Verfahren zur Kontextauswertung nutzen häufig statische Modelle wie beispielsweise Regelwerke oder Skripte zur Auswertung von Kontextinformationen.

Nutzergenerierte Kommunikationsdienste

Existierende Telefoniesysteme bieten dem Teilnehmer häufig die Möglichkeit, Funktionen wie beispielsweise Anrufweiterleitung, Anrufunterdrückung oder E-Mail-Notifikation zu nutzen. Fortgeschrittenere Systeme ermöglichen zudem die Erstellung einfacher Regeln. Diese Ansätze reichen hin zu offenen APIs und meist XML-basierten Skriptsprachen, welche es einer Entwicklergemeinde ermöglichen soll, neue verbesserte Telekommunikations-Dienste zu integrieren. In Abschnitt 2.4.5 und 2.4.5 wurden diese *Nutzergenerierten Kommunikationsdienste* eingeführt.

Existierende Skriptsprachen

Mit der Diensterstellung in Next Generation Networks (NGN) speziell mit in XML notierten Skriptsprachen beschäftigen sich [BJ02] und [LF03]. Es werden einige XML Skriptsprachen zur Anrufsteuerung (Call Control) genannt, darunter CPML, TML, XTML und CallML. Der Fokus in [BJ02] liegt aber auf der Service Control Markup Language (SCML), die vom JAIN Forum entwickelt wird, und der Call Processing Language (CPL). Diese beiden Ansätze werden miteinander verglichen. In dem Bereich der APIs und Skriptsprachen zur Anrufverarbeitung existieren weitere Ansätze wie beispielsweise SIP-CGI, OSA/Parlay, SIP Servlets, JAIN-SIP Lite, XTML, VoiceXML und CCXML, welche in Abschnitt 6.5 vorgestellt werden.

CPL, Vorarbeiten am Fachgebiet KOM

Die zuvor genannten Ansätze für skript- oder regelbasierte Erstellung von Kommunikationsdiensten ermöglichen es, den Anruf auf der Basis einzelner, vorhandener Informationen wie beispielsweise der Absenderkennung, der Zeit oder anderer anrufbezogener Daten zu verarbeiten. Diese Ansätze ermöglichen zum Teil die Integration externer Informationsquellen in den Entscheidungsprozess oder können dahingehend erweitert werden. In einer vorangegangenen Arbeit am Fachgebiet KOM wurde CPL erweitert, um externe Informationsquellen zu integrieren [Gör05], [Sch04]. Durch die beschriebene Erweiterung kann beispielsweise der Kontext von Personen in die Anrufverarbeitung mit einfließen. Der Nutzer hat jedoch selbst die Aufgabe die Art und Weise der Auswertung der Informationen in Form von kaskadierbaren Regeln zu bestimmen.

Deduktive Ansätze

Deduktive Ansätze nutzen Aussagen über Zusammenhänge (Inferenzregeln), um eine Menge von Aussagen über logische Schlussfolgerungen zu erweitern. Beispielsweise kann aus der Aussage „Der Benutzer X befindet sich in Raum R" und dem Zusammenhang „Raum R ist der Arbeitsplatz von Benutzer X, Y und Z" die Menge der Aussage erweitert werden um „Der Benutzer X sitzt an seinem Arbeitsplatz".

Experten Systeme

Sogenannte regelbasierte *Experten Systeme* nutzen oft eine Menge von logischen Regeln, um Zusammenhänge für ein bestimmtes Anwendungsgebiet zu beschreiben. Diese Regeln müssen jedoch von *Experten* definiert werden und mit einer komplexen Menge anderer Regeln harmonieren.

Reasoning

Zusammenhänge können jedoch komplexe Formen annehmen. So können Aussagen über die Beziehung von Kontexten zu Kontextinformationen dargestellt werden. Eine Auswertung von gesammelten Kontextinformationen erfolgt dann über einen sogenannten *Reasoner*. Dieser *Reasoning-Prozess* wendet die gesammelten Kontextinformationen beispielsweise auf die in einer Ontologie dargestellten Aussagen an, um die Menge der gültigen Kontexte zu bestimmen.

SWRL, RuleML

An dieser Stelle kann die *Semantic Web Rule Language* (SWRL)[1]) beispielhaft als relativ weitverbreiteter Ansatz genannt werden. Mit SWRL kann Reasoning unter Ausnutzung von Wissen, welches in Form einer Ontologie vorgehalten wird, durchgeführt werden kann. SWRL nutzt die *Rule Markup Language* in Kombination mit OWL, um aus einer Menge von semantisch beschriebenen Informationen weitgreifende logische Schlussfolgerungen zu treffen.

Adaptive Ansätze

Prinzipiell geht es bei dynamischen Ansätzen darum, automatisch die Methoden zur Auswertung an neue Informationsquellen und Nutzerkonzepte anzupassen. Bei der Anpassung des Modells geht es wiederum darum ein Muster zwischen den beobachteten Informationen und dem Verhalten des Nutzers zu bestimmen und dies in Form eines Modells zu repräsentieren.

Induktive Modellgenerierung

Im Gegensatz zu den deduktiven Ansätzen, welche durch die Anwendung von vorhandenen Regeln Aussagen treffen, bauen induktive Verfahren ihre Regeln anhand von Beobachtungen und dem Erkennen von Zusammenhängen auf. Dieser induktive Ansatz ist die Basis der Verfahren die im folgenden Teil der Arbeit Verwendung finden.

Herkömmliche Anwendungsbereiche

Induktiven Verfahren zur Modellgenerierung werden im Bereich des sogenannten *Data-Minings* zur Auswertung großer Datensätzen auf bestimmte Merkmale hin genutzt. Dies erfordert meist das Vorgehen einer Form, wie sie in Abbildung 5.2 dargestellt ist.

In der Regel wird mit einer *Lernphase* begonnen, welche auf der Basis von Trainingsdaten ein Modell erstellt. Im Falle des *supervised-learnings* (siehe Abschnitt 5.3.2), enthalten diese Trainingsdaten, neben den auszuwertenden

[1] `http://www.w3.org/Submission/SWRL/`

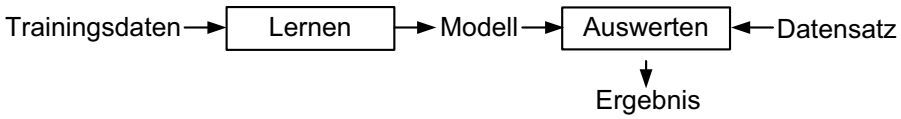


Abbildung 5.2: Herkömmliche Nutzung von Entscheidungsmodellen

Informationen den jeweils erwarteten Ergebniswert. Im folgenden Schritt wird dieses Modell in der *Auswertungsphase* zum Auswerten der Datensätze genutzt, um die (wahrscheinlichsten) Ergebniswerte zu bestimmen. In diesen Anwendungsszenarien sind die auszuwertenden Datensätze meist sehr umfangreich. Daher wird ein Modell benötigt, welches vor allem eine effiziente Auswertung erlaubt. Für die Erstellung des Modells kann jedoch erhöhter Aufwand in Kauf genommen werden.

Iterative Anpassung des Modells

In den angestrebten Anwendungsszenarien soll sich das Modell jedoch an das Konzept des Nutzers anpassen lassen und auch zur Laufzeit neue Sensoren in die Entscheidung mit einbeziehen. Dies macht eine iterative Anpassung des Modells notwendig, wie sie in Abbildung 5.3 schematisch dargestellt ist.

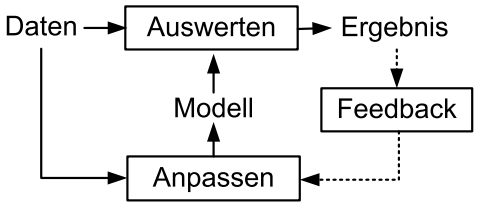


Abbildung 5.3: Iterative Nutzung von Entscheidungsmodellen

Im Gegensatz zur zuvor beschriebenen, herkömmlichen Nutzung von Entscheidungsmodellen, finden Lernphase und Auswertungsphase nicht jeweils nur einmal und getrennt voneinander statt, sondern wechseln sich gewissermaßen ab. Dabei wird das Modell schrittweise bzw. iterativ erweitert. Im Falle des *supervised-learnings* (siehe Abschnitt 5.3.2) kann eine Anpassung des Modells durchgeführt werden, wenn der Benutzer Feedback zu dem Ergebnis einer Auswertung gibt. Dieses Feedback in Zusammenhang mit den Daten, welche zur Auswertung herangezogen wurden, entspricht dem Wunsch des Nutzers in der jeweiligen Situation. Diese Kombination von Daten mit dem darauf gegebenen Feedback wird im folgenden Teil der Arbeit als eine *Trainingsinstanz* bezeichnet. Die Daten, welche ausgewertet werden sollen, also bei denen, ohne eine Vorgabe von Feedback oder ähnlichem, das Resultat bestimmt werden soll, werden als *Klassifikationsinstanzen* bezeichnet.

In diesem Anwendungsszenario spielen die Trainingsinstanzen die Rolle der Trainingsdaten. Die Menge der verfügbaren Trainingsinstanzen ist zu Beginn leer und wird zur Laufzeit mit Einträgen gefüllt. Ein iterativer Ansatz macht eine Anpassung des Modells zur Laufzeit notwendig. Neben einer effizienten Auswertung ist somit auch eine effiziente Methode zur Anpassung des Modells notwendig. Eine detaillierte Klassifizierung von Algorithmen findet sich in Abschnitt 5.3.2.

Vergleich möglicher Ansätze

Eine hohe Benutzerfreundlichkeit bei der Nutzung des Gesamtsystems spielt eine wichtige Rolle in dieser Arbeit. Einfach zu bedienende Schnittstellen zur Bedienung sind eine Voraussetzung für eine hohe Akzeptanz der Anwendung beim Benutzer.

Um die Auswertung an die sich ändernden Anforderungen von Nutzern anzupassen oder neu hinzugekommene Informationsquellen verarbeiten zu können, ist es notwendig die Art und Weise der Auswertung anzupassen.

Probleme von Ansätzen mit statischen Modellen oder Regeln

Das Problem statischer Ansätze ist der Aufwand und die Komplexität, welche seitens des Nutzers durch die Erstellung und Anpassung des Entscheidungsmodells entsteht.

Bis zu einem gewissen Grad können Werkzeuge helfen die Komplexität zu reduzieren oder handhabbar zu machen. Beispielsweise gibt es für die zuvor genannte Erstellung von Skripten der CPL-Sprache Werkzeuge mit einer grafischen Oberfläche um die Benutzerfreundlichkeit erhöhen, wie den CPL-Editor, der an der Humboldt-Universität zu Berlin in Kooperation mit Siemens im Rahmen des Projekts *X-ING* [BBP+01] erstellt wurde und ein weiterer Editor für CPL mit dem Namen *CPLEd* [Ian07].

Im Falle der Auswertung über Regeln wäre es notwendig diese Regeln anzupassen oder im Falle deduktiver Auswertungsmethoden wäre es notwendig die Menge der Zusammenhänge, welche zur Auswertung genutzt werden zu verändern.

Vorteile adaptiver Ansätze

Ein Ansatz, welcher das Modell automatisch anpassen kann, hat gegenüber einem statischen Ansatz folgende Vorteile:

- **Komplexität des Modells zur Beschreibung des Nutzerkonzeptes:** Um das Konzept eines Nutzers umfassend zu beschreiben, müssen meist eine Vielzahl von Merkmalen genutzt werden. Dieses Nutzerkonzept soll nun durch ein Modell beschrieben werden. Die Komplexität des benötigten Modells steigt mit der Anzahl relevanter Merkmale. Eine automatische Generierung des Modells kann (abhängig von der zugrundeliegenden Qualität der Kontextinformationen und des Feedbacks – QoC und QoF) auch komplexe Muster identifizieren.
- **Erstellung und Anpassung des Modells:** Nutzerspezifische Anforderungen müssten vom Nutzer selbst formulierbar sein. Ein solch komplexes Modell manuell durch Skripte oder Regeln zu beschreiben, ist eine aufwendige und schwierige Aufgabe, so dass sie von vielen Nutzern nur begrenzt durchführbar ist. Die Definition von Regeln oder Zusammenhängen ist, spätestens bei einer großen Menge von Parametern und komplexeren Nutzerkonzepten, eine hoch komplexe Aufgabe und daher für viele Anwender oft nicht geeignet. Insbesondere wenn die Menge der zu verarbeitenden Informationen dynamisch variiert, ist die Verarbeitung über statische Ansätze nicht sinnvoll. Hinzu kommt, dass ein manueller Ansatz jeweils aufwendig an neue Verhaltensmuster des Nutzers angepasst werden muss. Eine automatische Generierung und Anpassung des Modells würde den Aufwand seitens des Nutzers deutlich reduzieren.
- **Unbekannte Informationsquellen:** Eine Rahmenbedingung, welche vorgibt, dass alle Informationsquellen im Voraus bekannt sein müssen, würde die Einsatzfähigkeit eines Systems stark begrenzen. Erst die Anforderung auch unbekannte Informationsquellen verarbeiten zu können, macht das Konzept generisch einsetzbar, dynamisch erweiterbar und verringert den initialen Konfigurationsaufwand. Bei hoher Anzahl relevanter Informationsquellen wäre eine manuelle Auswahl und Integration von Informationsquellen nur mit hohem Aufwand möglich. Daher ist es für ein anwenderfreundliches System notwendig, automatisch relevante Informationsquellen zu bestimmen und dynamisch in den Ablauf zu integrieren. Diese Eigenschaft führt auch weg von einem statischen (bei dem der Nutzer selbst alle relevanten Informationsquellen kennen muss), hin zu einem automatischen Ansatz zur Generierung des Modells.

Fazit des Vergleichs

Gerade in Anwendungsszenarien mit großen, dynamischen Mengen von Informationsquellen, in denen die Nutzung durch den Anwender gesteuert erfolgen soll, werden adaptive Ansätze benötigt. Auf der anderen Seite sind adaptive Ansätze stark abhängig von der Informationsgrundlage, der Aufbereitung und der Darstellung der gesammelten Informationen und den Fähigkeiten des Lernsystems. Je nach Anwendungsszenario können einmal generierte Modelle, auf andere Nutzer übertragen und als initiales Modell wiederverwendet werden. Trotzdem hängt ein adaptiver Ansatz auch vom Mitwirken der Anwender selbst ab und erfordert in manchen Anwendungsszenarien die Bereitschaft zu Beginn der Nutzung eine gewisse Phase des Trainings des Modells zu akzeptieren. Diese Arbeit behandelt daher die Machbarkeit und die Betrachtung der dabei zu erwartenden Eigenschaften bei der Nutzung adaptiver Auswertungsverfahren im Rahmen von Kommunikationssystemen.

Statische Ansätze haben jedoch auch ihre Daseinsberechtigung und können genutzt werden, um eine gewisse Grundfunktionalität zu gewährleisten. Beispielsweise können statische Ansätze dazu beitragen, ein initiales Modell aufzubauen oder in Fällen herangezogen werden, in denen mit Hilfe des adaptiven Modells (noch) keine sichere Entscheidung getroffen werden kann. Ebenso können mit Hilfe statischer Ansätze auch Ausnahmesituationen abgedeckt werden, für die im Normalfall kein Training möglich ist, wie beispielsweise Notrufe.

5.1.2 Anforderungen an adaptive Kontextbestimmung

Das angestrebte System für kontextberücksichtigende Kommunikationsdienste definiert eine Reihe anspruchsvoller Anforderungen, welche aus unterschiedlichen Bereichen und Sichtweisen entstammen. Für den Einsatz eines solchen Systems müssen diese Bedingungen je nach Szenario unterschiedlich gewichtet werden. Diese Anforderungen betreffen neben der Umsetzung des Systems auch die Verfahren, welche zur Erstellung des Entscheidungsmodells eingesetzt werden. Zur Bestimmung geeigneter Verfahren zur Auswertung der Informationen wurden die folgenden Anforderungen von den allgemeinen Anforderungen an das Gesamtsystem aus Abschnitt 2.3 abgeleitet:

Nutzerseitige Anforderungen

Das System hat zum Ziel das Konzept des Nutzers zu lernen und nutzt dazu das Feedback des Nutzers. Der Ablauf erfordert daher eine Interaktion mit den Nutzern (engl. *human-in-the-loop*). Dadurch entstehen weitere Anforderungen:

- **Unbekanntes Nutzerkonzept:** Das Konzept, welches der Nutzer als Grundlage für die Entscheidung (welches das *gewünschte* Ergebnis ist) nutzt, ist dem System nicht bekannt.
- **Korrektheit der Ergebnisse:** Falsche Entscheidungen erzeugen unerwünschte Aktionen. Das Modell muss so gut wie möglich das Verhalten des Nutzers abbilden.
- **Zeitnahe Adaption:** Das Feedback des Nutzers muss möglichst zeitnah vom System aufgenommen und durch das Modell repräsentiert werden.
- **Nutzerkonzeptänderungen:** Veränderungen des Verhaltens eines Nutzers müssen schnell erkannt und vom Modell adaptiert werden (siehe Kapitel 5.3.1).
- **Falsche Angaben:** Bei der Interaktion mit Menschen muss mit falschen Feedback-Angaben gerechnet werden. Ebenso können Informationsquellen falsche, veraltete oder widersprüchliche Daten liefern.
- **Entlastung des Nutzers:** Der Nutzer soll bei seinen Arbeiten unterstützt werden. Das bedeutet, durch den Einsatz des Dienstes soll der Aufwand seitens des Nutzers reduziert werden. Das Ziel der Modelladaption ist es, den Benutzer zu entlasten, indem der Aufwand zur manuellen Anpassung des Modells eingespart wird.

Anforderungen der Informationsgewinnung

Die folgenden Anforderungen entstehen durch die Einbeziehung externer Informationsquellen in der Suche nach relevanten und verfügbaren Informationen:

- **Unbekannte Informationsquellen:** Neue Informationsquellen und deren Formate müssen von dem System gefunden und angewendet werden können.
- **Dynamische Menge von Informationen:** Durch wechselnde Mengen verfügbarer oder relevanter Sensoren muss mit variablen Mengen von Informationen gerechnet werden.
- **Fehlende Informationen:** Mit dem Wechsel der Umgebung, in der sich das Kontext Objekt (z.B. der Nutzer) befindet, werden neue Informationsquellen relevant, während andere wegfallen. Ein Modell, welches auf einer bestimmten Menge von Informationen trainiert wurde, muss gegebenenfalls auch mit einer reduzierten Menge arbeiten können.
- **Abweichungen in der Genauigkeit:** Externe Informationsquellen können beispielsweise durch Messfehler fehlerhafte Aussagen über die Zustände der Merkmale treffen.
- **Änderungen in der Aussage von Informationsquellen:** Betrachtet man ein System über einen längeren Zeitraum, so können sich die Aussagen von Informationsquellen verändern. Beispielsweise kann durch eine Veränderung der Position einer Webcam das betrachtete Areal einer Bewegungserkennung oder durch eine Modifikation der Infrastruktur von Funknetzwerken die Positionserkennung beeinträchtigt werden. Der dabei erzeugte Effekt ist vergleichbar zu dem einer Nutzerkonzeptänderung.

Anwendungsseitige Anforderungen

Je nach Anwendungsszenario und der Anwendung, welche den Kontext nutzen soll, entstehen weitere Anforderungen:

- **Aufwand für die Auswertung des Modells:** Es ist damit zu rechnen, dass die Auswertung eines Modells vergleichsweise häufiger vorkommt, als dessen Neuerstellung. Daher sollte der Aufwand in den Bereich der Modellerstellung verlagert werden. Je nach Anwendungsgebiet des Kontextdienstes ist die Auswertung des Modells mehr oder weniger zeitkritisch. Im Falle des beschriebenen Kommunikationsszenarios gelten

hier teilweise Echtzeitkriterien. Zudem muss neben zur Auswertung des Modells auch der Aufwand für die Informationsgewinnung berücksichtigt werden.

- **Aufwand für die Erstellung und Anpassung des Modells:** Der Aufwand bei der Erstellung ist weniger zeitkritisch, da keine direkte Interaktion mit dem Nutzer erfolgt. Auf der anderen Seite ist es möglich, dass ein Nutzer nach dem Geben eines Feedbacks testet, ob sich das System schon adaptiert hat. Zudem muss der Aufwand gering gehalten werden, um einer Vielzahl von Nutzern den Dienst erbringen zu können oder um auf ressourcenbeschränkten Systemen zum Einsatz zu kommen.
- **Abgeschlossenes System:** Das Zielkonzept sieht vor, den Kontextdienst möglichst wie eine *Black-Box* funktionieren zu lassen. Auf der einen Seite der Black-Box werden Informationen bereitgestellt und auf der anderen Seite können Anfragen zur Auswertung der Informationen gestellt werden, beziehungsweise Feedback gegeben werden kann (siehe Abbildung 5.1). Alle Funktionalitäten innerhalb dieser Black-Box sollten daher so gekapselt sein, dass sie ohne zusätzliche Eingriffe von außen funktionieren.
- **Generisches Konzept:** Der Kontextdienst soll in sehr unterschiedlichen Umgebungen und in verschiedenen Ausprägungen zum Einsatz kommen können. Ziel ist es eine Architektur zu entwickeln, welche zur Auswertung von Informationen in dynamischen und heterogenen Umgebungen dient und für unterschiedliche Anwendungen herangezogen werden kann. Daher wird in dieser Arbeit eine möglichst generische Architektur und Funktionsweise verfolgt und aufgezeigt.
- **Repräsentation des Ergebnisses:** Je nach Anwendungsszenario und der Anwendung, welche letztlich den Kontext nutzt, ist es notwendig, eine geeignete Darstellungsform des Ergebnisses zu ermöglichen. Beispielsweise können je nach Kontextdimension einzelne Kontext-Klassen oder eine Kombination mehrerer Kontextklassen als Resultat erforderlich sein.

Qualitätsbetrachtung

Die zu bewältigenden Herausforderungen maschineller Lernsysteme für den Einsatz in kontextbasierten Kommunikationsdiensten ähneln den Verhaltensweisen eines Menschen in unterschiedlichen Situationen. Da der maschinelle Learner die zuvor vom Nutzer wahrgenommenen Aufgaben übernimmt, ist sein Verhalten diesem anzupassen.

Eine der essenziellen Herausforderungen ergibt sich durch die Geschwindigkeit und die Genauigkeit des Lernsystems bei der Situationserfassung und -beurteilung:

- Eine nahezu perfekte Genauigkeit bei der Situationsbeurteilung ist wertlos, wenn das Ergebnis zu spät eintrifft.
- Ein sehr schnelles Resultat des Learners bei niedriger Genauigkeit ist ebenfalls nicht zielführend.

Das Ziel besteht darin zwischen den Größen Geschwindigkeit und Genauigkeit eine gegenseitige Abstimmung zu ermitteln. Somit gilt es, eine hohe Effektivität bei der Erneuerung, Änderung und dem Ausbau des Verhaltens zu erreichen. Eine lange Lernzeit für den Aufbau oder die Änderung von Verhaltensweisen ist in diesem schnelllebigen Umfeld des Wechsels von Relevanz und Irrelevanz der Lerndaten sehr ineffektiv. Zusammenfassend steht das Lernsystem also vor der Herausforderung der schnellen Reaktion und der Anpassung des Verhaltens an rasant wechselnde Umweltsituationen.

Generell ist die Geschwindigkeit eines Lernsystems bei der Situationsbeurteilung abhängig von der Art und dem Ausmaß der Datenhaltung. Auch die Anpassungsfähigkeit von Verhaltensweisen ist von dieser Spezifikation betroffen. Gerade beim Einsatz der maschinellen Learner auf ressourcenärmeren, mobilen Endgeräten, sind der Datenumfang und die Datenmodellierung von entscheidender Bedeutung für eine hohe Leistungsfähigkeit in realen Szenarien.

5.2 Architektur des Kontextdienstes

Im diesem Abschnitt wird die Architektur des Kontextdienstes im Rahmen der Gesamtarchitektur aufgezeigt und die Abhängigkeiten zu anderen Komponenten des Systems aufgezeigt. Zudem werden die Aufgaben und notwendigen Funktionalitäten zur Integration adaptiver Verfahren erläutert.

Der Kontextdienst kann Entscheidungen treffen, indem verfügbare Informationen aus der Umwelt herangezogen und ausgewertet werden. Die zur Auswertung herangezogenen Informationen basieren auf den Kontextinformationen $\{i\} = I$, welche bei der Informationsgewinnung aus den Sensoren aggregiert wurden.

Wie in Abschnitt 2.4.3 definiert, bestimmt eine Entscheidung die gültige Kontextklasse $k_{d,o,t}$ innerhalb einer Kontextdimension $d \in D$, für ein bestimmtes Kontextobjekt $o \in O$ und zu einem Zeitpunkt t. Die Auswertung

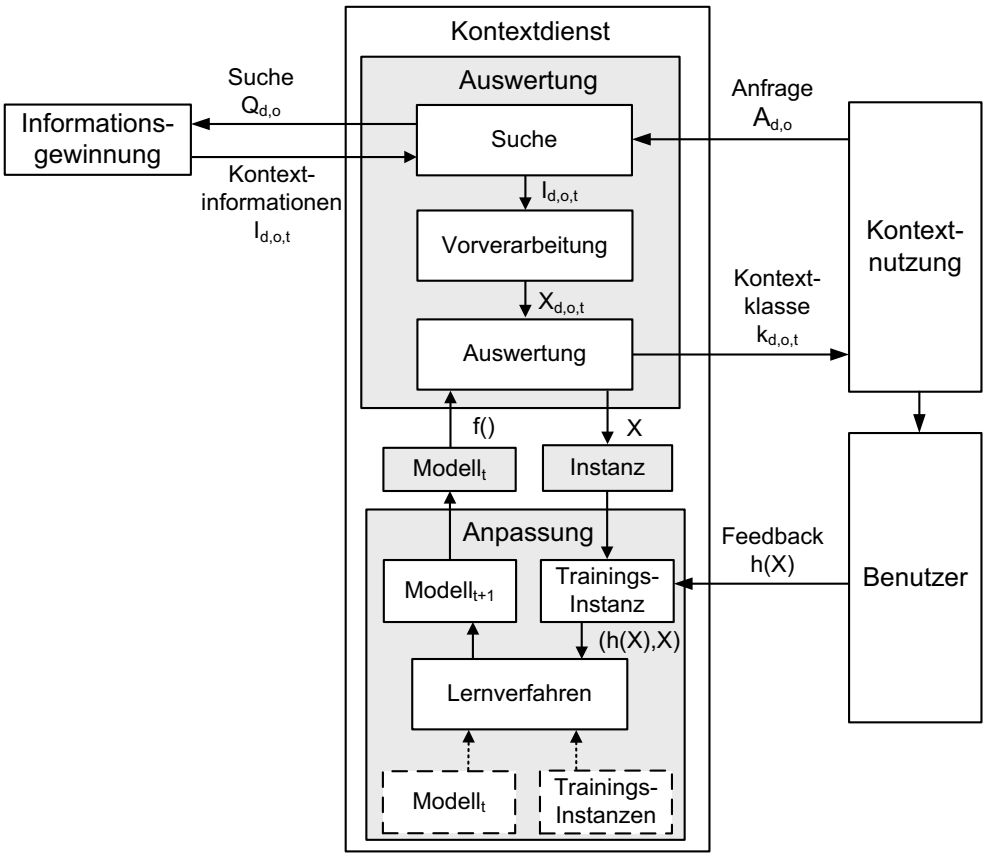


Abbildung 5.4: Auswertung und Anpassung im Kontextdienst

wiederum wird durch ein Entscheidungsmodell durchgeführt, welches durch die Funktion $f()$ beschrieben wird. Dieses Modell soll zur Laufzeit dynamisch angepasst werden können.

5.2.1 Funktionsweise

Wie in Abschnitt 5.1.1 erläutert und begründet wurde, eignet sich für den Kontextdienst ein iterativer Ansatz. Dieser iterative Ansatz ist in Abbildung 5.3 dargestellt und lässt sich in zwei Phasen unterteilen: Die Auswertungsphase und die Anpassungsphase.

Phase 1 - Auswertung

In dieser Phase wird der Kontextdienst genutzt um eine Entscheidung zu treffen (siehe Abbildung 5.4 oben).

Der Auslöser für diese Phase des Kontextdienstes kann beispielsweise eine konkrete Anfrage sein, oder der Kontext eines Kontextobjektes wird periodisch aktualisiert. Im ersten Schritt müssen die Kontextinformationen I gesammelt werden (es sei denn diese Informationen wurden bereits in der Anfrage mitgegeben). Hierzu wird eine Suche über das System zur Informationsgewinnung durchgeführt, die nach Informationen sucht, welche Aussagen zur Kontextdimension d ermöglichen und sich auf das Kontextobjekt o beziehen.

Zur Auswertung dieser Menge an Kontextinformationen $I_{d,o,t}$ müssen diese geeignet vorverarbeitet und in eine für das Auswertungsmodell geeignete Form der Repräsentation X (im Folgenden auch als *Instanz* bezeichnet) überführt werden (siehe Abschnitt 5.3.1).

Es muss möglich sein, die Auswertung von Instanzen in einem eingeschränkten Zeitrahmen (siehe QoS aus Abschnitt 3.4.1) durchzuführen. Dazu ist es notwendig, im Voraus die Abhängigkeiten, die durch die Instanzen beschrieben werden zu finden, zu extrahieren und in Form eines Entscheidungsmodells zu repräsentieren.

Im nächsten Schritt erhält das System zur Auswertung die Menge an vorverarbeiteten Informationen in Form einer Instanz X und ein Entscheidungsmodell $f()$. Das Ergebnis dieser Auswertung entspricht dem ermittelten Kontext des Objektes o in der Dimension d zu der aktuellen, über die Sensoren beobachteten, Situation.

Phase 2 - Anpassung

Das Ergebnis einer Auswertung kann zur Steuerung bzw. zum Auslösen von Aktionen genutzt werden. Die Reaktion des Nutzers auf die gewählte Aktion entspricht $h(X)$ (*hidden concept*) für die zu einer Situation gesammelten Informationen X. Dieses *Feedback* kann anschließend genutzt werden, um das Entscheidungsmodell anzupassen. Je nach Anwendungsgebiet des Kontextdienstes kann das Feedback implizit (durch Beobachten der Reaktion des Nutzers) oder explizit erfolgen (indem der Nutzer das korrigierte Ergebnis selbst auswählt - siehe Kapitel 5.2.2).

Wird zu einer Auswertung aus Phase 1 ein Feedback vom Nutzer gegeben, so kann in der 2. Phase eine Anpassung des Entscheidungsmodells erfolgen (siehe hierzu Abbildung 5.4 unten). Der Wunsch des Nutzers zu einer bestimmten Situation lässt sich abbilden durch die Menge der in dieser Situation gesammelten Informationen X in Kombination mit dem Feedback des Nutzers $h(X)$. Das Tupel $(h(X), X)$ welches sich daraus ergibt, repräsentiert diesen Wunsch des Nutzers und wird als *Trainingsinstanz* bezeichnet. Die Menge der Trainingsinstanzen eines Nutzers spiegelt das Wissen des Dienstes über den Nutzer wider. Je nach Verfahren, welches letztlich zur Erstellung des Entscheidungsmodells genutzt wird, werden zur Erstellung eines angepassten Modells zusätzlich noch die Menge aller bisher gesammelten Trainingsinstanzen oder das bisher verwendete Modell benötigt.

Zur Erstellung des neuen Modells wird schließlich ein Lernverfahren angewendet, welches ein neues, angepasstes Modell $f'()$ (in Abbildung 5.4 dargestellt als $Modell_{t+1}$) bestimmt. Das neu erstellte Modell wird weiterhin als Grundlage für die folgenden Auswertungen in Phase 1 angewendet.

Kombination beider Phasen

Beide Phasen zusammengenommen ergeben wiederum einen Kreislauf. Durch Feedback des Nutzers kann dem Dienst das gewünschte Verhalten antrainiert werden.

Das vorgestellte Prinzip erlaubt die Steuerung des Kontextdienstes durch einfaches Feedbackgeben des Nutzers. Statt das Modell $f()$ manuell zu erzeugen, reicht das Geben von Feedback aus, um komplexe Verhaltensweisen nachzubilden. Dabei besteht ein Feedback aus dem korrigierten Ergebnis $h(X)$, also in der Regel nur der Spezifikation des Wertes, welcher in der entsprechenden Situation als Ergebnis vom Nutzer erwünscht ist (beispielsweise Kontext=*Büro* oder Aktion=*Anrufweiterleitung*).

5.2.2 Feedback

Die Lernverfahren, die für diesen Anwendungsbereich in Frage kommen, gehören zur Klasse der *überwachten* Lernverfahren. Das bedeutet, dass Feedback notwendig ist, um das Verfahren zu trainieren. Dieses Feedback bildet dabei das einzige Element, über das der Nutzer in Kontakt mit dem Kontextdienst tritt, wobei Feedback primär das korrigierte Ergebnis beinhaltet.

Die Umsetzung der Methode, über die der Kontextdienst Feedback erhält, ist vom Einsatzszenario abhängig. Dabei kann zwischen drei Typen von Feedback unterschieden werden:

- **Implizites Feedback:** Erfolgt als direkte Reaktion des Nutzers auf die vom Kontextdienst getroffene Entscheidung. Der Nutzer bestimmt dabei das Ergebnis, welcher er in der aktuellen Situation erwarten würde. In den meisten Szenarien sind dabei jedoch nur begrenzt Eingabemöglichkeiten für das Feedback vorhanden. In einem Szenario, in welchem der Dienst beispielsweise genutzt wird, um ein Gerät zu steuern, indem er es ein- und ausschalten kann, kann das manuelle Betätigen des Schalters als Feedback genutzt werden. Bei dem Telefonie-Szenario können auch Reaktionen wie *Anruf-Abweisen* oder *Nicht-Annehmen* als Feedback verwendet werden.
- **Explizites Feedback:** Diese Variante ermöglicht die Erstellung von Feedback über eine *Historie*. Die Historie gibt Einblick in zurückliegende Entscheidungen des Systems, und der Nutzer kann nachträglich zu diesen Entscheidungen Feedback geben. Diese Art von Feedback erlaubt einen größeren Funktionsumfang, erfordert aber auch, dass der Nutzer über den Zugriff auf entsprechende Geräte verfügt, um die dafür notwendigen Funktionalitäten umzusetzen (beispielsweise grafische Oberfläche oder Spracherkennung).
- **Automatisches Feedback:** Wenn eine Entscheidung gefällt wurde und der Benutzer über einen gewissen Zeitraum keine Einwände dagegen erhoben hat, indem er negatives Feedback abgegeben hat, kann davon ausgegangen werden, dass die Entscheidung korrekt war. Dies kann dazu genutzt werden die Entscheidung

selbst wiederum als Feedback zu betrachten und dazu zu verwenden das gelernte Nutzerkonzept mit weiterem positivem Feedback zu festigen.

5.2.3 Abhängigkeiten des Kontextdienstes

Die Qualität der Entscheidungen des Kontextdienstes ist stark von äußeren Gegebenheiten abhängig.

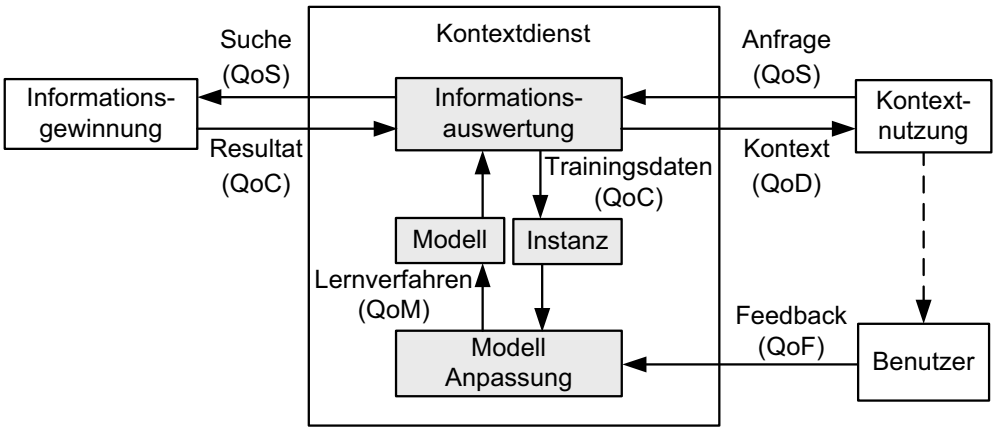


Abbildung 5.5: Abhängigkeiten der Informationsgewinnung

Das Ziel, ein Modell aufzubauen, welches sich möglichst gut mit Konzept des Nutzers deckt (siehe Abschnitt 3.4.4), basiert im Wesentlichen auf:

- Den zeitlichen Rahmenbedingungen in denen die Auswertung und die Suche durchgeführt werden müssen (QoS, siehe Abschnitt 3.4.1).
- Der Qualität der gesammelten Informationen (QoC, siehe Abschnitt 3.4.3).
- Der Qualität des Feedback des Nutzers, welche als Grundlage zum Aufbau des Modells dienen (QoF, siehe Abschnitt 3.4.5).
- Dem Verfahren zur Generierung des Modells, sowie der Aussagekraft und den Eigenschaften des Modells (QoM, siehe Abschnitt 3.4.4.
- Den QoS-Rahmenbedingungen des Systems und der Anwendung. Durch eine Zeitbeschränkung der Suche und Auswertung kann die QoC und die QoD beeinträchtigt werden. Durch eine Beschränkung des maximalen Rechenaufwandes oder Speicherplatzes für die Erstellung des Modells, kann die QoM reduziert werden.

Während die QoC von der Informationsgewinnung, die QoF vom Nutzer und die QoS vom umgebenden System abhängig sind, hängen die letzten beiden Aussagen von der Umsetzung des Kontextdienstes selbst ab (wie in Abbildung 5.5 dargestellt.

Je mehr relevante Informationsquellen gefunden wurden und je mehr relevante Informationen zur Verfügung stehen, desto eher kann eine korrekte Auswertung der Daten erfolgen. Um zumindest die Grundlage für eine korrekte Entscheidung zu haben, ist es notwendig, die Daten mindestens einer relevanten beziehungsweise ausschlaggebenden Informationsquelle vorliegen zu haben. Zu viele irrelevante Informationen führen zudem meist zu einer geringeren Qualität der Entscheidung.

5.2.4 Schnittstelle zum Lernverfahren

In Abschnitt 5.1.2 wurde die Anforderung an das System definiert, ein *generisches Konzept* zu bieten, um den Einsatz in unterschiedlichen Anwendungsszenarien mit unterschiedlichen Anforderungen zu ermöglichen. Je nach Verfahren, welches zur Auswertung, beziehungsweise zur Generierung des Auswertungsmodells herangezogen wird, sind unterschiedliche Eigenschaften im Bereich der Effizienz und Genauigkeit zu erwarten. In diesem Zusammenhang ist es notwendig, je nach Anwendung die Wahl eines geeigneten Verfahrens zur Auswertung der Daten zu ermöglichen.

Wie später in Abschnitt 6.4.1 genauer erläutert wird, wurde hierzu auf eine Reihe verschiedener Lernverfahren aus dem WEKA-System[2] zurückgegriffen.

Nutzerübergreifende Anwendung der Modelle

Das Modell bestimmt das Ergebnis einer Auswertung in Abhängigkeit zu den Kontextinformationen. Der Nutzer passt durch sein Feedback das Modell an sein Konzept an. Das bedeutet, dass ein Modell das Konzept eines Nutzers abbildet. Es stellt sich die grundlegende Frage, ob ein Modell nur speziell auf einen Nutzer angewandt werden kann.

Eine syntaktische Adressierung von Sensoren führt dazu, dass Aussagen verschiedener Sensoren möglicherweise nicht miteinander vergleichbar sind oder es zumindest einer weiteren Instanz bedarf, welches zusätzliches Domänenwissen nutzt um die Sensoren trotz ihrer unterschiedlichen Adressen aufeinander abbildet. Im Rahmen dieser Arbeit wurde ein Ansatz zur semantischen Adressierung von Sensoren entwickelt und umgesetzt (siehe Abschnitt 6.4.1). Sofern die Kontextinformationen semantisch adressiert werden, können diese Aussagen prinzipiell auch von vergleichbaren Sensoren oder gar Sensoren anderer Nutzer bezogen werden. Das Modell wird auf diesen Aussagen basierend aufgebaut und bietet durch die Übertragbarkeit der Aussagen auf andere Kontextobjekte, die Möglichkeit, dass das Modell auch Nutzerübergreifend angewendet werden kann. Haben mehrere Nutzer ein ähnliches Konzept, so können sie gemeinsam ein Modell trainieren. Dies führt zu einer schnelleren Adaption des Gesamtkonzepts der Nutzer und neue Nutzer können von einem bereits existierenden Modell profitieren.

Wie stark sich die Konzepte der Nutzer überdecken, hängt jedoch stark von Anwendungsfall ab. Während die Nutzer einer Lokationsbestimmung meist ein komplett einheitliches Konzept verfolgen, unterscheiden sich die Konzepte bei der Bestimmung der Beziehung zwischen Nutzern stärker. Im letzteren Fall wäre eine Nutzung eines Modells pro Nutzer vorzuziehen, um eine gegenseitige Beeinflussung des Modells auszuschließen. Als initiales Modell kann hierbei auch das Modell eines Nutzers mit möglichst vergleichbarem Konzept herangezogen werden, wie beispielsweise das eines Kollegen.

Als dritte Variante, kann auch eine Mischform beider zuvor genannter Varianten eingesetzt werden, bei dem sich mehrere Nutzer zwar ein gemeinsames Modell nutzen, zusätzlich jedoch der Bezeichner des Nutzers als weiteres Unterscheidungsmerkmal in die Trainingsinstanz integriert wird. Überdecken sich die Konzepte der Nutzer in bestimmten Bereichen, so profitieren sie in diesen Bereichen von dem bereits von den anderen Nutzern gegebenem Feedback. Bei Bereichen, in denen die Nutzerkonzepte widersprüchlich zueinander sind, kann der Bezeichner des Teilnehmers, implizit bei der Generierung des Modells, als Unterscheidungsmerkmal herangezogen werden.

5.2.5 Vorverarbeitung

Beim Einsatz von Lernverfahren spielt die Vorverarbeitung der Daten eine wichtige Rolle. Im Gegensatz zu Datensätzen, wie sie beispielsweise aus Benutzerumfragen gewonnen werden ist die Erfassung von Daten durch Sensoren mit zusätzlichen Herausforderungen verbunden.

So besteht die Gefahr fehlender Wertegrößen durch einen Sensorausfall. Aber auch räumliche und zeitliche Einschränkungen bei der Erfassung von Umgebungsattributen sind Schwachstellen. Ungenauigkeiten bei Messungen oder Unsicherheiten durch Mehrdeutigkeiten der Umgebungsgrößen stellen weitere Risikopotenziale dar. Eine Aufgabe der Vorverarbeitung dient dazu, solche Fehlerquellen zu neutralisieren bzw. zu korrigieren. Hierbei wird nicht nur jeder einzelne Sensorwert, sondern auch die Gesamtaussage der gesammelten Informationen betrachtet. Dadurch sinkt einerseits die potentielle Fehleranfälligkeit, andererseits kann durch die Realisierung von Synergien zwischen diesen Messungen eine Informationswertsteigerung erreicht werden [Gör05]. Sind beispielsweise ein GPS-Empfänger sowie ein Thermometer als Sensoren im Einsatz, so lässt sich durch die getrennte Behandlung die Position und die Temperatur bestimmen. Durch eine Aggregation der beiden Datenquellen wäre es allerdings auch möglich, Fehlerquellen auszuschließen. So wäre eine Kombinationen von 45 Grad Celsius Außentemperatur und dem Aufenthaltsort in der Antarktis wohl kaum gültig. Auch vorzeitige Rückschlüsse durch gewisse Merkmalskombinationen der Sensoren wären denkbar, um den Trainings- bzw. Datenaufwand für die Lernsysteme zu reduzieren. Die Aufgabe der Vorverarbeitung besteht vor allem in der Behandlung folgender Aspekte:

- **Vergleichbarkeit der Werte:** Um Entscheidungsmodelle auszuwerten, müssen die darauf angewendeten Werte mit denjenigen vergleichbar sein, mit denen das Modell erstellt wurde. An dieser Stelle gilt es anhand der Meta-Information zu bestimmen, welche Werte vergleichbar sind und welche durch Umformatierung vergleichbar gemacht werden können.

[2] http://www.cs.waikato.ac.nz/ml/weka/

- **Erweiterung um bekannte Zusammenhänge:** Aus einer Menge von Informationen können höherwertige Aussagen ermittelt werden - analog zu dem Ansatz des *Ontologiebasierten Schließens* aus Abschnitt 5.1.1. Dieses Wissen über die Bedeutung von Sensoren und deren Zusammenhänge kann in Form zusätzlicher Sensoren oder Vorverarbeitungsschritte eingebunden werden.
- **Zeitliche Aspekte:** Einige Zusammenhänge werden erst durch die Betrachtung der Informationen über einen längeren Zeitraum hinweg möglich. Da Sensoren im Allgemeinen nur die Betrachtung des aktuellen Zustandes erlauben, müssen gegebenenfalls weitere Mechanismen hinzugefügt werden, welche die Suchergebnisse mit Aussagen erweitern und auch zeitliche Komponenten berücksichtigen.
- **Aggregieren von Werten:** Durch Aggregation von Informationsquellen können genauere Aussagen getroffen werden (umfangreiche Arbeiten hierzu sind in [Gör05] zu finden). Ebenso kann der Umfang der Datenmenge reduziert werden, was letztlich auch Einfluss auf die Performanz des Systems hat.
- **Ungültige Werte filtern:** Um bessere Ergebnisse zu erzielen, sollten veraltete, ungenaue oder ungültige Werte entfernt werden.
- **Nicht vorhandene Sensorwerte ergänzen:** Je nach eingesetztem Typ eines Entscheidungsmodells können Probleme entstehen, wenn für die Entscheidungen heranzuziehende Informationen nicht verfügbar sind. Eine mögliche Methode um gegen dieses Problem anzugehen, ist das Interpolieren der fehlenden Werte durch vergleichbare Werte aus anderen Trainingsinstanzen. Die Doktorarbeit *Working with Real-World Datasets* [Sch04b] von Schöner bietet hierzu eine detaillierte Betrachtung dieses Themenbereichs.
- **Anpassung der Repräsentation an Lernverfahren:** Es existieren Lernverfahren, welche beispielsweise ausschließlich mit diskreten oder kontinuierlichen Daten arbeiten können. In diesem Zusammenhang ist es notwendig, die gesammelten Daten entsprechend umzuwandeln, welche nicht den Datentypen entsprechen, die verarbeitet werden können.

Integration von Vorverarbeitung in den Datenfluss

Die Vorverarbeitung der Informationen kann direkt zwischen die Suche und die Auswertung der Daten eingebunden werden. Da das Ergebnis variabel ist, müssen jedoch unterschiedliche Mechanismen eingesetzt werden. In der Regel können solche Mechanismen jedoch nur bei bestimmten Kombinationen von Informationen eingesetzt werden. Daher gilt es, die Suchergebnisse auf das Vorkommen solcher Kombination hin zu analysieren.

Mechanismen

Die Mechanismen können je nach Sensor und Anwendungsfall sehr speziell ausfallen. Grundlegend basieren die adressierten Mechanismen auf folgenden Aufgaben:

- **Aggregation:** Gesammelte Werte können kombiniert werden, um beispielsweise eine höhere Genauigkeit zu erzielen.
- **Erweitern:** Manche Aussagen können weiterverarbeitet werden um höherwertige Aussagen zu gewinnen (engl. *Feature Extraction*) oder um die Anwendbarkeit auf Lernverfahren zu optimieren.
- **Transformation:** Informationen, welche unterschiedlichen syntaktischen Typen angehören, jedoch die gleiche semantische Aussage beinhalten, können transformiert werden, um miteinander vergleichbar zu sein.
- **Filter:** Filter können eingesetzt werden, um Aussagen zu überprüfen und gegebenenfalls zu entfernen.

Bedingungen

Je nach der zu verarbeitenden Informationen können die Mechanismen nur unter bestimmten Bedingungen eingesetzt werden. Diese Bedingungen können je nach Information und Mechanismus unterschiedlich aussehen. Letztlich müssen für die Anwendung eines Mechanismus folgende Elemente einer Information betrachten werden:

- **Format:** Um Daten miteinander verarbeiten zu können, müssen die Daten in geeigneten Formaten vorliegen.
- **Semantik:** Die Aussage einer Information bestimmt maßgeblich, mit welchen anderen Aussagen diese verarbeitet werden kann.
- **Zuordnung:** Um Aussagen miteinander zu verarbeiten, muss ebenso die Beziehung zwischen den Objekten betrachtet werden, auf die sich die Aussagen beziehen.
- **Inhalt:** Letztlich bestimmt auch der Inhalt der Aussage selbst, ob ein Mechanismus wie beispielsweise ein Filter angewendet werden soll.

Dynamische Menge von Bedingungen zur Vorverarbeitung

Auf der anderen Seite ist die Menge der Bedingungen und damit die Menge der zu suchenden Kombinationen von Informationen nicht statisch, sondern kann sich ebenfalls zur Laufzeit ändern. Ein Sensor oder eine Anwendung könnte zusätzliche Mechanismen zur Vorverarbeitung in das System einspeisen. Beispielsweise könnte ein Temperatursensor geeignete Mechanismen wie Filter und Aggregation einsetzen, um andere Sensoren wie Helligkeits- oder Rauchsensoren aus der unmittelbaren Umgebung einzubeziehen und somit höherwertige Aussagen wie die Erkennung von Bränden zu ermöglichen.

Ansatz: Regelbasierte Vorverarbeitung

Das zuvor umschriebene Problem der Vorverarbeitung lässt sich verallgemeinert als „Wenn BEDINGUNG X erfüllt, dann wende MECHANISMUS Y an" Aussagen darstellen. Diese Aussagen lassen sich in Form von generisch anwendbaren Regeln formulieren, welche sich einfach erweitern lassen.

Einsatz von Ontologien

Ebenso wie bei der ontologiebasierten Suche aus Abschnitt 4.6.2, kann eine Ontologie zur Vorverarbeitung sinnvoll eingesetzt werden. Der Einsatz einer Ontologie ermöglicht eine Bestimmung der Zusammenhänge von Aussagen beziehungsweise der Relation der Objekte, auf die sich die Aussagen beziehen.

Zum anderen ermöglicht der Einsatz einer Ontologie die Darstellung oder Ableitung von Regeln. Zudem wird die Beschreibung komplexer Zusammenhänge möglich gemacht, wie sie zur Erweiterung der Suchergebnisse um höherwertige Aussagen benötigt wird (siehe auch *Ontologiebasiertes Schließen* 5.1.1).

Beispiele für Regeln zur Vorverarbeitung

Grundlegend besteht eine Regel aus zwei Teilen: Der Bedingung und dem Mechanismus. Die Bedingung kann aus mehreren Elementen (mittels AND/OR) verknüpft werden. Die Tabelle 5.1 bietet eine Reihe von vereinfachten Beispielen zur Veranschaulichung.

Bedingung (S=Sensor)	**Mechanismus**
$I_1.type = phoneBook$ AND $I_2.type = phoneBook$ AND $I_1.owner = I_2.owner$	$Aggregation(join, I_1, I_2)$
$I_1.type = humidity$ AND $I_2.type = humidity$ AND $I_1.roomID = I_2.roomID$	$Aggregation(meanValue, I_1, I_2)$
$I_1.type = smoke$ AND $I_2.type = heat$ AND $Distance(I_1, I_2) < 10\ units$	$AddInformation(fireDetected)$
$I_1.type = location$ AND $I_2.type = location$ AND $I_1.owner = I_2.owner$ AND $I_1.location! = I_2.location$	$Filter(RemoveConflict, I_1, I_2)$
$I_1.type = temperature$ AND $I_2.type = temperature$ AND $I_1.unit! = I_2.unit$	$Transformation(I_1, I_2)$

Tabelle 5.1: Beispiele für Regeln zur Vorverarbeitung

Im ersten Beispiel werden die Inhalte zweier Adressbücher kombiniert, falls sie zum selben Benutzer zugeordnet werden können. Das zweite Beispiel nimmt den Durchschnitt zweier Feuchtigkeitssensoren, falls sie sich im selben Raum befinden. Im dritten Beispiel werden zwei Informationen zu einer höherwertigen Aussage vereint, in diesem Falle führt die Erkennung von Rauch und Hitze zur Aussage, dass womöglich ein Feuer ausgebrochen ist. Das vorletzte Beispiel hebt widersprüchliche Aussagen auf, wie beispielsweise zwei Aussagen über die Position einer Person, welche nicht übereinstimmen. Das letzte Beispiel transformiert das Format einer Aussage (in diesem Fall die eines Temperaturfühlers), so dass sie mit ähnlichen Aussagen anderer Sensoren vergleichbar ist.

5.3 Integration von Lernverfahren

In diesem Abschnitt wird gezielt auf das Thema Lernverfahren eingegangen. Aus dem gegebenen Anwendungsszenario heraus, ergibt sich eine Reihe von Effekten, welche die Lernverfahren vor verschiedene Herausforderungen stellen. Diese Effekte werden erläutert und bestehende Ansätze zur Behandlung aufgezeigt. In diesem Zusammenhang werden die existierenden Lernverfahren kategorisiert, auf ihre Anwendbarkeit hin untersucht und ausgewählt.

Die Erkennung von Mustern in den Trainingsinstanzen und der darauf basierende Modellaufbau zur Klassifikation der Instanzen ist das Ziel des Einsatzes von Lernsystemen (engl. *Learner*). Die Erstellung beziehungsweise

Anpassung eines Modells an das Nutzerkonzept repräsentiert den Wissensaufbau des Systems und ist somit als *Lernen* zu bezeichnen.

Jedes Lernverfahren wird auf seine spezifische Weise ein Modell aufbauen. Unabhängig davon, welche Lernart oder welches Modell nun seine Anwendung findet, wird das, was gelernt werden soll, als *Nutzerkonzept* bezeichnet. Ein Modell kann somit als eine Nutzerkonzeptbeschreibung aufgefasst werden [WF01]. Das Nutzerkonzept (im Folgenden auch als Zielkonzept benannt) eines Learners stellt eine Funktion dar.

Als Eingabeparameter für eine Modell-Auswertung dienen die Werte der *Klassifikationsinstanz*. Die Menge der Attribute (in diesem Falle vergleichbar mit den Sensordaten) einer Instanz spannt den sogenannten *Domänenraum X* auf. Die Ergebnisse einer Modell-Auswertung entsprechen einer oder mehrerer Kontextklassen k einer Kontextdimension. Zur Vereinfachung des Problems wird vorerst von einer einzelnen Kontextklasse als Ergebnis ausgegangen.

Die Konzeptbeschreibung eines Learners kann wie folgt als Funktion dargestellt werden: Sie überführt die Attributwerte $x_1, ..., x_n$ der Klassifikationsinstanzen aus dem Domänenraum X in den eindimensionalen Raum der Klassen $k \in K$.

$$f(x_1, ..., x_n) = k \tag{5.1}$$

Die Grundlage für den Aufbau eines Modells bieten die *Trainingsinstanzen*. Eine Trainingsinstanz besteht aus den Attributwerten der Instanz in Kombination mit der Bezeichnung des gewünschten Resultats.

Zur Beschreibung des Optimierungsziels ist es notwendig, das versteckte Konzept h (hidden concept) einzuführen. Im Zusammenhang mit dem Einsatzszenario für ein Kommunikationssystem kann man das versteckte Konzept auch als *Wunsch des Nutzers* bezeichnen. Dieses, dem Lernsystem unbekannte, Konzept beschreibt die richtige Klassenzugehörigkeit für jede Instanz des Domänenraumes X. Es stellt somit das optimale Konzept dar.

Das Optimierungsziel eines jeden Konzeptes ist in Fällen mit diskreten Kontext-Klassen wie folgt zu formulieren:

$$f(x_1, ..., x_n) = h(x_1, ..., x_n) \qquad \forall (x_1, ..., x_n) \in X \tag{5.2}$$

Im Falle kontinuierlicher Kontext-Klassen kann das Ziel wie folgt formuliert werden:

$$|f(x_1, ..., x_n) - h(x_1, ..., x_n)| \to 0 \qquad \forall (x_1, ..., x_n) \in X \tag{5.3}$$

Das versteckte Konzept ist dem Nutzer bzw. Trainer des Lernsystems bekannt. Sein Bestreben ist es nun, mit einer endlichen Menge von Trainingsinstanzen, die Konzeptbeschreibung möglichst frei von Differenzen an das versteckte Konzept anzunähern. Unter einer Differenz ist hierbei ein von der versteckten Konzeptbeschreibung h abweichendes Ergebnis der Konzeptbeschreibung f zu verstehen.

Im Idealfall beschreibt das Entscheidungsmodell $f()$ das Zielkonzept $h()$. Das bedeutet, dass das Modell nicht nur $f(X) = h(X)$ in möglichst vielen bereits beobachteten Fällen als Ergebnis erhalten soll, sondern auch $f(X') = h(X')$ für neue und bisher unbekannte Situationen X'.

5.3.1 Informationsgrundlage

Die funktionale Abbildung eines Kontextinformation I_i in Abhängigkeit von der Zeit t in ein Element D_i des Domänenraumes D ist wie folgt definiert [BI98]:

$$I_i : t \to D_i \tag{5.4}$$

Der Domänenraum D beinhaltet dabei die Menge von Werten, welche der Sensor nach der Abbildung seiner erfassten Daten annehmen kann. In Abhängigkeit von der Art des Sensors, seines Einsatzgebietes oder Einschränkungen an die zu übermittelnden Daten kann diese Wertemenge endlich oder unendlich sein. Die Elemente des Domänenraumes D sind daher sowohl durch kontinuierliche, als auch durch diskrete Werte bestimmt.

Aus der Kombination von Sensordaten mit Informationen aus der Sensorbeschreibung lassen sich Wertepaare (engl. *Key-Value Pairs*) bilden, welche ein Sensordatum einer (möglichst eindeutigen) Bedeutung zuordnet (siehe Abschnitt 6.4.1). Durch den Einsatz der kontextbasierten Kommunikationsdienste in der realen Umwelt ergeben

sich jedoch weitere Problemstellungen für die zugrunde liegenden Lernsysteme, welche durch eine angemessene Robustheit des Learners ausgeglichen werden müssen.

Irrelevante Werte

Die zu verarbeitende Menge an Sensoren wird durch die semantische Suche auf eine Menge relevanter Sensoren eingeschränkt. Diese Suche betrachtet nur die semantische Beschreibung eines Sensors, um die Relevanz zur aktuell betrachteten Kontextdimension und dem Kontextobjekt zu bestimmen. Ob die Aussage eines Sensorwertes letztlich relevant für die Auswertung des Nutzerkonzeptes ist, kann damit jedoch nicht entschieden werden.

Bei der Suche wird somit meist auch eine Menge von Sensoren erfasst, welche keine Relevanz für das Konzept des Nutzers haben. Diese Menge kann je nach Szenario, installierten Sensoren, Sensorbeschreibung oder der Umsetzung der Suche stark variieren.

Ansätze

Ein Ansatz besteht darin, eine Relevanzbestimmung im Bezug zu dem Feedback des Nutzers durchzuführen. Bestimmt man die Korrelation zwischen den Sensordaten und den Klassenwerten innerhalb der gesammelten Trainingsdaten oder betrachtet man das Modell, um dort die Relevanz eines Sensors zu bestimmen, so kann die Suche auf relevante Sensoren beschränkt werden. Zum Zeitpunkt der Auswertung ist dies ein sinnvoller Ansatz, da zudem auch die Suche verkürzt werde kann. Daher wird dieser Ansatz in zukünftigen Arbeiten beim Fachgebiet KOM weiter bearbeitet. Zum Zeitpunkt der Adaption müsste jedoch weiterhin eine Suche über alle potentiell relevanten Sensoren durchgeführt werden, da es sonst dazu führt, dass neue Trainingsinstanzen nur bereits adressierte Sensorwerte beinhalten und somit das Modell keine neuen Sensoren aufnehmen und je nach Relevanz für das Nutzerkonzept adaptieren könnte.

Fehlerhafte Informationen

Fehler in den Kontextdaten, beispielsweise erzeugt durch Übertragungsfehler oder Umwelteinflüsse auf die Sensoren, stellt Lernverfahren vor eine schwierige Herausforderung [CN87, HK01]. Sie müssen entsprechend differenzieren, ob es sich bei den Trainingsinstanzen nur um einzelne Fehler handelt oder ein Wechsel des Kontexts beginnt. Entscheidet sich das System, eine Instanz als fehlerhaft zu markieren und nicht in den Modellaufbau zu integrieren, so fördert es zwar seine Robustheit, jedoch verliert es an seiner Fähigkeit zur frühzeitigen Erkennung einer *Konzeptänderung* (dieser Effekt wird im Verlauf dieses Abschnittes genauer erläutert). Festzuhalten ist somit, dass sich die Robustheit eines Lernsystems gegenüber Fehlern und die Wirksamkeit der Erfassung von Konzeptänderungen gegenseitig bedingen.

Des Weiteren beeinflusst die Rate der Fehler ebenfalls die Lernfähigkeit. Ein n prozentiger Fehler eines Attributes bedeutet, dass n Prozent der Attributwerte abweichend oder unabhängig vom Nutzerkonzept gewählt wird [WK96]. Die Auswirkung ist allerdings unter Beachtung der Attributart (diskret oder kontinuierlich) sowie der Ausprägung des Nutzerkonzeptes zu vollziehen. So verliert ein Fehler von 100% bei einem Nutzerkonzept, welches vier von fünf möglichen diskreten Werten eines Attributs beinhaltet, an Bedeutung, da nur 20% der Zufallsziehung eine Zuordnung der Instanzen zu einem anderen Nutzerkonzept bewirken.[3]

Ansätze

Brodley und Friedl [BF99] führen eine generische Form der Erkennung von fehlerhaften Informationen ein (*outlier-detection*), welches in Form eines Vorverarbeitungsschrittes auf die Trainingsinstanzen angewendet werden kann. Ein ähnliches Vorgehen wird von Frank und Witten ebenfalls vorgeschlagen [FW00]. Han und Kamber diskutieren vier Methoden zur Glättung zur Behandlung von Fehlern vor. Unter anderem verwenden sie hierbei Ansätze wie *binning* und *clustering* [HK01].

Fehlende Werte

Lücken in der Informationsgrundlage eröffnen weitere Herausforderungen für die Lernsysteme. Fehlende Werte können beispielsweise durch Sensorausfälle oder verspätete Suchergebnisse hervorgerufen werden. Bedingt durch

[3] Diese Betrachtung erfolgt unabhängig von dem möglichen Einfluss der Kombination von Werten verschiedener Attribute.

die Beschränkungen der Suche und den sich wechselnden Umgebungen in denen nach relevanten gesucht wird, können zu einer Vielzahl der Sensoren aus den Trainingsdaten keine vergleichbaren Sensoren aus der Umgebung bezogen werden.

Im Vergleich zu fehlerhaften Attributwerten bieten fehlende Werte dem Lernsystem eine höhere Sicherheit, da sie im Gegensatz zu Abweichungen eindeutig identifiziert werden können. So ist es möglich nicht vorhandene Werte durch Prognosen zu ermitteln oder sie einfach bei dem Aufbau des Klassifikationsmodells zu ignorieren.

Das Wegbleiben eines Wertes bedingt nicht notwendigerweise eine Verschlechterung der Aussagekraft einer Trainingsinstanz. So kann beispielsweise ein Laptop über das WLAN mit dem eigenen Heimnetz verbunden sein. Verliert er diese Verbindung, so kann sich dies einerseits durch einen zufallsbedingten Ausfall des WLANs ereignen, andererseits ist es aber auch möglich, dass der Laptop außerhalb der Reichweite positioniert wurde und damit in eine neue Situation gerät. Der Wegfall eines entsprechenden Sensorwertes kann als Aussage zur Beschreibung dieser neuen Situation genutzt werden.

Ansätze

Es existieren einige Ansätze, um mit fehlenden Werten umzugehen. Diese Ansätze reichen vom kompletten Entfernen von Instanzen mit fehlenden Attributen, bis hin zu Ansätzen, welche fehlende Werte mit möglichst wahrscheinlichen Werten füllen oder gar wiederum Lernverfahren anwenden, um diese Werte zu ermitteln [HK01]. Anhand der Ergebnisse von verschiedenen Arbeiten aus der Literatur kann sich jedoch keine Methode vollständig von den anderen hervorheben [JH01]. Im Rahmen dieser Arbeit wird der zweite Ansatz testweise angewendet und die Auswirkungen verglichen (siehe Abschnitt 5.4.7).

Fehlerhaftes Feedback

Fehlerhaftes Feedback beschreibt Trainingsinstanzen deren assoziierte Kontextklasse nicht mit dem eigentlichen Konzept des Nutzers übereinstimmt $(X, h'(X))|h'(X) \neq h(X)$. Zum einen können solche Trainingsinstanzen durch versehentliche Falschangaben seitens des Nutzers entstehen. Zum anderen können solche Effekte auch bei automatisierten Vorgängen entstehen, beispielsweise wenn das System vorsieht, automatisch (positives) Feedback zu generieren (siehe Abschnitt 5.2.2). Eine solche automatische Generierung von positivem Feedback kann durchgeführt werden, indem Instanzen, welche nach einem gewissen Zeitraum nicht vom Nutzer als falsch identifiziert oder korrigiert wurden, automatisch mit der damaligen Entscheidung des Modells versehen $(X, f(X))$ und als Feedback verwendet werden.

Konzeptänderung

Das Konzept beschreibt das vom Nutzer gewünschte Verhalten des Systems. Ändern sich die Wünsche, das Verhalten oder die Umgebung des Teilnehmers auf die das System trainiert wurde, so entspricht das neue Ziel-Konzept nicht mehr dem gelernten Modell. Diese entstehenden Konzeptwechsel werden in der Literatur auch als *Konzeptdrift* (engl. *Conceptdrift*) bezeichnet [Tsy04, CJK+01].

Problem: Over-Training

Im Zusammenhang mit Konzeptänderungen entsteht ein neues Problem – das sogenannte *Over-Training*. Dieser Begriff beschreibt das Phänomen, dass je länger ein Modell auf ein stabiles Konzept trainiert worden ist, es umso länger dauert es, einen Wechsel des Konzeptes durch das Modell zu adaptieren [WK96].

Ansätze zur Erkennung von Konzeptänderungen

Die Erkennung von Konzeptänderungen kann genutzt werden, um die Generierung des Modells zu beeinflussen und um gegebenenfalls Maßnahmen zu ergreifen und die Konzeptänderung schnell durch das Modell zu adaptieren. Zur Erfassung von Konzeptänderungen bedienen sich Klingenberg und Renz drei verschiedener Indikatoren [KR98]:

- **Betrachtung der Genauigkeit des Modells:** Basierend auf der aktuellen Klassifikationsgenauigkeit des Zielkonzeptes kann bestimmt werden, ob sich das Nutzerkonzept verändert hat. Sinkt die durchschnittliche Genauigkeit plötzlich ab, so kann ein Konzeptwechsel die Ursache dafür sein.
- **Eigenschaften des Klassifikationsmodells:** Beispielsweise über die Erfassung der Komplexität des Nutzerkonzeptes oder die Anzahl der Klassifikationsregeln. Ändert sich das Konzept eines Nutzers, so reichen die

Aussagen oder Regeln des Konzeptes möglicherweise nicht mehr aus, um das neue Verhalten abzudecken. Entsprechend wird von dem Lernverfahren die Menge der Aussagen erweitert. Gerade bei Systemen, welche eine Generalisierung vornehmen, ist die Anzahl der Aussagen, welche das Konzept beschreiben ein guter Indikator, ob das Konzept des Nutzers erfasst werden kann.

- **Charakteristika der Trainingsinstanzen:** Eine Analyse der Verteilung der Attributwerte oder der Veränderungen in ihrer Gruppierung kann ebenfalls herangezogen werden, um eine Konzeptveränderung anzuzeigen. Ein neues Konzept führt zu Trainingsinstanzen, welche im Allgemeinen vermehrt andere oder neue Sensorwerte beinhalten. Somit kann über Verfahren wie beispielsweise Clustering bestimmt werden, ob die neuen Trainingsinstanzen zu den in der Vergangenheit gesammelten Instanzen *passen*.

Gerade das Auftreten von fehlerhaftem Feedback kann die Erkennung von Konzeptänderungen beeinflussen und zu Fehlern in der Erkennung führen.

Klassifizierung von Konzeptdrift und Konzeptshift

Man kann die Konzeptveränderungen nach der Stärke ihrer Ausprägung, sowie nach ihrer Schnelligkeit klassifizieren [CGM03]. Je größer die Konzeptveränderung ist, desto mehr Aufwand zur Variation des Konzeptes muss ein Lernsystem erbringen, um die korrekte Klassifikation der veränderten Kontextdaten zu gewährleisten.

Von einer Konzeptveränderung kann das gesamte Konzept oder aber nur ein Teil des Konzeptes betroffen sein. Der Abstand beziehungsweise die Differenz zweier Konzepte beschreibt wie stark die Konzeptveränderung ausgeprägt ist.

Widmer und Kubat definieren die Geschwindigkeit einer Konzeptänderung (speed of drift) und differenzieren somit zwischen rapiden und langsamen bzw. trägen Übergängen zwischen Konzepten. Die Geschwindigkeit wird dabei durch eine Funktion α festgelegt. α beschreibt die Wahrscheinlichkeit, dass das aktuell gültige Nutzerkonzept noch mit den alten Kontextdaten übereinstimmt. Entsprechend beschreibt $\alpha = 1$ die volle Gültigkeit des alten Konzeptes A, während $\alpha = 0$ die komplette Übernahme des neuen Konzeptes B beschreibt. Die folgende Abbildung 5.6 veranschaulicht diese Geschwindigkeitsfunktion beim Wechsel des Kontextes.

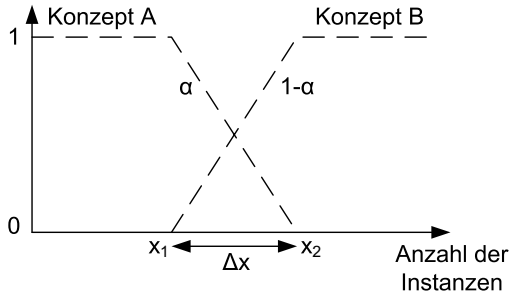


Abbildung 5.6: Geschwindigkeitsfunktion eines Konzeptdrifts [WK96].

Nach einer Anzahl von x_1 Beispielen tritt das alte Konzept in die Wechselphase ein und benötigt entsprechend Δx Trainingsinstanzen, um einen kompletten Übergang zum neuen Konzept zu bewirken. Der Wert der Variable α definiert somit die Anzahl benötigter Trainingsinstanzen für den Übergang. Zu jedem Lernzeitpunkt werden folglich $\alpha * 100\%$ der Instanzen bezüglich des alten Konzeptes und $(1 - \alpha) \cdot 100\%$ gemäß dem neuen Konzept klassifiziert [WK96].

Anhand dieser Messgrößen kann zwischen einem Konzeptdrift und einem Konzeptshift differenziert werden:

- *Konzeptdrift:* Ein kontinuierlich und langsam verlaufender Konzeptwechsel. Dieser beinhaltet in der Regel ein geringes Ausmaß der Konzeptverschiebung (extend of drift) sowie eine niedrige Änderungsrate (rate of drift).
- *Konzeptshift:* Ein abrupt und sprunghaft auftretender Wechsel des Konzeptes. Im Allgemeinen bedeutet dies ein höheres Ausmaß des Wechsels und eine ebenfalls höhere Rate der Konzeptänderungen [All96].

In dieser Arbeit werden beide Effekte als Konzeptänderung mit jeweils unterschiedlich starker Ausprägung bezeichnet.

Wiederkehrende Konzepte

Wiederkehrende Konzepte sind in realen Domänen häufig anzutreffen. Oft beruhen solche scheinbar wiederkehrenden Konzepte jedoch auf fehlenden Informationen. So kann das Konzept eines Nutzers für das Lernverfahren, wie zwei gänzlich unterschiedliche Konzepte wirken, sobald ein Attribut zur Differenzierung zwischen den beiden Konzept-Teilen fehlt. In Abbildung 3.7 aus Abschnitt 3.4.4 wurde diese Problematik skizziert. Wenn das fehlende Attribut oder Merkmal beispielsweise die Lokation der Person beschreibt, die Person den Raum wechselt und dort ein entsprechend anderes Verhalten aufweist, so kann diese Situation für das Lernverfahren fälschlicherweise als Konzeptwechsel gedeutet werden.

Ein ständig neues Erlernen dieser wiederkehrenden Konzepte ist wohl die einfachste, aber auch ineffizienteste Lösung. Entsprechend bietet es sich an, eindeutig identifizierbare Lernzustände mit hoher Qualität aufzubewahren und bei erneutem Bedarf wieder zu laden [HH98, WK96].

Die Herausforderung besteht nun einerseits in der Identifikation akzeptabler Konzepte, welche für eine Abspeicherung in Frage kommen. Andererseits ist es notwendig bereits gespeicherte Konzepte zu analysieren, um diese eventuell zu laden und damit Teile des aktuellen Zielkonzeptes zu ersetzen.

Zur Lösung dieser Problematiken sind wiederum heuristische Verfahren hilfreich. Allerdings ist darauf zu achten, dass nur gezielt ausgewählte Konzepte aufbewahrt werden. Speichert eine Heuristik zu viele Konzepte ab, ergibt sich daraus ein erheblicher Performanceverlust des Lernsystems durch den erhöhten Ressourcenaufwand. So wird in allen gängigen Lernsystemen, die ein solches *Langzeitgedächtnis* unterstützen, bei wiederkehrenden Konzepten eine Suche im *Gedächtnis* vollzogen [Koy01, Koy00a]. Auf der anderen Seite erzeugt ein solches Langzeitgedächtnis zusätzlichen Aufwand in Form von Speicherplatz zur Hinterlegung von Modellen, als auch Rechenkapazität zum Vergleich des aktuellen Modells mit gespeicherten zur Bestimmung, ob sich das *alte* Modell besser auf das aktuelle Konzept anwenden lässt.

5.3.2 Klassifizierung von Lernverfahren

Durch die verschiedenen Anwendungsgebiete und Anforderungen an maschinelle Lernsysteme haben sich im Laufe der Zeit unterschiedliche Kategorien entwickelt. Die Einteilung der Lernverfahren in Kategorien, sowie die Auswahl geeigneter Kategorien wird im folgenden Teil behandelt und ist in Abbildung 5.7 skizziert.

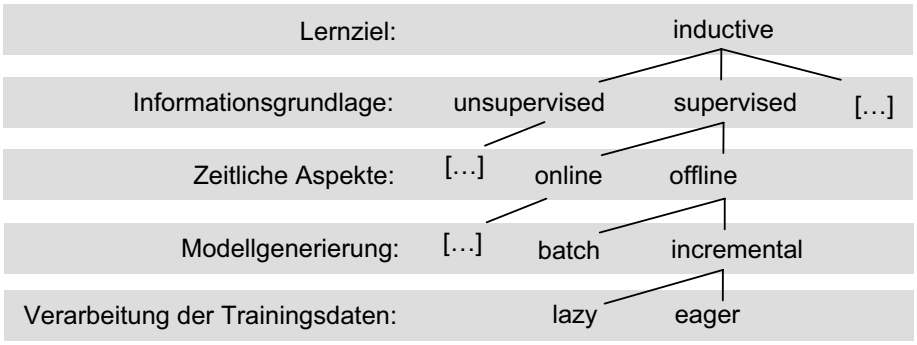


Abbildung 5.7: Klassifizierung von Lernverfahren

Lernziel

Bei der Einteilung in Kategorien kann grundsätzlich zwischen den folgenden *Lernzielen* unterschieden werden:

- **Klassifizierendes Lernen:** Verfahren dieser Kategorie analysieren die Zusammenhänge zwischen Attributen und Klassen in einer Menge von Trainingsinstanzen, um Instanzen mit unbekannten Klassen zu klassifizieren.
- **Assoziierendes Lernen:** Das assoziierende Lernen beachtet Assoziationen zwischen allen Attributen der Instanzen, inklusive des Klassenattributes.
- **Einteilendes Lernen:** Die Gruppierung von Instanzen findet im Clustering statt. Hierzu werden Instanzen zu Gruppen zugeordnet, welche sich durch Häufungen von Instanzen im Domänenraum ergeben.
- **Bestärkendes Lernen:** Verfahren dieser Kategorie bestimmen ein Modell auf der Basis von binärem Feedback (richtig/falsch), jedoch ohne direkten Bezug zu den Informationen.

- **Vorhersagendes Lernen:** Das vorhersagende Lernen bestimmt wahrscheinliche Abfolgen von Zuständen, beziehungsweise Klassen.
- **Regressionsanalyse:** Die Regressionsanalyse hat nicht die Prognose von Klassenwerten zum Ziel, sondern die Erstellung einer Funktion welche die erwarteten Ausgangswerte approximiert.

Für diese Arbeit müssen Kontextklassen bestimmt werden. Diese Aufgabe wird von klassifizierenden Lernverfahren durchgeführt. Verfahren, welche Zusammenhänge zwischen Attributen und Klassen anhand von Trainingsinstanzen ermitteln, werden auch als *induktive Lernverfahren* bezeichnet. *Vorhersagendes Lernen* ist für die Anwendung zur Bestimmung des aktuellen Kontextes nicht konzipiert. Jedoch bietet es Raum für zukünftige Erweiterungen des Systems, um beispielsweise proaktiv Aktionen durchzuführen. Interessante Arbeiten hierzu liefern Mehlhase [Meh08] und Ferscha et. al. [FMR03].

Informationsgrundlage

Eine weitere Klassifikation kann anhand der *Informationsgrundlage* durchgeführt werden. Die Informationsgrundlage kann aus verschiedenen Elementen bestehen:

- **Instanz:** Eine Instanz repräsentiert die gesammelten Informationen und beinhaltet gegebenenfalls folgende Elemente:
  - **Attribute:** Die Attribute beinhalten die Kontextinformationen.
  - **Klassenattribute:** Die Klassenattribute beinhalten die Kontextklassen.
- **Trainingsmenge:** Die Trainingsmenge beinhaltet eine Menge von Instanzen (aus vorangegangenen Iterationen).
- **Modell:** Das Modell beschreibt das bisher erlernte Konzept.

Die verschiedenen Lernziele erfordern bestimmte Informationsgrundlagen. Die Informationsgrundlage kann aus Instanzen bestehen, welche Attribute mit oder ohne Klassenattributen beinhalten oder wie im Falle des bestärkenden Lernens nur aus binären Wahr-Falsch-Aussagen.

- **Unsupervised-Learner:** Verfahren der Kategorie assoziierende Learner und einteilende Learner benötigen dagegen keine überwachende Instanz, weshalb sie auch als *unsupervised-Learner* bezeichnet werden. Bei einem einteilenden Learner wird häufig eine Vorgabe für die gesuchte Anzahl von Clustern und deren Bezeichnungen benötigt.
- **Supervised-Learner:** Das klassifizierende Lernen benötigt Instanzen mit Attributen und Klassenattributen. Klassenattribute werden von einer Entität geliefert, welche das System trainiert beziehungsweise überwacht. Diese Instanz wird auch als *Supervisor* bezeichnet. Daher werden Algorithmen, welche einen Supervisor zum Training benötigen auch als *Überwachtes Lernen* (engl. *supervised learning*) bezeichnet.
- **Active-Learner, Semi-Supervised-Learner:** Das Geben von Feedback bedeutet Aufwand, beziehungsweise Kosten für den Nutzer. Semi-Supervised-Learner haben zum Ziel die Menge vom benötigten Feedback zu reduzieren, indem beispielsweise nur für kritische Fälle Feedback erfragt wird. Das Problem bei dem Ansatz der Active-Learner besteht in dem angestrebten iterativen Ansatz. Ohne die Betrachtung aller Instanzen über den gesamten Lernzeitraum hinweg können die Grenzen zwischen den Kontexträumen (wo die kritischen Fälle zu suchen wären) nur schwer bestimmt werden.

Ziel dieser Arbeit ist es, ein Verfahren anzubieten, welches das Feedback des Nutzers zur Anpassung der Auswertung nutzt. Diese Vorgabe erlaubt die Nutzung überwachter Lernverfahren.

Zeitliche Aspekte der Informationsgrundlage

Aufgaben im Bereich des Data-Minings behandeln häufig die Analyse großer Datenmengen, wie beispielsweise einer Datenbank mit Kundendaten. Die Reihenfolge der einzelnen Datensätze hat in solchen Aufgabenbereichen keine Relevanz. Verfahren, welche für diesen Aufgabenbereich entworfen wurden, behandeln daher alle Instanzen gleich, unabhängig von der Reihenfolge in welcher sie verarbeitet werden. Im Allgemeinen wird das Modell im Voraus *offline* generiert und später zur Auswertung genutzt.

Andere Anwendungsszenarien erfordern eine inkrementelle Vorgehensweise. Durch die Zuführung von Trainings- und Klassifikationsinstanzen in einer beliebigen Reihenfolge, ist das Lernverfahren dazu gezwungen seine Entscheidungen bzw. Klassifikationen *online* zu bestimmen. Die Vorteilhaftigkeit zeigt sich also durch den gleichzeitigen Trainings- und Klassifikationseinsatz sowie in der Schnelligkeit von Konzeptänderungen [Ome00].

Anhand dieses Merkmals können die Lernverfahren weiter differenziert werden:

- **Offline-Learner:** Sie verarbeiten Instanzen ohne Bezug zu zeitlichen Aspekten oder Reihenfolge. Alle Instanzen haben somit die gleiche Aussagekraft.
- **Online-Learner:** Die Reihenfolge der Verarbeitung der Instanzen beeinflusst maßgeblich das resultierende Modell. *Neue* Instanzen haben dadurch eine höhere Relevanz als diejenigen, die zuvor verarbeitet wurden.

Beide Varianten gehen jedoch davon aus, dass alle Aussagen innerhalb einer Instanz eine Situation zu einem bestimmten Zeitpunkt beschreiben. Andere Ansätze berücksichtigen auch zeitliche Abhängigkeiten innerhalb einer Instanz. So ist beispielsweise bei Sprach- oder Schrifterkennung die Reihenfolge der Laute oder Zeichen wichtig, um das resultierende Wort zu ermitteln. In der Literatur ist dieses Problem auch unter der Bezeichnung *Sequential Supervised-Learning* zu finden.

Ablauf der Modellgenerierung

Ein weiteres Merkmal beschäftigt sich mit der Frage, ob das Lernsystem einen Modellaufbau durch jeweils *einzelne* Trainingsinstanzen ermöglicht oder ob zuerst *alle* Instanzen zum Training verarbeitet werden müssen und darauffolgend das Modell erstellt.

- **Batch-Learner:** Vollzieht eine vorherige Verarbeitung aller Trainingsinstanzen.
- **Incremental-Learner:** Involviert jede Instanz einzeln in den Modellaufbau.

Die folgende Tabelle 5.2 stellt die Verfahrensweisen der beiden Abläufe zur Modellgenerierung gegenüber.

Batch-Learner	**Incremental-Learner**
1. Initialisieren des Klassifikationsmodells 2. Verarbeiten *aller* Traininginstanzen 3. Erstellen des Klassifikationsmodells	1. Initialisieren des Klassifikationsmodells 2. Verarbeiten *einer* Traininginstanz 3. Erstellen, Ändern des Klassifikationsmodells 4. Gehe zu Schritt 2, falls noch weitere Trainingsinstanzen vorhanden sind

Tabelle 5.2: Verfahrensweise von Batch- und Inkrementellen Lernverfahren. In Anlehnung an [Sar04].

Verarbeitung der Trainingsdaten

Es gibt zwei grundlegende Vorangehensweisen, bei der Verarbeitung der Trainingsdaten:

- **Lazy-Learner:** Werten die Trainingsdaten erst zum Zeitpunkt der Anfrage aus.
- **Eager-Learner:** Erstellen ein Modell aus den Trainingsdaten, welches zum Anfragezeitpunkt herangezogen wird.

Ein bekanntes Beispiel für einen Lazy-Learner ist das *NEARESTNEIGHBOUR*-Verfahren. Bei einem solchen Ansatz werden die gesammelten Trainingsinstanzen zum Anfragezeitpunkt nach möglichst vergleichbaren Instanzen durchsucht. Bei einer Vielzahl von Instanzen und Sensoren spannt sich ein hochdimensionaler Raum auf. Der Aufwand der Auswertung wächst linear abhängig von der Anzahl der Instanzen und Sensoren.

Je nach Anwendungsszenario und den dabei zu erwartenden Mengen von Instanzen und Sensoren kann es notwendig werden, im Vorfeld der Auswertung eine Generalisierung der Trainingsdaten durch ein Modell vorzunehmen. Die einzelnen Instanzen werden dann nicht mehr als vereinzelte Datenpunkte im Domänenraum angesehen, sondern bilden einen Teilraum. Umschließt ein solcher Teilraum eine zu klassifizierende Instanz, so erhält diese den entsprechenden Klassenwert der Trainingsinstanzen. Bei der Generalisierung handelt es sich im Allgemeinen um heuristische Verfahren. Daher kann nicht garantiert werden, dass maschinelle Lernsysteme unter der Verwendung einer Generalisierung die optimale Konzeptbeschreibung liefern [WF01]. Die Ermittlung einer ressourcensparenden und gleichzeitig wirkungsvoll arbeitenden Generalisierungsfunktion ist daher ein wichtiger Bestandteil bei der Entwicklung von Lernsystemen.

5.3.3 Auswahl relevanter Kategorien von Lernverfahren

Im vorangegangenen Abschnitt wurden von Lernverfahren hinsichtlich ihrer Eigenschaften und ihrer der Anwendbarkeit innerhalb des angestrebten Systems kategorisiert. Dabei wurden Kategorien aufgrund ihrer Eigenschaften ausgeschlossen. In diesem Abschnitt werden diejenigen Kategorien genauer betrachtet, die grundsätzlich für den Einsatz innerhalb dieser Arbeit geeignet sind.

Offline-Learner

Die Möglichkeit der Interpretation der gesamten Menge an Trainingsdaten verschafft Offline-Learnern im Gegenteil zu Online-Learnern einen entscheidenden Vorteil. So ist es ihnen gestattet, mit globalen Optimierungsmechanismen präzisere Modelle zur Klassifikation zu ermitteln. Beispielsweise lassen sich, basierend auf allen Trainingsinstanzen, mehrere Generalisierungsfunktionen austesten und mittels einer Kreuzvalidierung überprüfen, um jene mit der höchsten Genauigkeit der Klassifikation zu bestimmen. Der Trainingsfortschritt ist bei Offline-Learnern also direkt nachweisbar [Sar04].

Offline-Batch

Die generelle Verfahrensweise von Offline-Batch-Learnern ist einfach. Alle verfügbaren Trainingsinstanzen werden analysiert bzw. verarbeitet und zum Modellaufbau genutzt. Eine der bekanntesten Ausprägungen dieser Offline-Learner sind das Entscheidungsbaumverfahren ID3 [Ome00] oder die Support Vector Machinen (SVM).

Offline-Inkrementell

Im Gegensatz zu den Batch-Learner induzieren die inkrementellen Offline-Learner jede Trainingsinstanz, gemäß ihrer Lernreihenfolge, einzeln. Somit sind ihnen, zwar bereits verarbeitete Instanzen bekannt und können in die Modellbildung mit einbezogen werden, jedoch fehlt ihnen die Möglichkeit auf die gesamte Menge an Trainingsbeispielen zurückzugreifen. Hierdurch entsteht ein erheblicher Nachteil gegenüber den Batch-Learnern. Jedoch ist es ihnen durch die Anpassung des Klassifikationsmodells anhand einzelner Instanzen möglich, Rechenzeit zu sparen. Der Hauptvorteil der inkrementellen Lernsysteme ergibt sich allerdings durch die Möglichkeit, während des Lernens Klassifikationen vorzunehmen. Sie sind im Gegensatz zu Batch-Learnern nicht gezwungen, vor der ersten Klassifikation alle Traininginstanzen zu verarbeiten. Beispiele für inkrementelle Offline-Verfahren sind ID5 (Erweiterung des ID3-Entscheidungsbaumverfahrens, um die inkrementelle Lernfähigkeit [AMB+94]) oder NAIVE BAYES.

Eigenschaften

Dem Vorteil der höheren Genauigkeit von Offline-Lernsystemen steht der hohe Ressourcenverbrauch gegenüber. Somit ist es nötig, eine stetig steigende Menge an Trainingsinstanzen zu verwalten. Zusätzlich erhöht sich beim Einsatz von Batch-Learnern die Rechenzeit erheblich, da bei jedem erneuten Modellaufbau alle Instanzen involviert sind. Gerade im Zusammenhang mit Konzeptänderungen spielen zeitliche Aspekte eine zentrale Rolle. Offline-Verfahren unterstützen die Behandlung von Konzeptänderungen nicht von sich aus. Angenommen eine Menge von x Instanzen, welche das Konzept h beschreiben, wurden zum Aufbau eines entsprechenden Modells genutzt und ein Konzeptshift zu Konzept h' hat stattgefunden. Je nachdem wie stark sich die Konzepte im Domänenraum überlagern benötigt ein Offline-Learner in der Regel bis zu x Instanzen des Konzeptes h' in der Trainingsmenge (welche zudem noch *ungültig* gewordene Instanzen beinhaltet), um Konzept h' zu adaptieren. Dies bedeutet, ein Benutzer müsste sehr häufig Feedback geben, bis sein neues Konzept adaptiert werden würde, was den Aufwand seitens des Nutzers stark erhöhen würde.

Mögliche Formen der Integration

Aus diesem Grund sind Verfahren aus der Kategorie der Offline-Learner nicht direkt einsetzbar. Allerdings stammt eine Vielzahl aktuell verfügbarer Verfahren aus dieser Kategorie. Zudem sind die verfügbaren Offline-Verfahren häufig ausgereifter, wodurch eine höhere Qualität der Modelle zu erwarten ist als durch verfügbare Verfahren aus der Kategorie der Online-Learner.

Offline-Verfahren können jedoch um zusätzliche Mechanismen ergänzt werden, so dass zeitliche Aspekte berücksichtigt werden können. Mechanismen, welche dabei Verwendung finden, basieren meist auf Fenster- und Gewichtungsverfahren. Diese Mechanismen können in Form eines *Meta-Verfahrens* integriert werden, welche das eigentliche Lernverfahren kapseln. In Kombination mit einem solchen Meta-Verfahren können Offline-Learner um

die Fähigkeit, Konzeptänderungen zu berücksichtigen erweitert werden und können somit als Ansatz zur Erstellung von Modellen herangezogen werden.

Meta-Verfahren zur Berücksichtigung zeitlicher Aspekte

Bestehenden Ansätze in dem Bereich sehen vor, die Offline-Verfahren in sogenannte Meta-Verfahren zu kapseln. Meta-Verfahren wie beispielsweise *MetaL* oder *Winnow* ergänzen dabei das eigentliche Verfahren um Fenster- oder Gewichtungstechniken [Wid97, MM00b, Lit88]. Fenstertechniken lassen alte Trainingsinstanzen ausscheiden, da nur Instanzen innerhalb eines Fensters zur Modellerstellung herangezogen werden. Erweiterungen bei den Fenstertechniken erlauben die Variation der Fenstergröße abhängig von der Stabilität des Konzeptes. Gewichtungstechniken erlauben das *Altern* von Werten [Koy00b]. Ältere Werte werden dabei schwächer gewichtet als neuere. Die Gewichte werden anschließend bei der Modellerstellung berücksichtigt. Eine weitere Variante besteht in der Nutzung von weiteren Lernverfahren innerhalb des Meta-Verfahrens zur Bestimmung der aktuellen Genauigkeit. Setzt man zusätzlich ein Lernverfahren ein, welches sich schnell an das aktuelle Konzept anpasst, so kann dessen Genauigkeit als Indikator für Änderungen im Konzept genutzt werden [BM08].

Die Techniken lassen sich auch kombinieren, führen aber nur bedingt zum Ziel. Da alte Trainingsinstanzen durchaus noch gültig und relevant sein können, ist mit diesen Techniken das eigentliche Problem, ungültige Instanzen zu entfernen, noch nicht gelöst.

Online-Lernverfahren

Eine grundlegende Eigenschaft der Online-Learner ist der richtige Umgang mit Konzeptänderungen. Diese Änderungen des Konzeptes, welche sich in den neuen Trainingsinstanzen wiederfindet, erfordert im Zeitverlauf eine ständige Neuanpassung des Modells.

Die bekanntesten Verfahren sind STAGGER, IB2 und die FLOating Rough Approximation (FLORA) Familie. Online-Lernverfahren sind bisher nur selten im Einsatz - daher sind Implementierungen und Erfahrungswerte aus realistischen Einsatzszenarien rar.

Der grundlegende Ansatz von Online-Lernverfahren sieht vor, als Informationsgrundlage keine vollständige Menge aller Trainingsinstanzen zum Aufbau eines neuen Modells zu nutzen. Stattdessen greifen Online-Lernverfahren gegebenenfalls auf einen Konzeptspeicher und/oder einen partiellen Instanzenspeicher zurück [Sar04].

Für Systeme mit der Möglichkeit, Trainingsbeispiele zu speichern, ist es daher erforderlich, eine Instanzverwaltung zu etablieren, um die Menge an gespeicherten Instanzen auf eine gewisse, aber endliche Menge zu beschränken. Diese extrahiert Instanzen aus dem Speicher und entfernt sie aus dem Lernsystem [Mal04].

Neu hinzukommende Instanzen lassen sich mit bereits induzierten Instanzen nicht mehr vergleichen, so dass eine Anpassung nur anhand des bestehenden Klassifikationsmodells möglich ist. Entsprechend sind diese Systeme oft unzuverlässiger, jedoch lassen sich diese Schwierigkeiten durch die Involvierung statistischer Größen und die Anwendung von Heuristiken teilweise ausgleichen [Sar04]. Bekannte Ansätze für Online-Lernverfahren, sowie deren Kategorisierung anhand dieser Merkmale werden in Abschnitt 5.6.1 erläutert.

5.4 Analyse bestehender Verfahren zur Informationsauswertung

Im vorherigen Abschnitt wurden die Kategorien von Verfahren betrachtet, welche in dem angestrebten System prinzipiell zur Anwendung kommen können. In diesem Abschnitt werden bestehende Verfahren aus dem Bereich Offline-Lernverfahren herangezogen und analysiert. Da keine Verfahren aus den Bereichen der Meta- und der Online-Lernverfahren zur Verfügung standen, deren Implementierung einen direkten Vergleich ermöglichte, werden in den Abschnitten 5.5 und 5.6 geeignete Ansätze aus diesen Bereichen ausgewählt und dahingehend erweitert und umgesetzt, um mit dem bestehenden Verfahren vergleichbar zu sein.

Bei der Umsetzung des Kontextdienstes steht eine Reihe von Lernverfahren zur Auswahl. Jedes Lernverfahren und dessen Modell hat andere Eigenschaften im Bezug auf Aufwand zur Modellerstellung bzw. Modellauswertung, Robustheit, Speicherbedarf oder der verwendbaren Datenformate. Welches Verfahren am besten geeignet ist, ist stark abhängig von den Rahmenbedingungen des jeweiligen Einsatzszenarios. Dieser Abschnitt beinhaltet die Entwicklung eines Testaufbaus, welcher sich zur Analyse der Verfahren eignet, sowie die Durchführung und die

Ergebnisse dieser Analysen. Die Ergebnisse einer Vielzahl von Tests werden zur Bewertung der Verfahren im Bezug auf die verschiedenen Rahmenbedingungen genutzt.

Aus den allgemeinen Anforderungen auf Abschnitt 5.1.2 lassen sich die folgenden Anforderungen für Lernverfahren ableiten:

1. Hohe Qualität des Modells, gute Adaption des Nutzerkonzeptes.
2. Auswertung unter Einhaltung von Zeitvorgaben.
3. Möglichst geringe Systemanforderungen für die Generierung des Modells.
4. Robustheit gegenüber den Eigenschaften der Informationsgrundlage.
5. Verarbeitung von diskreten und kontinuierlichen Werten.
6. Benutzerdefinierbare Menge an Kontextklassen.
7. Schnelle Anpassung an sich ändernde Konzepte.

5.4.1 Ansatz und Durchführung der Analyse

Im nächsten Schritt sollen die Verfahren auf die zuvor beschriebenen Anforderungen hin untersucht und verglichen werden. Hierzu muss ein Ansatz für die Durchführung der Analyse gewählt werden.

Ansatz aus dem Bereich Offline-Lernverfahren

Im Bereich des Data-Minings werden zur Analyse von Verfahren im Allgemeinen komplette Trainings-Datensätze betrachtet. Die Instanzen dieser Datensätze sind zeitlich unabhängig voneinander und können in beliebiger Reihenfolge auf das Lernverfahren angewandt werden. Um in diesem Zusammenhang valide Aussagen treffen zu können werden meist Verfahren wie die *Ten-fold Crossvalidation* genutzt. Dieses Verfahren teilt die Trainingsmenge in zehn gleich große Teile. Daraufhin werden zehn Durchläufe gestartet. In jedem Durchlauf werden neun der Teilmengen zum Aufbau des Modells genutzt und die verbleibende Trainingsmenge als Testdatensatz, zur Bestimmung der Genauigkeit des Modells. Der Vorteil dieser Technik liegt in der Vermeidung zufälliger Clusterbildung, da jede der Partitionen bzw. deren Instanzen mindestens einmal zum Klassifizieren und neunmal zum Training genutzt werden.

Notwendige Anpassungen

Der Vergleich der verschiedenen Verfahren soll zeigen, welche Verfahren für das angestrebte, iterative Lernen geeignet sind. In diesem Zusammenhang kann eine klassische Vorgehensweise wie die Ten-fold Crossvalidation nicht direkt angewandt werden. Zum einen sollen das Laufzeitverhalten und die Genauigkeit der Modelle über den gesamten Zeitraum hinweg analysiert werden. Zum anderen sind die Testdaten zeitlichen Abhängigkeiten unterworfen, wie beispielsweise den Zeitpunkten des Auftretens der Konzeptänderung.

Ansatz aus dem Bereich Online-Lernverfahren

Das STAGGER-Konzept [SG86] ist in der gängigen Literatur ein Standard zum Vergleich von Online-Lernverfahren im Zusammenhang mit dem Auftreten von Konzeptänderungen [KM03]. Das mit dem Lernsystem STAGGER entwickelte Konzept basiert auf einem Domänenraum mit drei nominalen Attributen. Es gibt zudem drei binäre Zielkonzepte mit fest vorgegebener Reihenfolge [WK96]. Anhand dieser Vorgaben lässt sich erkennen, dass es sich bei dem STAGGER-Konzept um ein sehr einfaches Lernproblem handelt. So sind insgesamt nur 27 verschiedene Attributkombinationen bzw. Instanzen möglich und entsprechend schnell kann das Konzept erlernt werden. Bei der Durchführung der Analyse anhand des STAGGER-Konzeptes werden von jedem Zielkonzept nacheinander (entsprechend der Reihenfolge der Zielkonzepte) jeweils 40 Trainingsinstanzen generiert und einzeln auf das Lernverfahren angewendet. Zur Bestimmung der Klassifikationsgenauigkeit werden anschließend weitere 100 zufällig generierte Testbeispiele klassifiziert.

Notwendige Anpassungen

Im Gegensatz zu dem Ten-fold Crossvalidation Ansatz berücksichtigt das STAGGER-Konzept zeitliche Abhängigkeiten in den Kontextdaten und es ermöglicht die Bestimmung der Klassifikationsgenauigkeit auch während der Trainingsphase. Die Anforderungen sehen jedoch nicht nur die Verarbeitung von diskreten, sondern auch die Verarbeitung von *kontinuierlichen* Werten. Ebenso wird eine benutzerdefinierbare Menge an Kontextklassen vorausgesetzt. Zudem müssen weitere Eigenschaften wie beispielsweise fehlerhafte oder fehlende Werte, sowie die Skalierbarkeit in der Anzahl der Sensoren innerhalb des zu betrachtenden Lernproblems adressiert werden.

Aus diesen Gründen wurde ein auf die benötigten Anforderungen angepasstes Vorgehen zur Generierung von Testdaten und zur Durchführung der Analyse der Lernverfahren entworfen, welches in den folgenden Abschnitten beschrieben wird.

Betrachtung der Genauigkeit

In den Arbeiten, welche mit dem STAGGER-Konzept arbeiten, wird die durchschnittliche Wahrscheinlichkeit die richtige Klasse als Ergebnis zu erhalten als Maßstab für die Genauigkeit herangezogen. Hierzu werden in jeder Iteration eine bestimmte Menge von beliebig gewählten Testinstanzen generiert und klassifiziert. Die relative Menge der richtig klassifizierten Testinstanzen entspricht der Wahrscheinlichkeit mit der eine korrekte Klassifikation im Bezug auf den gesamten Domänenraum zu erwarten ist.

Andere Arbeiten nutzen hierbei auch den Ansatz von *Precision* und *Recall*. Diese Werte geben Aufschluss darüber, wie viele Instanzen einer Klasse korrekt klassifiziert wurden (Recall) und wie viele Instanzen zusätzlich zu dieser Klasse zugeordnet wurden, obwohl sie eigentlich einer anderen Klasse angehören (Precision). In diesem Zusammenhang werden diese Werte meist für jede Klasse einzeln ermittelt, wodurch Aussagen über die Klassifikationsgenauigkeit innerhalb jeder Klasse möglich werden. Das sogenannte F-Maß wird dabei meist als Metrik angewandt, welches Precision und Recall vereint, indem es das harmonische Mittel zwischen beiden Werten bildet.

Im Rahmen dieser Analyse werden die Konzepte künstlich generiert, wobei alle Klassen eines Konzeptes gleichermaßen zufällig erzeugt werden. Daher wird in diesem Fall keine Aussage über die Genauigkeit innerhalb einzelner Klassen benötigt. In dem Testszenario wird eine Instanz, welche fälschlich zu einer Klasse zugeordnet wurde und entsprechend dessen Precision-Wert beeinflusst, gleichzeitig einer anderen Klasse *nicht* zugeordnet, was sich wiederum in dessen Recall-Wert niederschlägt. In dieser Analyse wird eine Aussage über die Wahrscheinlichkeit einer korrekten Klassifikation innerhalb des gesamten Konzeptes benötigt. Hierzu wird die Genauigkeit analog zu dem Vorgehen bei den Arbeiten, welche mit dem STAGGER-Konzept arbeiten, berechnet, welches der Betrachtung des durchschnittlichen Recall-Wertes über das Gesamtkonzept hinweg entspricht.

Testdaten

Die Herausforderung bei dem Vergleich bestehender Verfahren besteht grundlegend in der Bereitstellung geeigneter Testdaten. Die Testdaten besitzen eine Vielzahl von Charakteristiken[4]:

1. Anzahl der Instanzen in der Trainingsmenge.
2. Anzahl der Sensoren innerhalb der Instanzen der Trainingsmenge.
3. Menge kontinuierlicher (KS) und diskreter Sensoren (DS).
4. Größe des Wertebereiches der diskreten Sensoren (SD).
5. Anteil nicht verfügbarer Sensorwerte (PM).
6. Anteil von Sensoren, mit fehlerbehafteten Messwerten (PB).

Um Testdaten zu erhalten, welche diesen Eigenschaften in konkret vordefinierten Wertebereichen unterliegen, wurde ein Testdatengenerator entworfen und umgesetzt. Der Aufwand und die Genauigkeit bei der Modellgenerierung hängen im Allgemeinen von der Beschaffenheit der Sensordaten ab. Die Sensordaten können daher nicht beliebig gesetzt werden, sondern müssen im Bezug auf ein Nutzerkonzept erstellt werden. Dieses Konzept wird ebenfalls generiert und ist abhängig von den folgenden Eigenschaften:

1. Größe der Domäne der Klassenwerte (CA), beziehungsweise Anzahl der Klassen.
2. Menge der Sensoren, die für eine Klasse relevant sind (SR).
3. Wahrscheinlichkeit für fehlerhaftes Feedback (PF).
4. Häufigkeit des Auftretens von Konzeptänderungen(CD).
5. Umfang der von der Konzeptänderung beeinflussten Klassenwerte (DA).

Alle diese Eigenschaften werden als Eingabeparameter zur Generierung von Testdaten verwendet.

[4] Die angegebenen Abkürzungen finden sich bei den nachfolgenden Darstellungen wieder und dienen der Beschreibung der in den jeweiligen Szenarien eingesetzten Parametern.

Durchführung

Die Durchführung des Vergleichs läuft wie folgt ab (für den jeweiligen Schritt sind die relevanten Parameter in Klammern angegeben):

1. **Generierung der verfügbaren Sensoren (KS,DS,SD):** Hierbei werden für eine Menge von Sensoren der Datentyp (kontinuierlich oder diskret) und die Wertebereiche (Domäne) festgelegt.
2. **Generierung des Konzeptes** $h()$ **(CA,SR):** Es wird Konzept erstellt, welches CA verschiedene Klassen aufweist. Zudem wird für jede Klasse festgelegt, welche Sensoren relevante Informationen bereithalten und welche Werte ein relevanter Sensor innerhalb einer Klasse einnehmen kann.
3. **Starte mit leerer Trainingsmenge und *leeren* Modellen:** Ohne eine Trainingsmenge ist das Modell ebenfalls leer und somit kann keine Aussage getroffen werden.
4. **Auswahl einer Klasse** k**:** Jede Instanz soll später zu einer Klasse zugeordnet werden können.
5. **Erstellung einer Instanz** X **mit** $h(X) = k$ **(PM,PB):** Hierzu werden die Sensorwerte generiert. Jeder relevante Sensor nimmt einen Wert an, welcher innerhalb der Klasse gültig ist (bezogen auf die in den Schritten 2 und 3 festgelegten Werte).
6. **Klassifizierung der Instanz** $f(X) = k'$ **mit allen Modellen:** Die generierte Instanz wird mit den Modellen ausgewertet. Wobei deren Auswertungsgeschwindigkeit gemessen wird.
7. **Vergleich** $k = k'$**:** Die Ergebnisse des Modells werden mit denen des generierten Konzeptes verglichen. Hierbei kann bestimmt werden, ob das Modell, innerhalb der in den Schritten 4 und 5 simulierten Situation, mit der Entscheidung des Konzeptes übereinstimmt.
8. **10 malige Wiederholung der Schritte 4 bis 7.**
9. **Bestimmung der durchschnittlichen Genauigkeit des Modells:** Um Aussagen über die durchschnittliche Genauigkeit in der aktuellen Iteration zu erhalten werden 10 Klassifikationen durchgeführt und die durchschnittliche Genauigkeit betrachtet.
10. **Erstellung einer Instanz** X **mit** $h(X) = k$ **(PM,PB):** Eine Situation wird herausgegriffen, um als Feedback herangezogen zu werden.
11. **Erweitere Instanz um Feedback** $(X, h(X))$ **(PF):** Zu dem Feedback wird das Resultat hinzugefügt, welches dem Konzept entspricht.
12. **Füge die Instanz der Trainingsmenge hinzu.**
13. **Erstellung der neuen Modelle** f' **auf Basis der erweiterten Trainingsmenge:** Die verschiedenen zu analysierendem Lernverfahren werden angewendet, um neue Modelle zu generieren. Hierbei kann der Aufwand der Modellgenerierung gemessen werden.
14. **Gegebenenfalls Modifikation des Konzeptes (CD,DA):** Das Konzept wird bei Auftreten einer Konzeptänderung modifiziert, um ein sich änderndes Benutzerverhalten nachzubilden.
15. **Fahre bei Schritt 4 fort bis 250 Instanzen in der Trainingsmenge enthalten sind.**
16. **Speichere Messdaten.**
17. **10 malige Wiederholung der Schritte 1 bis 16.**
18. **Ermittlung der durchschnittlichen Messwerte:** Alle Messwerte werden über 10 Durchläufe gemittelt. Dies ermöglicht eine Aussage das durchschnittliche Verhalten der Lernverfahren und deren Vergleich.

Zum einen werden bei der Durchführung der Analyse das Laufzeitverhalten und der Aufwand zur Erstellung und zur Auswertung des Modells betrachtet. Zum anderen wird die Genauigkeit der Modelle über den gesamten Zeitraum hinweg ausgewertet.

Wahl der Parameter, Basisszenario

Die nächste Aufgabe besteht in der Wahl der Werte für die Parameter für ein *Basisszenario*. Da die Menge der adressierbaren Anwendungsszenarien möglichst breit aufgestellt sein soll, können einige Parameter nicht fest vorgegeben werden. Andererseits ist eine Analyse über alle Kombinationen von Parametern nicht durchführbar. Für den Vergleich bestehender Verfahren wurde daher eine Reihe von Szenarien erstellt, bei denen jeweils der Einfluss einzelner Parameter analysiert wird.

Alle anderen Parameter werden im Allgemeinen auf *feste* Werte gesetzt. Als Grundlage für die Festlegung dieser festen Parameter wurde ein durchschnittliches, zu erwartendes Anwendungsszenario betrachtet.

Die zu verarbeitende Menge an Sensoren muss nicht mit der Menge an verfügbaren Sensoren übereinstimmen. Durch die semantische Suche wird die Menge an zu verarbeitenden Sensoren in der Regel auf die Menge relevanter

Sensoren eingeschränkt. Daher geht es eher um die Betrachtung der Sensoren, welche sowohl für die betrachtete Kontextdimension und das Kontextobjekt relevant sind. Die Menge an Sensoren, auf die dies zutrifft kann schwer vorhergesagt werden. In den betrachteten Szenarien handelte es sich jedoch meist um Größen von ca. 10 bis 50 Sensoren. Daher wurden 32 Sensoren als fester Parameter für die Anzahl von Sensoren gewählt, welche sich zudem zu gleichen Teilen auf kontinuierliche ($KS = 16$) und diskrete Sensoren ($DS = 16$) aufteilen. Für die diskreten Sensoren wurde ein Wertebereich von 16 verschiedenen Einträgen festgelegt ($SD = 16$).

Bei der Suche werden jedoch auch Sensoren erfasst, welche keine Relevanz für das Konzept des Nutzers haben. Dies kann je nach Szenario, installierten Sensoren, Sensorbeschreibung oder der Umsetzung der Suche stark variieren. An dieser Stelle wurde festgelegt, dass jeweils nur eine Menge von 40% der Sensoren für das aktuelle Konzept relevant ist. Das bedeutet nur 40% der Aussagen einer Trainingsinstanz sind abhängig von der Kontextklasse – alle anderen Sensorwerte sind gleich verteilt über ihren jeweiligen Wertebereich ($SR = 40$).

Bei der Bestimmung von Kontextklassen geht es in der Regel um die Differenzierung zwischen einer relativ kleinen Menge an Kontextklassen. Bei der Bestimmung der Presence-Zustände einer Person um die Differenzierung zwischen Kontextklassen wie beispielsweise VERFÜGBAR, ABWESEND, BESCHÄFTIGT, oder aber bei der Steuerung von Geräten wie beispielsweise Licht um die Zustände EIN, AUS. Daher wurde hier eine Menge von 4 Kontextklassen angenommen ($CA = 4$).

Die zu analysierenden Effekte, wie fehlende Sensorwerte PM, fehlerhafte Sensorwerte PB, fehlerhaftes Feedback PF oder Konzeptänderungen CD, CA werden jeweils einzeln betrachtet. In diesem Zusammenhang werden alle anderen Effekte ausgeblendet und bei der Generierung der Daten nicht herangezogen.

Einschränkungen durch Generierung der Daten

Eine kritische Betrachtung dieses Ansatzes des Vergleichs über generierte Daten zeigt auch nachteilige Eigenschaften. So unterscheidet sich ein Vergleich anhand generierter Daten von einem Vergleich anhand von Daten, welche aus der realen Umgebung bezogen wurden. In der Realität sind die Sensordaten und das Auftreten von Kontextklassen nicht gleich verteilt. Gerade bei numerischen Attributwerten ist häufig eine Normalverteilung anzutreffen.

In der Realität können die gesammelten Daten weitaus komplexere Eigenschaften aufweisen, wie beispielsweise die Häufung von Instanzen in bestimmten Situationen, in dem es dem Nutzer einfacher möglich ist, Feedback zu geben. Die Sensoren können zudem statistische Abhängigkeiten untereinander aufweisen.

Die Genauigkeit eines Modells ist zudem abhängig davon, wie ähnlich die zu klassifizierende Instanz den Instanzen aus der Trainingsmenge ist. Bei dem gewählten Vorgehen werden die Beispiele zufällig gezogen, was die Wahrscheinlichkeit eine ähnliche Instanz zu ziehen abhängig von der Größe des Domänenraumes macht. In den angestrebten Szenarien ist zu erwarten, dass das Feedback und die Auswertung in ähnlichen Situationen stattfinden, weshalb die zu klassifizierende Instanz mit höherer Wahrscheinlichkeit den bisher erhaltenen Trainingsinstanzen ähnelt.

Die Generierung der Daten dient nur der Veranschaulichung des Verhaltens der Verfahren auf verschiedene vordefinierbare Eigenschaften in der Trainingsmenge und der Konzepte. Szenarien mit ähnlichen, vordefinierten Eigenschaften wären in der Realität zudem kaum zu konstruieren.

5.4.2 Auswahl der Verfahren

In Abschnitt 5.3.3 wurde eine Reihe von Kategorien von Lernverfahren betrachtet und anhand ihrer grundlegenden Eigenschaften bewertet. Zu jeder Kategorie, welche sich dabei als geeignet herausstellte, existiert eine Reihe von Verfahren. Für den Vergleich der konkreten Eigenschaften und des Laufzeitverhaltens wurden nun eine Reihe bekannter Verfahren mit unterschiedlichen Ansätzen gewählt:

NAIVE BAYES

NAIVE BAYES ist ein Lernverfahren, welches auf der statistischen Entscheidungstheorie beruht. Nach dem Bayestheorem wird eine zu klassifizierende Instanz jener Klasse zugeordnet, der es mit der höchsten Wahrscheinlichkeit angehört. Ausgangspunkt für eine solche Aussage stellt dabei die Annahme, dass eine Zuordnung einer Instanz zu

einer Klasse durch die Dichtefunktion einer Verteilung ausgedrückt werden kann [Kre05]. Das Bayestheorem greift dabei auf den Satz der bedingten Wahrscheinlichkeit zurück:

$$P(A|B) = \frac{P(B|A)}{P(B)} * P(A)$$

Dieser Satz kann genutzt werden, um die Wahrscheinlich für ein Ereignis A zu berechnen, unter der Annahme, dass ein Ereignis B stattgefunden hat. Wird als B die aktuelle Situation X herangezogen, entspricht A der zu ermittelnden Kontext-Klasse k_i. Der Ansatz von NAIVE BAYES beruht auf der Annahme, dass die Kontextinformationen $x \in X$ unabhängig voneinander sind und sich zu einer vollständigen Wahrscheinlichkeit ergänzen:

$$P(X|k_i) = \prod_{x \in X} P(x|k_i)$$

Somit lässt sich die Klassenzugehörigkeit k_i mit dem Bayestheorem wie folgt ermitteln [LW00, WF01]:

$$P(k_i|X) = \frac{\prod_{x \in X} P(x|k_i)}{\prod_{x \in X} P(x)} * P(k_i)$$

Der Eingabewert X entspricht den auszuwertenden Kontextinformationen. Das Ergebnis der Auswertung ergibt die Wahrscheinlichkeit, dass eine Klasse k_i gültig ist.

Diese Aussage lässt sich durch die Auswertung der Trainingsdaten wie folgt ermitteln:

- $P(x|k_i)$ beschreibt die relative Häufigkeit in der das Attribut x im Zusammenhang mit der Klasse k_i beobachtet wurde.
- $P(x)$ ist die relative Häufigkeit des Auftretens eine Attributes.
- $P(k_i)$ entspricht der relativen Häufigkeit des Klassenwertes k_i.

Anhand der Trainingsdaten können diese Werte im Vorfeld berechnet werden (beispielsweise im diskreten Fall durch Abzählen). Die dabei erhaltenen Werte bilden das Modell, welches zur Auswertung herangezogen wird.

Wie bereits erwähnt, setzt NAIVE BAYES voraus, dass die Ereignisse untereinander unabhängig sind, d.h. die Attribute sind einzig und allein vom Klassenattribut abhängig. In der Realität ist die Annahme, dass die Kontextinformationen unabhängig voneinander sind, in den seltensten Fällen zu erfüllen, wodurch das Lernsystem häufig in die Kritik gerät. Doch sind in der Praxis viele Beispiele vorhanden, in denen NAIVE BAYES erfolgreich eingesetzt wird, wie beispielsweise bei E-Mail-Filtern zur Erkennung von Spam-Mails [MAP00].

Entscheidungsbaum (J48)

Entscheidungsbaumverfahren bilden eine hierarchische Anordnung von Entscheidungsregeln. Dabei stellen die Knoten dieser Baumstruktur die jeweiligen Regeln dar. Je nach getroffener Entscheidung in den Knoten wird eine der Verbindungen zum nächst tiefer liegenden Knoten verfolgt. Die Knoten auf der untersten Ebene (Blätter) beinhalten das Entscheidungsergebnis. Die Regeln in jedem Knoten sind so aufgebaut, dass nur jeweils eine eindimensionale Entscheidung, d.h. anhand eines Attributwertes, getroffen wird. Nachteilig an einer solchen baumartigen Strukturierung der Regeln ist die Tatsache, dass einmal fälschlicherweise eingeschlagene Wege nicht mehr korrigiert werden können. Besonders entscheidend sind dabei inkorrekt aufgestellte Regeln auf höheren Ebenen des Baumes oder fehlende Attribute, welche zur Auswertung eines Knotens benötigt werden.

Ein Entscheidungsbaum ist im Allgemeinen sehr effizient in der Auswertung. So müssen nur die Knoten überprüft werden, die auf dem ausgewählten Weg von der obersten zur untersten Ebene liegen. Der Aufwand beschränkt sich bei einer ausgeglichenen Struktur auf $O(\log_m n)$, wobei n die Anzahl der Knoten und m den Verzweigungsgrad angibt. Des Weiteren bildet die einfache Interpretation solcher Baumstrukturen im Gegensatz zu beispielsweise neuronalen Netzen einen erheblichen Nutzen bei der Auswertung [Koh96].

Bekannte Implementierungen dieses Ansatzes sind J48 (eine Java Implementierung von C4.5, welches wiederum eine Erweiterung des ID3 Ansatzes ist) und der Naive-Bayes-Tree (NB TREE), welcher zusätzlich in den Blättern (reduzierte) Naive-Bayes Modelle zur Auswertung beinhaltet[5].

[5] Mit dem *Hoeffding Tree* existiert auch ein inkrementelles Verfahren zur Generierung von Entscheidungsbäumen. Dieses Verfahren ist jedoch erst seit kurzem als Plugin für die WEKA-Erweiterung *MOA* (`http://www.cs.waikato.ac.nz/~abifet/MOA/`) verfügbar.

Neuronales Netz (NEURALNET)

Das künstliche neuronale Netz ist dem Verhalten echter Neuronen nachempfunden. Simulierte Neuronen reagieren auf bestimmte Reize und leiten diese Reize an andere Neuronen weiter, falls die eingehenden Reize stark genug waren. Hierzu werden die Eingänge der Neuronen gewichtet und aufsummiert. Überschreitet die Summe der eingehenden Reize einen Schwellwert, so leitet das Neuron den Reiz weiter. Durch das Trainieren des Netzes werden diese Werte so variiert, dass die zur richtigen Klassifikation nötigen Neuronen durch einen Reiz aktiviert werden.

Als Eingangsneuronen dienen die Attribute der Instanz. Jedoch beinhalten künstliche neuronale Netze nicht nur einen Ausgangsknoten bzw. -neuron, sondern jeweils so viele, wie Klassen im Konzept enthalten sind. Als Resultat wird dasjenige Ausgangsneuron, beziehungsweise dessen Klassenwert ausgewählt, welches den stärksten Reiz aufweist. Das *Multilayer Perzeptron* (MLP) beinhaltet mehrlagig angeordnete Neuronen (die sogenannten *Perzeptrons*). Zwischen Ausgangs- und Endknoten existiert mindestens eine weitere Ebene mit Neuronen (siehe Abbildung 5.8) [Rie94].

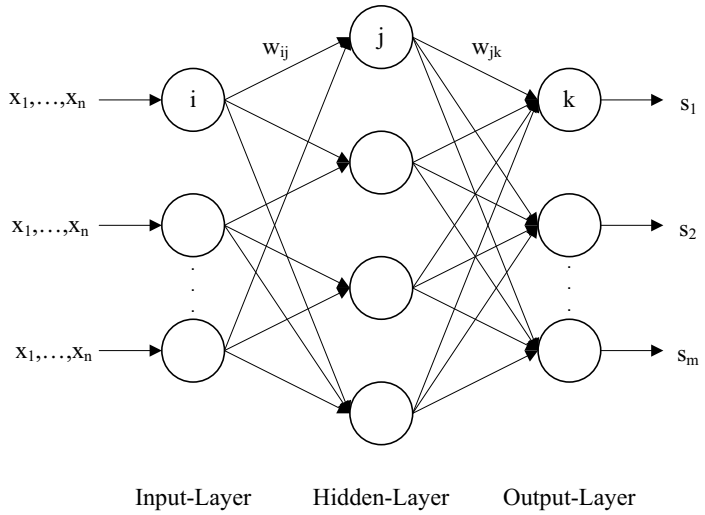


Abbildung 5.8: Struktur eines Mulitlayer Perzeptron [Rie94].

Die Variablen $x_1, ..., x_n$ bezeichnen in der Abbildung die Eingabesignale bzw. Attributwerte und $s_1, ..., s_m$ die Ausgabesignale bzw. Klassenwerte. Die vollständige Verknüpfung sowie die Vorwärtsausrichtung (*feed-forward neuronal network*) sind spezielle Eigenschaften des gewählten Verfahrens.

Durch das Verfahren der Rückwärtspropagierung (Backpropagation) werden die Gewichte angepasst. Dazu ist es nötig, jede Trainingsinstanz mittels des MLP klassifizieren zu lassen. Durch die Gegenüberstellung des Klassifikationsergebnisses s_i und des Klassenwertes der Trainingsinstanz a_i ist es nun möglich, den Fehler E zu berechnen:

$$E = \sum_{i=1}^{m} (s_i - a_i)^2$$

Die Änderung der Gewichtung bestimmt sich dabei durch die positiven Parameter ϵ die Lernrate (learning rate) sowie der Impuls α.

$$w_j(t+1) = w_j(t) + (-\varepsilon * \frac{\partial E}{\partial w_j}) + \alpha * \Delta w_j(t-1)$$

Die Höhe der Lernrate bewirkt dabei die Geschwindigkeit der Anpassung der Kantengewichte an die Änderungen in den Konzepten. Die Impulsvariable α hingegen bestimmt den Anteil der Variation im vorherigen Zeitschritt, welcher in die aktuelle Berechnung mit einfließt. Sie beeinflusst somit die Reduktion von Fehlern und eine übermäßige Anpassung. Grundsätzlich stellt das Training eines MLP folglich eine Variation der Kantengewichtung unter der Beachtung des quadratischen Fehlers dar.

NEARESTNEIGHBOUR

Um den Klassenwert einer Instanz zu bestimmen, durchsucht das NEARESTNEIGHBOUR-Verfahren die Trainingsinstanzen nach möglichst ähnlichen Instanzen. Die Ähnlichkeit wird durch eine Abstandsfunktion ermittelt. Der Klassenwert der Trainingsinstanz mit dem geringsten Abstand wird als Ergebnis genutzt (siehe Abbildung 5.9).

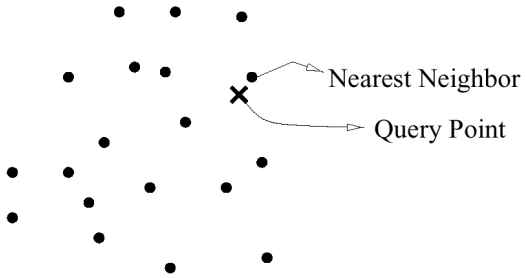


Abbildung 5.9: Beispiel: NEARESTNEIGHBOUR mit k=1 [BGRS99].

Eine häufig genutzte Variante ist das k-NEARESTNEIGHBOUR-Verfahren. Um weniger anfällig für Fehler in der Trainingsmenge zu sein, wird eine Menge k an Nachbarn bestimmt. Über diese Menge wird ein (je nach Umsetzung zusätzlich nach Abstand gewichtetes) Majority-Voting durchgeführt, um das Ergebnis zu bestimmen. Wird ein sehr großer Wert für k gewählt, so kann dass Modell weniger differenziert auf einzelne Aussagen eingehen. Wurde für einen Bereich wenig Feedback gegeben (siehe QoF in Abschnitt 3.4.5), so kann dies dazu führen, dass die Aussagen benachbarter Instanzen die Aussagen einzelner Instanzen überdecken.

In dieser Arbeit wird (sofern nichts weiter angegeben wurde) ein k-NEARESTNEIGHBOUR-Verfahren mit $k = 5$ verwendet.

Zur Suche der nächsten Nachbarn in einer Trainingsmenge mit n Instanzen und m Sensoren muss ein $n * m$ dimensionaler Raum durchsucht werden. Das Verfahren gehört daher zur Kategorie der Lazy-Learner und erzeugt bei der Auswertung gegebenenfalls erhöhten Aufwand.

Es gibt eine Reihe von Ansätzen dieses Problem zu behandeln [FW00, Das02b]. Ein Ansatz besteht beispielsweise darin, die Anzahl von Instanzen auf eine Menge von möglichst aussagekräftigen Instanzen zu reduziert. Eine solche Reduktion kann jedoch nur sinnvoll durchgeführt werden, wenn die Trainingsphase abgeschlossen ist. Dieser Weg ist für einen iterativen Ansatz also nicht geeignet.

Ein anderer Ansatz ist die Generalisierung der Instanzen. Sobald eine neue Instanz in die Trainingsmenge aufgenommen wird, wird ermittelt, ob diese Instanz mit einer bestehen Instanz kombiniert – generalisiert – werden kann. Eine solche generalisierte Instanz beschreibt nicht mehr einen Punkt im Raum, sondern ganze Wertebereiche und lässt sich gut in Form einer UND Verknüpften Regel darstellen. Dieser Ansatz wird im *NEARESTNEIGHBOUR-GENERALIZED*-Verfahren (*NN-GENERALIZED*) verfolgt.

Support Vector Machine (SVM)

Die Menge der Attribute der Trainingsinstanzen spannt einen hochdimensionalen Raum auf. Das Verfahren der Support Vector Machine (SVM) behandelt das Problem des Durchsuchens hochdimensionaler Räume, wie sie beispielsweise bei NEARESTNEIGHBOUR-Verfahren auftreten. Statt den hochdimensionalen Raum der Trainingsinstanzen zum Zeitpunkt der Auswertung zu durchsuchen, wird ein Modell erstellt, welches diesen Raum aufteilt. Ähnlich wie bei dem Aufbau eines Entscheidungsbaumes hat die Aufteilung des Raumes zum Ziel, Instanzen, welche zu einer bestimmten Klasse gehören, zu gruppieren, beziehungsweise durch die Aufteilung von anderen Instanzen zu trennen.

Der Raum wird durch Hyperebenen (Unterräume deren Dimension um eine geringer ist, als der Domänenraum) aufgeteilt. Einzelne Instanzen (die in diesem Zusammenhang als Support-Vektoren bezeichnet werden), welche einen minimalen Abstand zur Hyperebene aufweisen, dienen hierbei zur Beschreibung der Hyperebenen. Diese Support-Vektoren sind in Abbildung 5.10 durch Kreuze markiert.

Hochdimensionale Räume sind jedoch nicht immer durch Hyperebenen trennbar. Daher ist es entweder nötig die lineare Hyperebene durch eine nichtlineare zu ersetzen (wobei deren Berechnung die als akzeptabel annehmbare

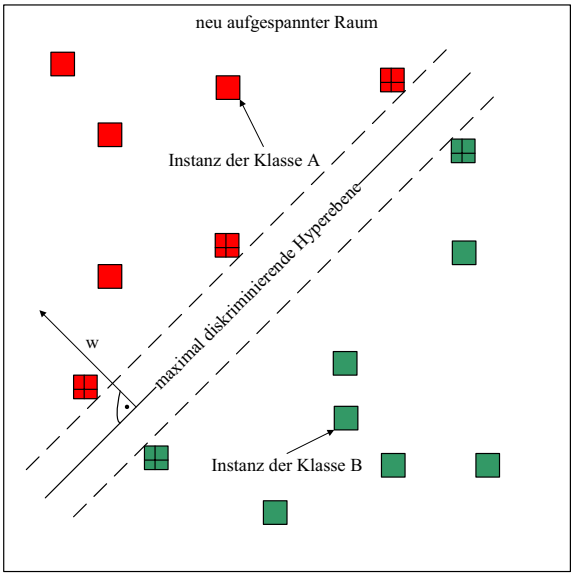


Abbildung 5.10: Trennung der Instanzen durch Hyperebene im Domänenraum [Mar03].

Zeit für Lernsysteme überschreiten würde) oder den Domänenraum so zu verändern, dass eine lineare Trennung möglich wird. Das SVM-Verfahren wählt den zweiten Weg und nutzt sogenannte *Kernel-Funktionen* [WF01] als Verfahren aus, um den Raum zu transformieren. In diesem neu aufgespannten Raum der Instanzen ist es nun möglich, diese mittels einer linearen Funktion bzw. einer Hyperebene zu trennen.

Das Training der SVM ist also nur dann von einem hohen Aufwand geprägt, wenn sich die Support-Vektoren, welche durch die Trainingsinstanzen gestellt werden, ändern und gegebenenfalls die Hyperebenen neu berechnet werden müssen. Anderseits besteht, durch die Integration der neuen Instanz, die Gefahr, dass die bisher vorhandenen Dimensionen des neu aufgespannten Raumes nicht mehr ausreichen, um eine lineare Trennung vornehmen zu können. Das Hinzufügen einer weiteren Dimension für diesen Raum löst zwar dieses Problem, jedoch bedingt es die erneute Ermittlung einer weiteren, geeigneten Kernel-Funktion.

In dieser Arbeit wird als Kernel-Funktion das Verfahren der Sequential Minimal Optimization (SMO) genutzt. Dieses Verfahren unterteilt das quadratische Optimierungsproblem (QP) der SVM in kleinere und analytisch zu lösende Problemstellungen [OFG97, Pla99, KSBM99].

5.4.3 Referenz: Basisszenario

In Abschnitt 5.4.1 wurden die Parameter für ein Basisszenario festgelegt:

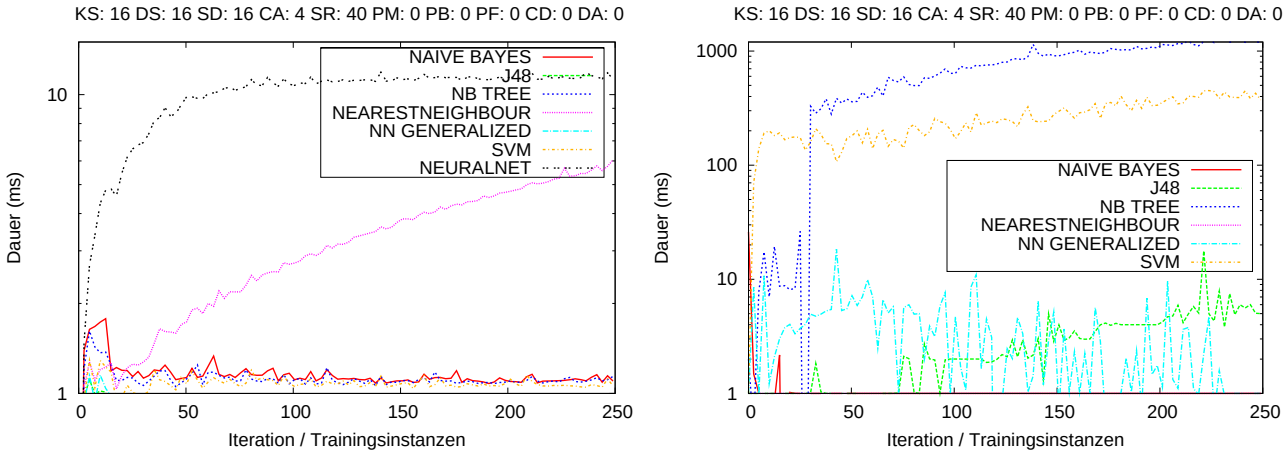
- Diskrete Sensoren: 16.
- Anzahl diskreter Werte: 16.
- Numerische Sensoren: 16.
- Anzahl von Kontextklassen: 4.
- Anzahl von Sensoren mit Relation zum Konzept: 40%.
- Keine Eigenschaften wie fehlerhafte oder fehlende Sensorwerte, fehlerhaftes Feedback oder Konzeptänderungen.

Alle Testläufe wurden auf einem Intel Core 2 Quad-Core mit 2,67 GHz und 3GB RAM durchgeführt, wobei von dem Thread jeweils nur ein Prozessorkern belegt wurde. In den folgenden Darstellungen werden die Mittelwerte über 10 Testläufe hinweg dargestellt[6].

[6] Die Konfidenzintervalle wurden aus Gründen der Übersichtlichkeit und Lesbarkeit weggelassen. Ein exemplarisches Szenario mit Konfidenzintervallen findet sich im Anhang im Abschnitt A.6.

Als Referenz für die Eigenschaften der Verfahren wird jeweils die in diesem Abschnitt dargestellte Auswertung des Basisszenarios verwendet. Von diesem ausgehend werden jeweils einzelne Parameter variiert und deren Auswirkungen analysiert.

Die Laufzeiteigenschaften des Basisszenarios wie die Modellauswertung und die Modellgenerierung sind in Abbildung 5.11 aufgezeigt.



(a) Modellauswertung (100 Durchläufe)

(b) Modellgenerierung

Abbildung 5.11: Aufwandsbetrachtung des Basisszenarios

Für die Modellauswertung wurde der Zeitbedarf für 100 Auswertungen gemessen. Eine einzelne Auswertung benötigt somit in der Regel weniger als 0.1 Millisekunden. Bei der Betrachtung des Aufwandes der Modellauswertung in Abbildung 5.11 a) wird der Unterschied zwischen Eager-, und Batch-Learner zu der Kategorie der Lazy-Learner deutlich. NEARESTNEIGHBOUR, welches zu den Lazy-Learnern zählt, erfordert bei der Auswertung einen Aufwand, welcher abhängig von der Anzahl der Trainingsinstanzen ist. Die anderen Verfahren zählen zu den Eager- und Batch-Learnern und generieren ein Modell, welches bei der Auswertung meist deutlich weniger Aufwand zur Auswertung benötigt und dessen Aufwand zur Auswertung unabhängig von der Anzahl der Trainingsinstanzen ist. NEURAL NET erzeugt bei der Auswertung einen erhöhten Aufwand, welcher durch die Anzahl der Durchläufe erzeugt wird, die benötigt werden, um ein stabiles Ergebnis an den Ausgangknoten zu erhalten.

In Abbildung 5.11 b) wird der Aufwand bei der Modellgenerierung analysiert. Bei dem Vergleich über die Anzahl der Trainingsinstanzen wird der Unterschied zwischen Batch basierten und inkrementellen Ansätzen deutlich. Die Lernverfahren NB TREE, J48 und SVM, welche der Kategorie Batch-Learner angehören, benötigen je Iteration mehr Aufwand, da sie in jedem Schritt ein neues Modell, aus allen bisher gesammelten Trainingsinstanzen, generieren. Trotzdem unterscheiden sich auch die Verfahren dieser Kategorie untereinander. So benötigt das Verfahren J48 deutlich weniger Aufwand als NB TREE und SVM. Das Verfahren NB TREE beginnt anfangs mit einem einfachen NAIVE BAYES basiertem Modell. Erst bei wachsender Anzahl von Trainingsinstanzen wird ein Entscheidungsbaum analog zum J48-Verfahren aufgebaut, welcher als Blätter einzelne NAIVE BAYES Modelle enthält. Dieses Verhalten ist in der Laufzeit zu erkennen, welche sich nach einer bestimmten Anzahl von Instanzen deutlich erhöht.

Alle anderen Verfahren gehören der Kategorie Incremental-Learner an und führen lediglich ein Update des bestehenden Modells durch. Bei NN GENERALIZED reduziert sich der Aufwand sogar mit der Anzahl der Iterationen. Durch die Eigenschaft des genutzten WEKA-Rahmenwerks[7] für Data-Mining, auf Strukturen mit festen, zwei dimensionalen Tabellen und festen Vorgaben der Wertebereiche der diskreten Attribute zu arbeiten, muss in Fällen, in denen neue Sensoren hinzukommen oder Sensoren bisher ungesehene diskrete Werte annehmen, ebenfalls ein neues Modell generiert werden. Dieser Fall tritt gerade in der Anfangsphase häufiger auf und wird bei NN GENERALIZED sichtbar. Der Grund für die Reduzierung des Aufwandes bei der Modellgenerierung bei NN GENERALIZED wird bei genauerer Betrachtung des Prinzips von dem Verfahren deutlich. Sobald eine Regelmenge mit ausreichender Aussagekraft erstellt wurde und neue Instanzen den bereits bestehenden Regeln entsprechen, wird keine weitere Anpassung notwendig.

[7] `http://www.cs.waikato.ac.nz/ml/weka/`

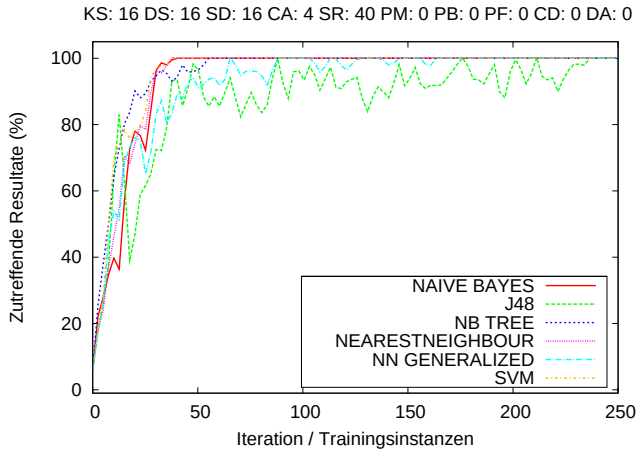


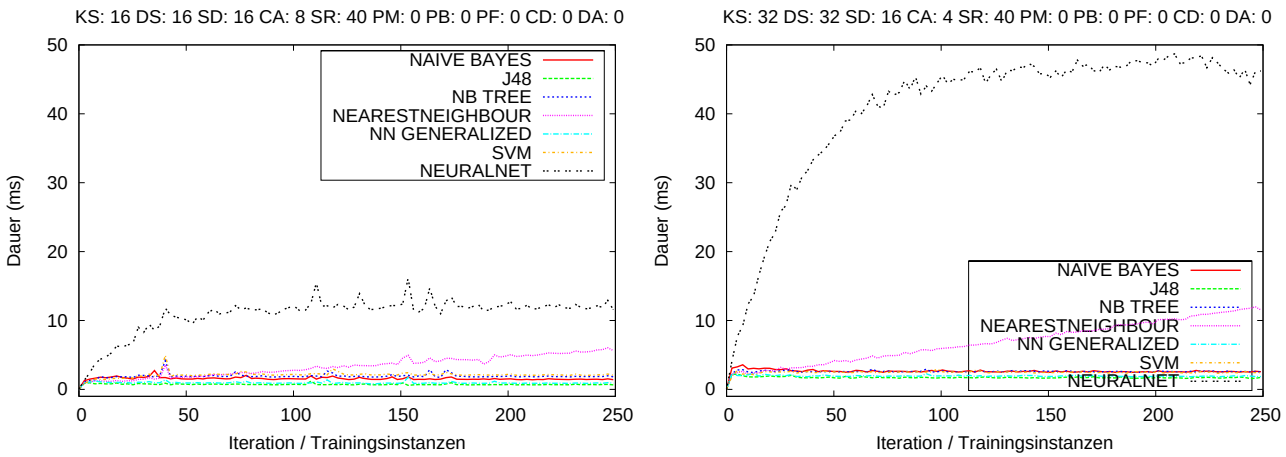
Abbildung 5.12: Genauigkeit des Modells innerhalb des Basisszenarios

Die Qualität des Basisszenarios ist in Abbildung 5.12 dargestellt. Unter den Eigenschaften des Basisszenarios zeigen alle Verfahren eine relativ gute Lernkurve, die je nach Verfahren nach 30 bis 50 Schritten, eine Klassifikationsgenauigkeit von 90% bis 100% erreicht. In Anbetracht des relativ großen Domänenraumes und der Anzahl an Sensoren, welche keine Relation zum Konzept aufweisen, ist dies ein ordentliches Resultat.

Die in diesem Abschnitt gesammelten Messwerte des Basisszenarios dienen nun als Referenz für die nachfolgenden Analysen, bei denen jeweils ausgewählte Parameter des Szenarios variiert werden.

5.4.4 Aufwandsbetrachtung - Evaluation der Modelle

Für das angestrebte Anwendungsszenario liegt der Fokus auf der Evaluation eines einzelnen Kontextes. Bei diesem Testaufbau wird der Zeitbedarf von 100 Auswertungen in Abhängigkeit zu der Größe des Domänenraumes gemessen.



(a) Doppelte Anzahl Klassen

(b) Doppelte Anzahl Sensoren

Abbildung 5.13: Abhängigkeiten des Aufwandes bei der Auswertung eines Modells

Die Tabelle 5.3 gibt einen zusammenfassenden Überblick über die erhaltenen Messwerte. Die dritte Spalte bietet eine grobe Aufwandsabschätzung, indem die Abhängigkeiten zu Größen in der Informationsgrundlage welche für den Zeitbedarf der Auswertung relevant sind, aufgezeigt werden:

Verfahren	μ_t	$max(t)$	Aufwand abhängig von
NAIVE BAYES	1,16 ms	2,19 ms	$\|Klassen\| * \|Sensoren\|$
J48	0,70 ms	1,13 ms	$log(\|RelevanteSensoren\|)$
NB TREE	1,12 ms	1,62 ms	$log(\|RelevanteSensoren\|) + \|Klassen\| * \|Sensoren\|$
NEARESTNEIGHBOUR	3,26 ms	6,05 ms	$\|Instanzen\| * \|Sensoren\|$
NN GENERALIZED	0,82 ms	1,36 ms	$\|Klassen\| * \|RelevanteSensoren\|$
SVM	1,08 ms	1,29 ms	$\|Klassen\| * \|Sensoren\|$
NEURALNET	10,11 ms	12,13 ms	$log(\|Instanzen\|) * \|Sensoren\|$

μ_t durchschnittlich gemessener Zeitbedarf.

Tabelle 5.3: Zeitbedarf t für 100 Evaluationen des Basisszenarios

- Die Auswertung eines NAIVE BAYES Modells berechnet die Wahrscheinlichkeit für alle *Klassen*, indem der Einfluss jedes *Sensors* einzeln ausgewertet wird.
- J48 baut einen Baum mit *relevanten Sensoren* in den Knoten auf. Bei der Auswertung wird jeweils nur ein Pfad von der Wurzel bis zum Blatt durchlaufen.
- NB TREE kombiniert den Ansatz von J48 mit dem von NAIVE BAYES. Ein NB TREE besitzt jedoch nur noch reduzierte NAIVE BAYES Modelle in den Blättern.
- NEARESTNEIGHBOUR bestimmt die Distanz zwischen der Klassifikationsinstanz und allen *Trainingsinstanzen*. Eine Distanz wird dabei über alle *Sensoren* hinweg berechnet.
- NN GENERALIZED baut für jede *Klasse* eine Menge von Regeln auf. Diese Regeln beschreiben die Zusammenhänge zwischen den Klassen und den *relevanten Sensoren*.
- SVM beschreibt eine Reihe von Hyperebenen zur Trennung zwischen den verschiedenen *Klassen*. Bei der Auswertung wird die Position einer Klassifikationsinstanz in Relation zu jeder Hyperebene ausgewertet, indem die *Sensoren* der Klassifikationsinstanz mit denen der Support Vektoren verrechnet werden.
- NEURALNET verfügt über einen Eingangsknoten je *Sensor*, sowie über einen Ausgangsknoten für jede *Klasse*. Bei der Auswertung werden die Werte an die Eingangsknoten angelegt und durch das Netzwerk bis zu den Ausgangsknoten propagiert. Der Aufwand ist stark Abhängig vom Aufbau des Netzes oder der Methode welche dazu herangezogen wird. Häufig wird beim Aufbau des *Hidden Layers* zwischen jeder Kombination zweier Eingangssensoren ein weiterer Knoten im Hidden Layer erzeugt. Bei dem NEURALNET wurde die Standardeinstellung verwendet, welches einen automatische Aufbau des Neuronalen Netzwerkes vorsieht. Dabei werden beim Training gegebenenfalls weitere Knoten (Hidden Layer) hinzugefügt. Bei mehr Trainingsschritten wurde daher das Netz entsprechend größer, was zu steigender Verzögerung bei der Auswertung führt.

Insgesamt benötigen alle Verfahren in den betrachteten Szenarien einen Rechenaufwand, welcher im Rahmen des Kommunikationsszenarios annehmbar wäre. Betrachtet man die Skalierbarkeit des Systems, so muss jedoch auch die Abhängigkeit zu den Problemgrößen hinzugezogen werden. Viele der Modelle sind abhängig von der Anzahl der Klassen und Sensoren. Die in dem Kommunikationsszenario zu erwartende Anzahl von Klassen wird meist relativ klein bleiben. Die Anzahl von Sensoren ist abhängig von der Menge an Sensoren, welche verfügbar sind und von der Informationsgewinnung bei der Suche ausgewählt werden. Die Suche hat die Aufgabe, die Menge der Sensoren auf *relevante* Sensoren zu reduzieren (siehe Abschnitt 4.6), so dass auch hier eine Beschränkung der Anzahl von auszuwertenden Sensoren zu erwarten ist. Die Anzahl von Instanzen, welche zur Modellgenerierung herangezogen werden, ist abhängig von der Umsetzung des Systems und ob Mechanismen zur Limitierung der Menge von Instanzen herangezogen werden (siehe Abschnitt 5.5).

Da der Zeitbedarf zur Auswertung ein wesentlicher Aspekt bei der Leistung des Gesamtsystems ist, werden diejenigen Verfahren bevorzugt, welche in dem Zusammenhang einen möglichst geringen Aufwand erzeugen.

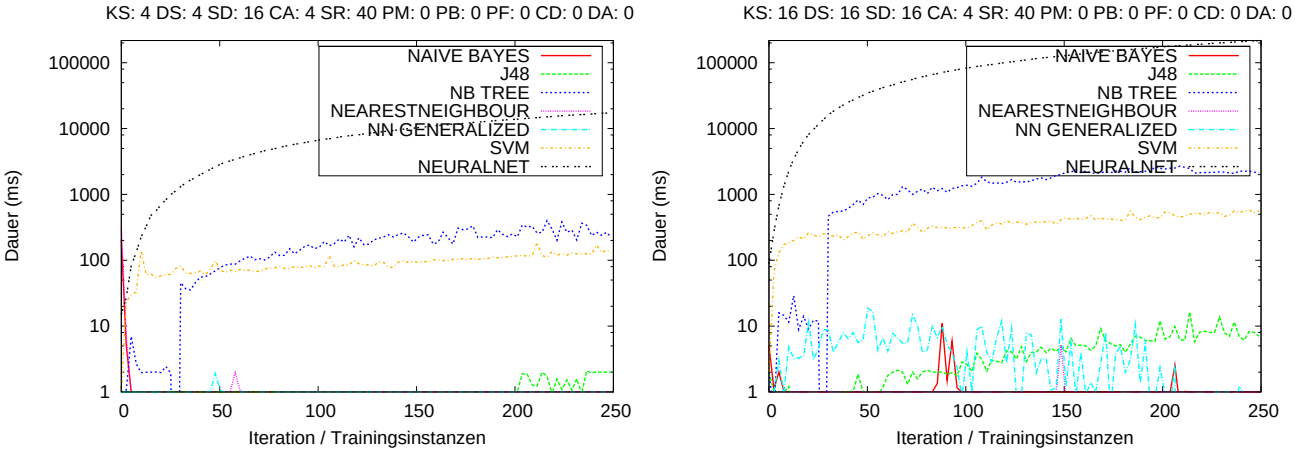
5.4.5 Aufwandsbetrachtung - Generierung der Modelle

Neben der Anzahl von Instanzen und Sensoren und der Größe der Klassendomäne haben auch andere Faktoren Einfluss auf die Modellgenerierung. Hierzu zählen die Verteilung der Sensoren auf diskrete und kontinuierliche

Sensortypen, die Domäne der diskreten Werte und die Anzahl von Sensoren mit Relation zum Konzept. Diese und weitere Analysen finden sich im Anhang in Abschnitt A.6.

Anzahl von Sensoren

Bei der Generierung eines Modells ist neben der Anzahl der zu verarbeitenden Trainingsinstanzen die Anzahl von Sensoren pro Trainingsinstanz ein wesentlicher Faktor für die Aufwandsbetrachtung. In Abbildung 5.14 wird Zeitbedarf für die Modellgenerierung in Abhängigkeit von der Anzahl von Sensoren dargestellt.



(a) 8 Sensoren

(b) 32 Sensoren

Abbildung 5.14: Vergleich verschiedener Mengen von Sensoren

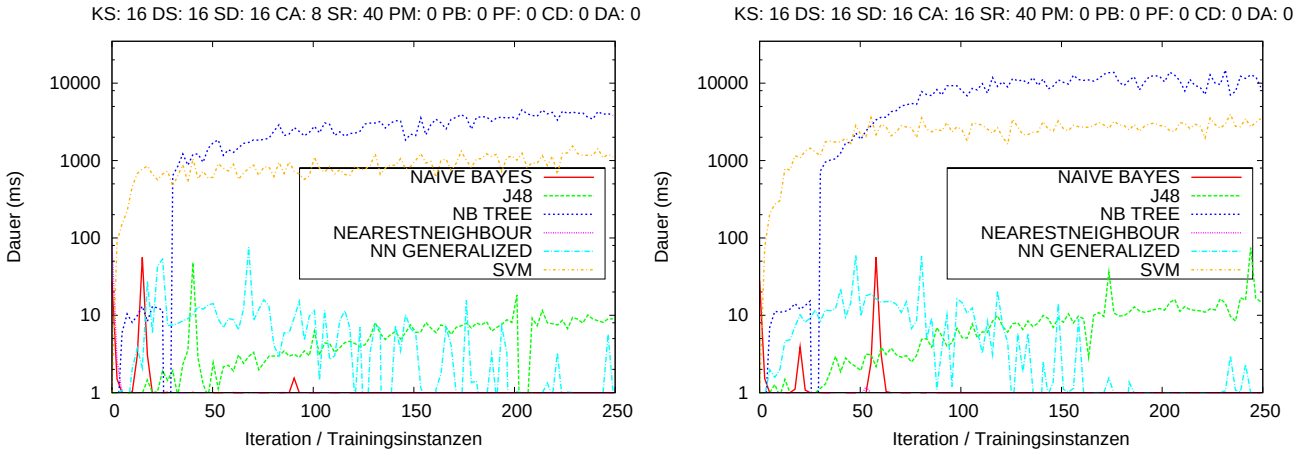
Die Anzahl von Sensoren pro Instanz betrifft jedes Verfahren. Da jedoch NAIVE BAYES und NEARESTNEIGHBOUR nur wenige Operationen pro Iterationsschritt erfordern, erzeugt die Erhöhung der Anzahl von Sensoren keine maßgebliche Veränderung bei diesen Verfahren. Die anderen Verfahren zeigen bei einer Verdoppelung der Anzahl von Sensoren meist eine Verdoppelung bis hin zu einer Vervierfachung des Aufwandes.

Das Verfahren NEURALNET erzeugt in seiner Standard-Konfiguration einen sehr hohen Aufwand. Ein Neuronales Netzwerk basiert auf einer inkrementellen Funktionsweise. Die Implementierung *Multilayer Perzeptron* in WEKA ist jedoch nur als Batch-Verfahren nutzbar. Durch die Wahl einer geeigneten Implementierung wäre ein Neuronales Netzwerk prinzipiell auch für die Anwendung im Rahmen dieser Arbeit geeignet. Allerdings erzeugt diese Variante mit bis zu mehreren Minuten pro Iteration einen zu hohen Aufwand. Es erfüllt daher nicht die Anforderungen die für diese Arbeit vorausgesetzt werden und wurde daher für die weiteren Analysen ausgelassen.

Größe der Klassendomäne

Die Anzahl der Kontextklassen variiert mit dem Anwendungsszenario. Während eine binäre Entscheidung zwei Kontextklassen beinhaltet, würde die Bestimmung der Bezeichnung des Raums in dem sich Nutzers aufhält eine Vielzahl von Klassen beinhalten. Wie sich der Aufwand bei der Modellgenerierung abhängig von der Anzahl Kontextklassen verhält, wird in Abbildung 5.15 dargestellt.

Bei der Verdoppelung der Klassendomänen verdoppelt sich auch der Aufwand von J48, während sich der von SVM und NB TREE annähernd vervierfacht. Bei anderen Verfahren wie NEARESTNEIGHBOUR, NAIVE BAYES oder NN GENERALIZED dagegen zeigen sich keine bemerkenswerten Auswirkungen.



(a) Klassendomäne: 8

(b) Klassendomäne: 16

Abbildung 5.15: Vergleich verschiedener Größen der Klassendomäne

5.4.6 Zusätzliche Eigenschaften der Modelle

Bisher wurden die grundlegenden Eigenschaften, wie die Performanz der Lernverfahren betrachtet. Im den nachfolgenden Abschnitten wird ebenso auf die Genauigkeit der Lernverfahren eingegangen. Jedes Lernverfahren verfügt jedoch über eine Reihe weiterer spezifischer Eigenschaften, welche für weiterführende Arbeiten von Interesse sind:

Aktuell relevante Sensoren

Bei manchen Modellen kann nach einer Entscheidung die Menge an Sensoren bestimmt werden, welche letztendlich zu der Entscheidung geführt haben. Betrachtet man beispielsweise einen Entscheidungsbaum (siehe Abschnitt 5.4.2) während er bei einer Entscheidung durchlaufen wird, so kann festgestellt werden, welche Sensoren ausgewertet wurden. Dieses Wissen kann genutzt werden, um Aufwand zu sparen, indem weitere Entscheidungen erst bei einer Änderung des Zustandes einer dieser Sensoren durchgeführt werden. Hierzu könnten geeignete Mechanismen zur Notifikation über Werteänderungen an bestimmten Sensoren genutzt werden.

Updatefähigkeit

In Abschnitt 5.3.2 wurden inkrementelle Lernverfahren beschrieben. Diese Eigenschaft führt neben der Verringerung des Aufwandes zur Modellgenerierung auch zu Einsparungen an Speicherplatz, da kein Vorhalten aller Instanzen erforderlich ist.

Gewichtete Instanzen

Manche Verfahren erlauben es, Trainingsinstanzen zu gewichten. Dies kann gerade im Rahmen von Fensterverfahren vorteilhaft sein (siehe Abschnitt 5.5). Ebenso könnte diese Eigenschaft als weitere Funktion innerhalb von Feedback genutzt werden. Beispielsweise könnte somit angegeben werden wie *sicher* das Feedback ist, also wie wahrscheinlich das Feedback mit dem Konzept übereinstimmt.

Wahrscheinlichkeit für eine korrekte Entscheidung

In einigen Verfahren ist es implizit möglich neben dem Resultat auch die Wahrscheinlichkeit für eine korrekte Entscheidung zu bestimmen (engl. *confidence*). Beispielsweise kann bei dem Aufbau eines Entscheidungsbaums an jedem Blatt vermerkt werden, wie viele der Trainingsinstanzen, welche zum Aufbau dieses Blattes beigetragen haben, auch das gleiche Resultat aufweisen, wie das Blatt selbst.

Ebenso können anhand ein zusätzliche Analysen (z.B. über die *Ten-fold Crossvalidation* siehe Abschnitt 5.4.1) Aussagen über die Qualität des Modells gemacht und die Genauigkeit je Kontextklasse bestimmt werden.

Eine Aussage über die Wahrscheinlichkeit einer korrekten Entscheidung kann letztendlich ebenso dazu genutzt werden, eine geeignete Wahl der auszuführenden Aktion durchzuführen. Im Falle einer unsicheren Entscheidung,

können geeignete Maßnahmen ergriffen werden. Beispielsweise kann dann eine Rückfrage an die betreffenden Teilnehmer oder ein Rückfall auf statische Regeln durchgeführt werden.

Multivariate Kontextklassen

Je nach Anwendungsszenario kann ein Kontextobjekt möglicherweise gleichzeitig in mehreren Kontextklassen sein. Manche Verfahren ermöglichen es, eine Menge von Resultaten als Ergebnis zu liefern. Wenn beispielsweise wie bei NAIVE BAYES (siehe Abschnitt 5.4.2) für jede Kontextklasse die Wahrscheinlichkeit des Zutreffens berechnet wird, so können alle Klassen als zutreffend bestimmt werden, welche über einem gewissen Grenzwert liegen. Für diese Arbeit wurde jedoch die Annahme getroffen, dass innerhalb einer Kontextdimension jeweils nur eine Kontextklasse zu einem Zeitpunkt gültig ist (siehe Abschnitt 2.4.3).

5.4.7 Qualitätsbetrachtung

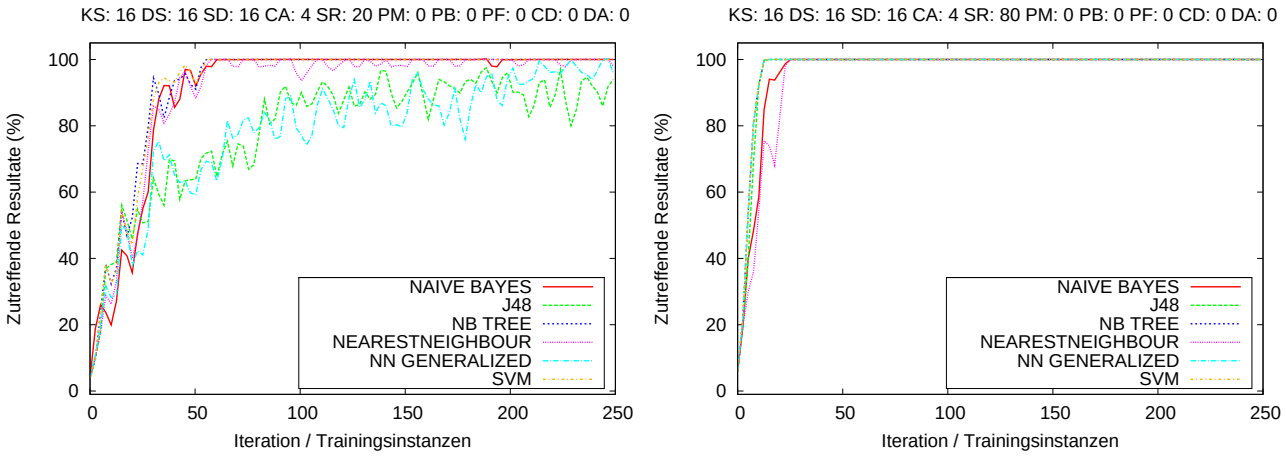
In Abschnitt 5.3.1 wurden die zu erwartenden Eigenschaften der Informationsgrundlage beschrieben. In diesem Abschnitt werden die Effekte, welche durch die Eigenschaften der Informationsgrundlage verursacht werden, analysiert.

Die Ausprägung des Effektes wird jeweils mit 0%, 25% und 50% betroffenen Elementen im Vergleich betrachtet. Wobei die Darstellung mit 0% jeweils den Optimalfall darstellt, welches durch das Referenzszenario in Abschnitt 5.4.3 dargestellt wird.

Die in den folgenden Abschnitten aufgeführten Darstellungen zeigen die Mittelwerte, welche über 10 Testläufe hinweg gemessen wurden[8].

Menge von Sensoren mit Relation zum Konzept

Wie sich die Qualität der Entscheidung abhängig von der Menge an relevanten beziehungsweise ausschlaggebenden Sensoren verhält, ist in Abbildung 5.16 dargestellt.



(a) Relevanz: 20%

(b) Relevanz: 80%

Abbildung 5.16: Vergleich verschiedener Anteile von Sensoren mit Relation zum Konzept

Eine sichere Entscheidung für eine Kontextklasse ist dann möglich, wenn sich die Domänen der Konzeptklassen nicht vollständig überdecken und mindestens ein Sensor eine Differenzierung im Bezug zu den anderen Kontextklassen erlaubt. Die Herausforderung besteht jedoch darin, aus einer Menge von Sensoren mindestens diesen einen, relevanten Sensor zu bestimmen. Je größer diese Menge ist, desto länger benötigen die Verfahren um relevante

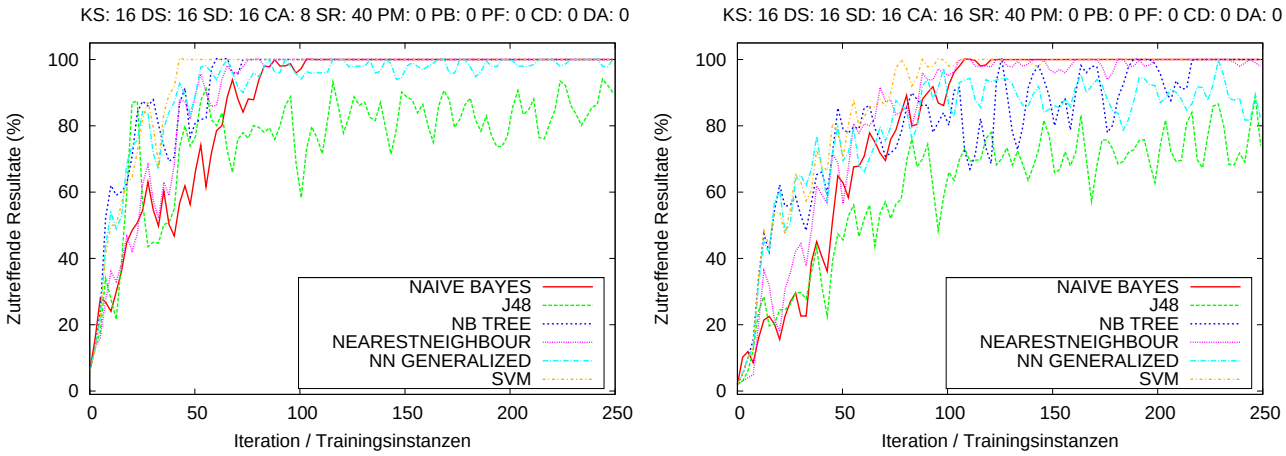
[8] Aus Gründen der Übersichtlichkeit und der Lesbarkeit wurde auf Konfidenzintervalle verzichtet. Ein exemplarisches Szenario mit Konfidenzintervallen findet sich im Anhang im Abschnitt A.6.

Sensoren von irrelevanten zu unterscheiden. Bei 32 Sensoren und einem Anteil von 20% an relevanten Sensoren sind 6-7 Sensoren relevant für die Entscheidung. Alle anderen Sensoren liefern gleich verteilte Zufallswerte. Um die Entscheidung von dieser Menge von den 20% relevanter Sensoren abhängig zu machen und eine Qualität der Entscheidung von 90% zu erreichen, benötigen Verfahren wie NAIVE BAYES, NEARESTNEIGBOUR, SVM und NB TREE ca. 50 Trainingsinstanzen. J48 und NN GENERALIZED benötigen dagegen mit ca. 150 Trainingsinstanzen deutlich länger.

Erhöht man die Rate relevanter Sensoren, steigt auch die Lernkurve deutlich an. Bei 40% relevanter Sensoren, wie sie im Basisszenario dargestellt wurden, erreichen alle Verfahren die Genauigkeit von 90% mit 25 bis 50 Trainingsinstanzen. 80% relevante Sensoren führen zu annähernd *optimalen* Lernkurven (die Anzahl notwendiger Trainingsinstanzen nähert sich der Anzahl der Kontextklassen an).

Größe der Klassendomäne, Anzahl von Sensoren

Das Verhältnis der Klassendomäne zu der Anzahl von Sensoren ist bei der Betrachtung der erreichbaren Qualität ebenfalls von Interesse. Steigt, wie in Abbildung 5.17 dargestellt, die Anzahl der Kontextklassen bei gleichbleibender Anzahl von Sensoren, sinkt die Qualität der Entscheidung und die Länge der Lernphase.



(a) Klassendomäne: 8

(b) Klassendomäne: 16

Abbildung 5.17: Vergleich verschiedener Größen der Klassendomäne

Bei einer steigenden Anzahl von Kontextklassen steigt die Wahrscheinlichkeit für Überschneidungen im Domänenraum (komplette Überschneidungen wurden vermieden) und die Klassen können schwerer voneinander differenziert werden.

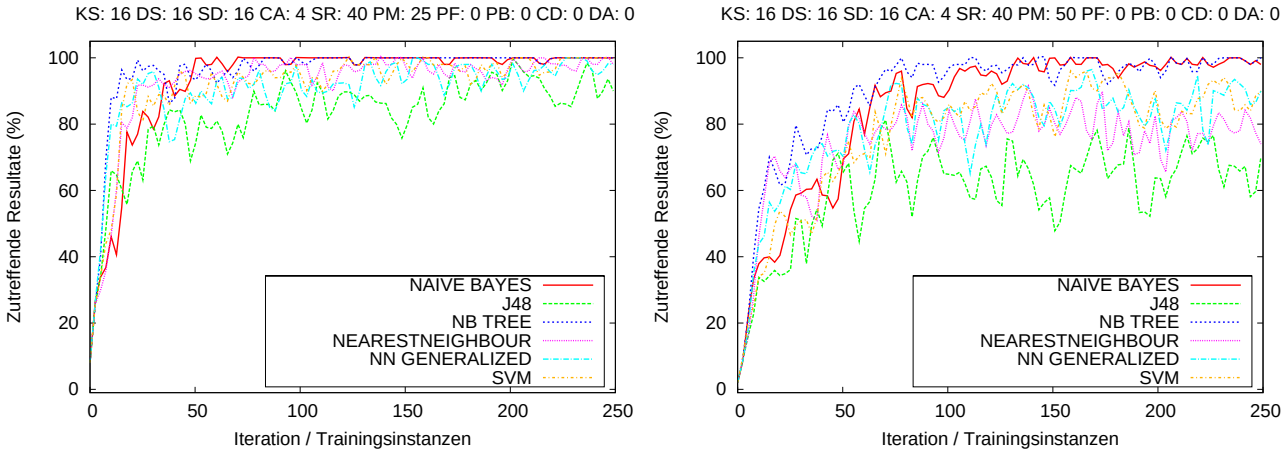
Fehlende Sensorwerte

In der Abbildung 5.18 sind die Auswirkungen fehlender Sensorwerte dargestellt.

In dem Szenario sind trotz fehlender Sensorwerte genug aussagekräftige Sensorwerte für jedes Konzept verfügbar. Dies wird deutlich durch die Genauigkeit, welche durch die Verfahren NAIVE BAYES, NB TREE oder SVM erreicht werden. Trotzdem zeigt das Entscheidungsbaumverfahren J48 sich stark von fehlenden Sensorwerten betroffen. Diese Tatsache beruht auf der Eigenschaft des Verfahrens in jedem Knoten explizit einen Sensorwert zu betrachten. Ist dieser Wert nicht verfügbar, wird der Teilbaum weiter verfolgt, zu dem mehr Trainingsinstanzen zugeordnet werden können.

Die Qualität von NEARESTBEIGHBOUR, SVM, und NN GENERALIZED ist ebenso von dem Effekt fehlender Sensorwerte beeinträchtigt, zeigt jedoch geringe Einbrüche wie J48.

Am wenigsten von fehlenden Sensorwerten betroffen sind diejenigen Verfahren, welche auf NAIVE BAYES aufsetzen. Bei NAIVE BAYES wird eine Wahrscheinlichkeit über alle Sensorwerte berechnet. Fehlt einer dieser Werte,



(a) 25% fehlende Sensoren

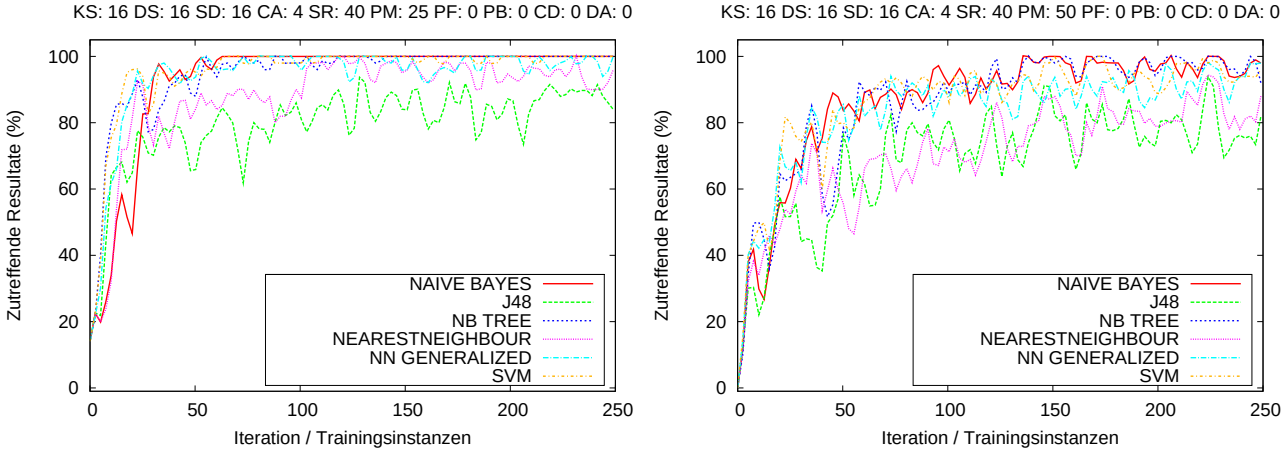
(b) 50% fehlende Sensoren

Abbildung 5.18: Vergleich verschiedener Raten von fehlenden Sensorwerten

so wird die Berechnung ohne den Einfluss dieses Wertes fortgeführt. Somit kann eine Aussage eines einzelnen Wertes, durch Aussagen anderer (statistisch abhängiger) Sensorwerte ergänzt werden.

Fehlende Sensorwerte - Optimierung durch Vorverarbeitung

Durch Schritte der Vorverarbeitung (siehe auch Abschnitt 5.2.5) können fehlende Aussagen ergänzt werden. Mit Hilfe von semantischen Informationen können hierzu Sensoren aggregiert werden. Im Rahmen dieses Vergleiches, bei dem keine semantischen Informationen vorliegen, wurde hierzu ein vereinfachtes Verfahren angewendet, welches lediglich einzelne fehlende Sensorwerte, durch den bisher beobachteten Durchschnittswert des entsprechenden Sensors ersetzt. In der Abbildung 5.19 wird der Optimierungsgrad der Genauigkeit durch diese Form der Vorverarbeitung deutlich.



(a) 25% fehlende Sensoren

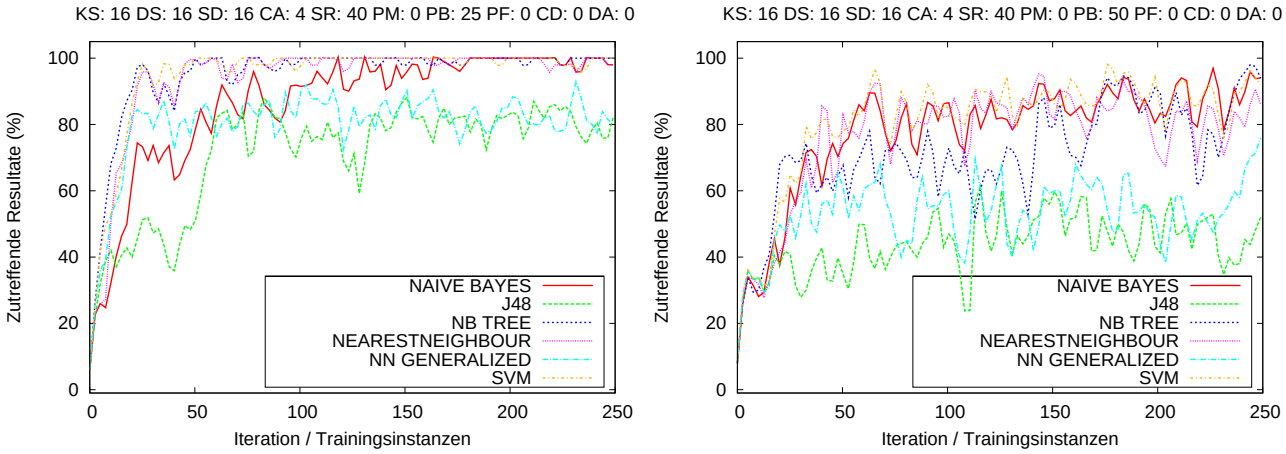
(b) 50% fehlende Sensoren

Abbildung 5.19: Vergleich verschiedener Raten von fehlenden Sensorwerten in Kombination mit Vorverarbeitung zur Ergänzung fehlender Werte

Die Analyse wurde mit denselben Parametern wie in Abbildung 5.18 durchgeführt, jedoch (bedingt durch den Aufbau) mit anderen, zufällig erstellten Instanzen. Trotzdem sind deutliche Unterschiede gerade bei dem Entscheidungsbaumverfahren J48 zu erkennen. Die Verfahren SVM, NN GENERALIZED, liefern durch dieses Vorgehen ebenfalls leicht verbesserte Ergebnisse, während NAERESTNEIGHBOUR und auf NAIVE BAYES basierende Verfahren davon relativ unbeeinflusst arbeiten.

Fehlerhafte Sensorwerte

In Abbildung 5.20 sind die Ergebnisse der Analyse, welche Auswirkungen fehlerhafte Sensorwerte haben, dargestellt. Im Fehlerfall liefern einzelne Sensorwerte zufällig, abweichende Werte (in diesem konstruierten Szenario gleich verteilt über ihren Wertebereich).



(a) 25% fehlerhafte Sensoren

(b) 50% fehlerhafte Sensoren

Abbildung 5.20: Vergleich verschiedener Raten von fehlerhaften Sensorwerten

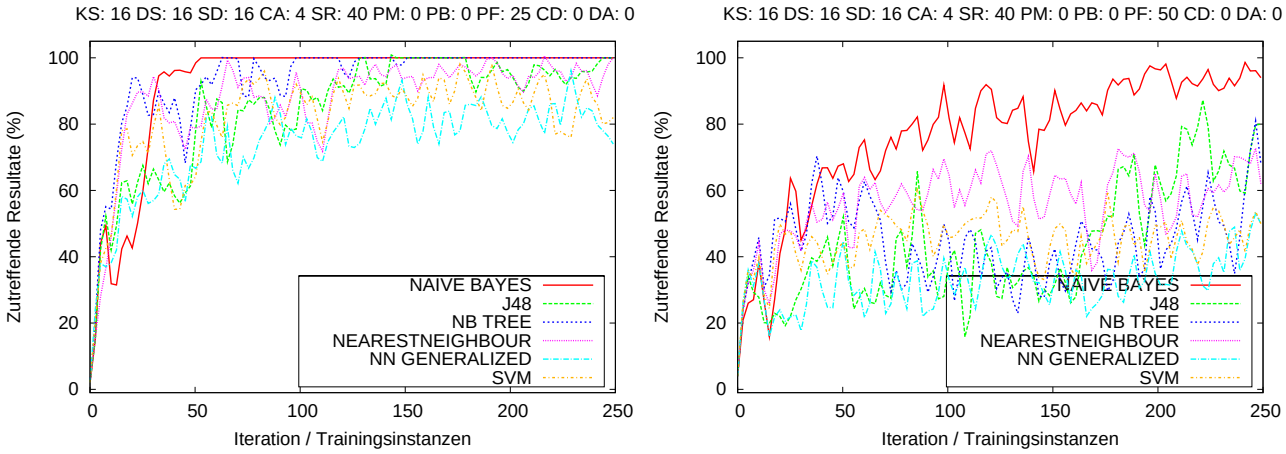
Fehlerhafte Sensorwerte, wie sie in Abschnitt 5.3.1 beschrieben werden, haben im direkten Vergleich mit fehlenden Sensorwerten einen ähnlichen, jedoch stärker ausgeprägten Effekt. Die Auswirkungen sind je nach Verfahren unterschiedlich. Das Entscheidungsbaumverfahren J48 hat, wie beim Effekt fehlender Sensorwerte, den stärksten Einbruch in der Genauigkeit. Während das Verfahren NN GENERALIZED mehr Schwierigkeiten hat mit fehlerhaften Sensorwerten umzugehen, als mit fehlenden Sensorwerten, ist es beim NEARESTNEIGHBOUR-Verfahren gerade umgekehrt. Ebenso wie bei fehlenden Sensorwerten schneiden NAIVE BAYES und SVM am besten ab, während sich die Genauigkeit von NB TREE nur noch im Mittelfeld bewegt. Insgesamt ist festzuhalten, dass es möglich ist trotz einer sehr hohen Rate von 50% fehlerhaften (also zufällig gewählten) Aussagen eine Genauigkeit von rund 80% zu erhalten.

Sofern Aussagen über die Bedeutung der Sensorwerte anhand ihrer semantischen Beschreibung möglich sind, können (vergleichbar zum Ersetzen fehlender Sensorwerte) durch geeignete Vorverarbeitungsschritte Verbesserungen in der Qualität erreicht werden.

Fehlerhaftes Feedback

Fehlerhaftes Feedback ist ein weiterer Effekt, welcher gerade in Szenarien vorkommen kann, in denen das Feedback von Nutzern generiert wird.

In geringem Umfang, wie bei der Darstellung mit 25% fehlerhaftem Feedback ersichtlich, kann dieses gut ausgeglichen werden und zeigt im Durchschnitt einen ähnlichen Effekt wie fehlerhafte und fehlende Sensorwerte. Bei der erhöhten Rate, bei der 50% aller Antworten zufällig gewählt wurden, sind die Auswirkungen im Durchschnitt deutlich stärker ausgeprägt als bei den anderen Effekten. Fast alle Verfahren weisen eine Qualität auf, die zum Teil



(a) 25% fehlerhaftes Feedback

(b) 50% fehlerhaftes Feedback

Abbildung 5.21: Vergleich verschiedener Raten von fehlerhaftem Feedback

nur wenig besser ausfällt als wenn das Ergebnis rein zufällig ausgewählt worden wäre (bei 4 Kontextklassen = 25%). Lediglich NAIVE BAYES erreicht eine relativ hohe Qualität, jedoch mit deutlich verlängerter Lernphase.

Konzeptänderungen

In der folgenden Abbildung 5.22 wurde das Konzept nach der 100. und 200. Iteration abgeändert, jeweils in unterschiedlich starker Ausprägung.

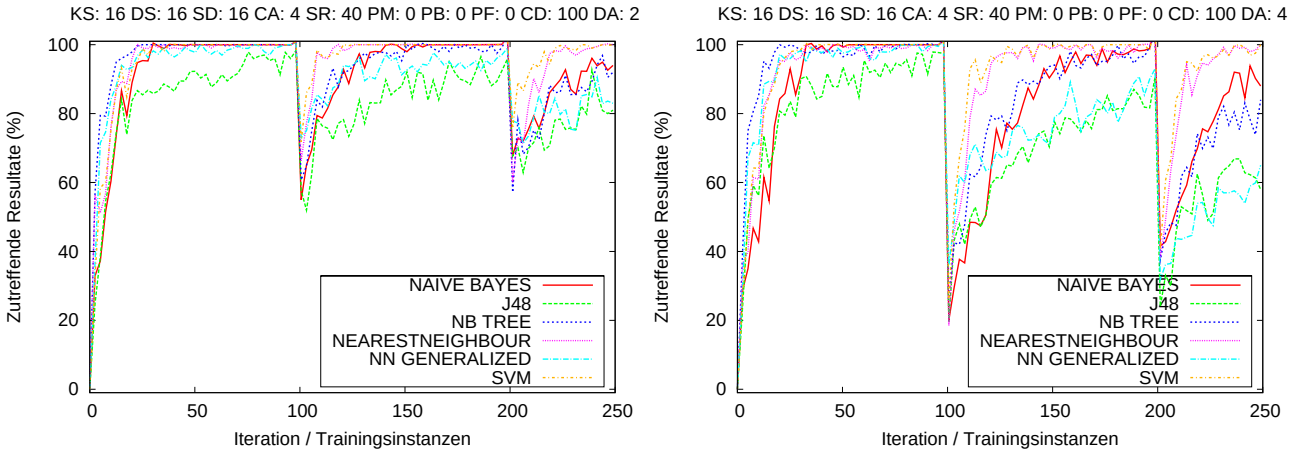
Der Einbruch der Genauigkeit verhält sich relativ zur Stärke der Ausprägung der Konzeptänderung. Bei 4 betroffenen Kontextklassen findet ein *Shift* des gesamten Konzeptes statt. Die Eigenschaft der Offline-Lernverfahren, alle Trainingsinstanzen gleichermaßen gewichtet in der Modellgenerierung zu berücksichtigen, wird ebenfalls deutlich. Neue Trainingsinstanzen müssen durch die Häufigkeit ihres Auftretens alte Trainingsinstanzen *überstimmen*. Die Auswertung bestätigt dies – je später eine Konzeptänderung stattfindet, desto länger ist die Anpassungsphase an das neue Konzept (das sogenannte Over-Training).

Durch den großen Konzeptraum ist die Wahrscheinlichkeit, dass nach einer Konzeptänderung das neue Konzept Überschneidungen mit dem alten Konzept aufweist relativ klein, so dass sich das neue Konzept gut von dem alten differenzieren lässt. Im Falle einer größeren Anzahl von Trainingsinstanzen, häufig auftretender Konzeptänderungen oder einem kleineren Konzeptraum wird die Auswirkung von Konzeptänderungen noch deutlicher. In Abbildung 5.22 c) wurde hierzu der Domänenraum der diskreten Attribute halbiert, was zu dem erwähnten Effekt der größeren Überschneidungen der Konzepte führt. Bei den folgenden Abschnitten, bei denen die Auswirkungen von Konzeptänderungen betrachtet und behandelt werden, wird das Szenario mit reduziertem Konzeptraum als Referenz herangezogen.

Optimierungen hinsichtlich Konzeptänderungen

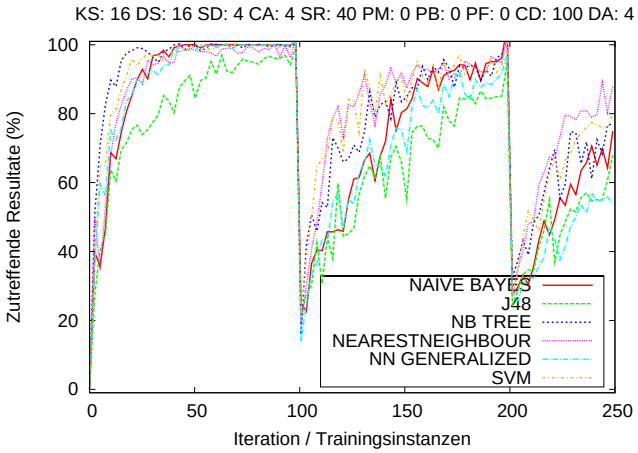
Ein Ansatz, das Problem von Konzeptänderungen anzugehen, besteht darin, die Instanzen selbst einen Zeitstempel beinhalten zu lassen. Somit sind die Instanzen selbst zeitbehaftet und ein Lernverfahren kann implizit zwischen alten und neuen Instanzen unterscheiden. Dieser Ansatz wurde getestet, indem ein weiterer Sensor hinzugefügt wurde, der als Zeitstempel die Nummer der aktuellen Iteration trägt. In der Abbildung 5.23 a) sind die Ergebnisse abzulesen. Während in der Anfangsphase gerade das Verfahren J48 schlechtere Ergebnisse aufweist, zeigt das regelbildende Verfahren NN GENERALIZED eine Verbesserung der Resultate. Allgemein betrachtet hat dieses Vorgehen jedoch nur geringe Auswirkungen auf die durchschnittlich erreichbare Genauigkeit in der Entscheidung.

Ein weiterer Ansatz besteht darin, eine Instanz nur dann der Trainingsmenge zuzuführen, wenn diese Instanz nicht bereits durch das Modell beschrieben wird. Hierzu wird zuvor eine *Test-Evaluation* durchgeführt, indem die



(a) 2 betroffene Kontextklassen

(b) 4 betroffene Kontextklassen

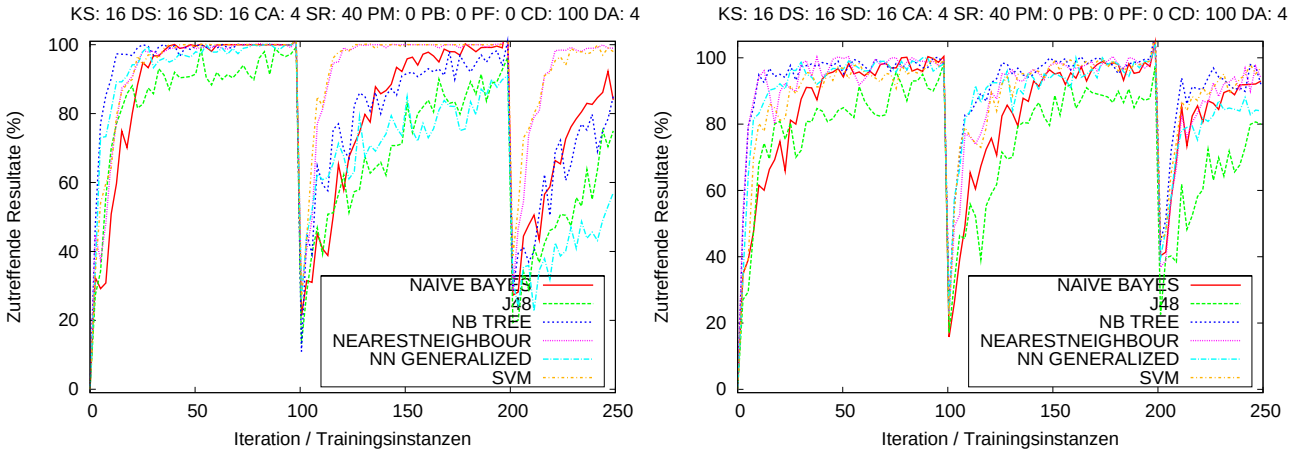


(c) 4 betroffene Kontextklassen unter Reduktion des Domänenraums der diskreten Attribute

Abbildung 5.22: Vergleich verschiedener Ausprägungen von Konzeptänderungen

Instanz auf das bereits bestehende Modell angewendet und überprüft wird, ob das Feedback mit dem Resultat der Auswertung übereinstimmt. Ist dies nicht der Fall, so wird die Instanz der Trainingsmenge zugeführt und das Modell wird neu erstellt beziehungsweise erweitert. Dies führt in erster Linie zu einer Reduzierung der Instanzen innerhalb der Trainingsmenge. Eine verringerte Anzahl von Trainingsinstanzen erhöht die Geschwindigkeit mit der eine Konzeptänderung durch das Modell adaptiert werden kann. Durch jede Konzeptänderung wird die Test-Evaluation häufiger fehlschlagen, wodurch die Trainingsmenge erweitert wird. Bei dem Zeitraum nach dem ersten Auftreten einer Konzeptänderung ist bei vielen Verfahren in Abbildung 5.23 b) deutliche Verbesserung der Qualität zu erkennen.

In der Tabelle 5.4 wird die Verbesserung für einige ausgewählte Verfahren verglichen. Die Werte entsprechen denen aus den Abbildungen 5.23 a) und b). Es werden die Anzahl von Trainingsinstanzen und die Genauigkeit in den kritischen Phasen am Anfang und nach dem Auftreten der Konzeptänderung betrachtet. Es ist zu erkennen, dass bereits eine geringe Anzahl von Instanzen ausreicht, um das Konzept hinreichend zu beschreiben. Tritt eine Konzeptänderung auf, so wird bei der Variante mit der Optimierung die Trainingsmenge erweitert. Hierbei ist vor allem bei NAIVE BAYES und bei NEARESTNEIGHBOUR eine deutliche Steigerung der Qualität zu erkennen. Bei SVM ist die Qualität etwas herabgesetzt, dafür ist durch die Reduzierung der Trainingsmenge die Geschwindigkeit der Modellgenerierung deutlich erhöht. Dieser Effekt wirkt sich in diesem Vergleich nur bei SVM aus, da



(a) Zeitstempel

(b) Update nur bei Fehlklassifikation

Abbildung 5.23: Vergleich verschiedener Optimierungen hinsichtlich Konzeptänderungen

NAIVE BAYES ein inkrementelles Lernverfahren und NEARESTNEIGHBOUR ein Lazy-Learner ist und somit beide unabhängig von der Größe der Trainingsmenge sind.

Verfahren	Schritt 0-50	Schritt 100-150	Schritt 200-250
Ohne Optimierung			
NAIVE BAYES	25/83%	125/68%	225/51%
NEARESTNEIGHBOUR	25/76%	125/87%	225/75%
SVM	25/79% (191 ms)	125/90%/ (260 ms)	225/79% (390 ms)
Mit Optimierung			
NAIVE BAYES	10/74%	24/78%	38/70%
NEARESTNEIGHBOUR	4/89%	10/89%	16/87%
SVM	7/82% (66 ms)	18/85% (144 ms)	28/78% (158 ms)

Tabelle 5.4: Durchschnitt der Messwerte: Anzahl von Instanzen / Genauigkeit (Zeitbedarf für Modellgenerierung)

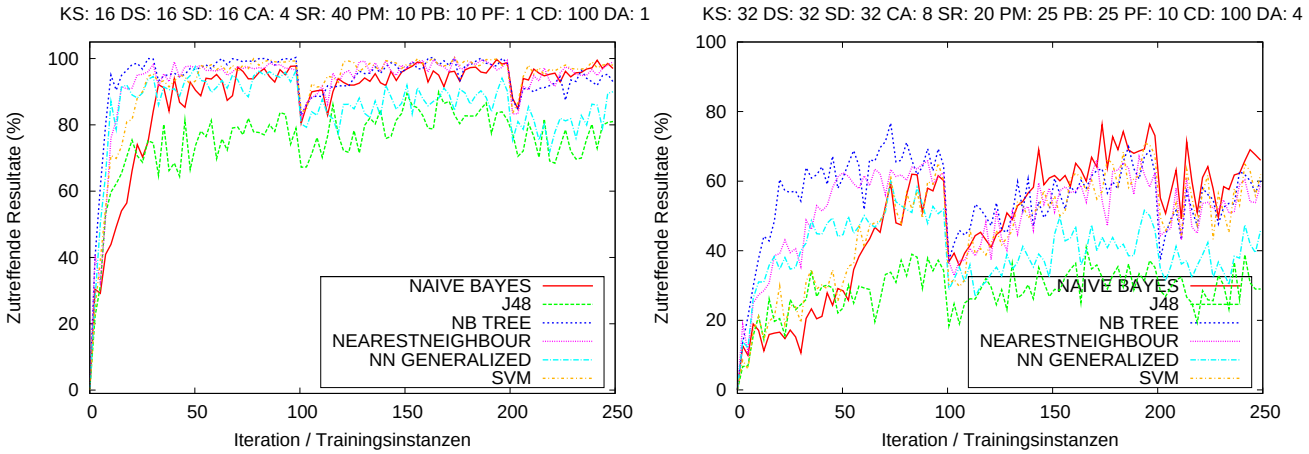
Bei häufigerem Auftreten von Konzeptänderungen wird die Trainingsmenge jedoch weiterhin wachsen, wodurch jeweils der positive Effekt dieser Optimierung verringert wird. Durch eine Verringerung der Anzahl *positiver Instanzen* wird die Robustheit des Modells gegenüber fehlerhaften Daten reduziert.

Kombinierte Szenarien

In diesem Vergleich wurden zwei Szenarien gegenübergestellt, bei denen die zuvor erläuterten Effekte in Kombination miteinander betrachtet wurden:

- **Mäßige Bedingungen:** Das erste Szenario beschreibt eine Umgebung mit mäßigen Bedingungen für das Lernverfahren. Hierzu wurden alle Parameter auf den Standardwerten belassen und Effekte wie fehlerhafte und fehlende Sensoren, sowie fehlerhaftes Feedback auf eine Rate von 10% eingestellt. Zudem tritt eine Konzeptänderung mit der Ausprägung von einem betroffenen Konzept auf vier Konzepten alle 100 Instanzen auf.
- **Schwere Bedingungen:** Das zweite Szenario dagegen stellt eine hohe Herausforderung für die Lernverfahren dar, da dort alle Effekte und Parameter deutlich schwerere Bedingungen beschreiben. Hierzu wurden die Parameter der Informationsgrundlage dahingehend verändert, dass nun (statt wie zuvor 32 Sensoren genutzt

wurden um 4 Konzepte zu beschreiben) insgesamt 64 Sensoren 8 Konzepten gegenüberstehen. Die Rate der Relationen der Sensoren zu den Konzepten wurde zudem von 40% auf 20% reduziert, so dass zwar immer noch gleich viele Sensoren in Relation zu einem Konzept stehen, diese aber aus einer doppelt so großen Menge an Sensoren ermittelt werden müssen. Alle Effekte wurden auf 25% eingestellt und jeweils alle 100 Instanzen tritt nun eine Konzeptänderung auf bei alle Konzeptklassen betroffen sind.



(a) Mäßig Bedingungen

(b) Schwere Bedingungen

Abbildung 5.24: Kombinierte Szenarien

Wie in der Abbildung 5.24 und der Tabelle 5.5 abgelesen werden kann, schneiden unter mäßigen Bedingungen alle Verfahren gut bis sehr gut ab und erreichen eine Genauigkeit von 80% bis knapp 100%. Bei den schweren Bedingungen unterscheidet sich die Leistung der Verfahren jedoch stärker voneinander. Während NAIVE BAYES, NT TREE, NEARESTNEIGHBOUR und SVM trotz erschwerter Bedingungen eine Genauigkeit von ca. 50% bis 60% erreichen, bleiben die Verfahren J48 und NN GENERALIZED eher im Bereich von 30% bis 40%.

Szenario	**NAIVE BAYES**	**J48**	**NB TREE**	**NEARESTN.**	**NN GEN.**	**SVM**
$\mu_{Genauigkeit}$ Mäßig	91,20%	80,14%	97,04%	94,88%	89,26%	94,04%
$\mu_{Genauigkeit}$ Schwer	48,10%	27,68%	52,94%	50,58%	34,98%	43,38%

μ_t durchschnittlich gemessener Zeitbedarf.

Tabelle 5.5: Genauigkeit der Verfahren unter verschiedenen Bedingungen

5.4.8 Fazit

Die Bewertung der Verfahren in der Tabelle 5.6 erfolgt anhand der zuvor festgelegten Rahmenbedingungen: Die *Qualität* des Modells beschreibt die Klassifikationsgenauigkeit unter den verschiedenen Eigenschaften der Informationsgrundlage. Die Zeile *Generierung* beschreibt den Aufwand, welcher zur Generierung des Modells benötigt wurde. Der Zeitbedarf für die Auswertung der Modelle wurde in Kombination mit der Skalierbarkeit im Bezug zu den Abhängigkeiten zu Größen der Informationsgrundlage betrachtet. Die Ergebnisse dieser Betrachtung sind in der Zeile *Evaluation* enthalten.

Generierung

Den Zeitbedarf zur Generierung der verschiedenen Modelle variiert je nach eingesetztem Verfahren sehr stark. In den verglichenen Szenarien, liegt der Zeitbedarf zwischen 0 und 2000 ms pro Iteration. Der Lazy-Learner NEA-

Eigenschaft	NAIVE BAYES	J48	NB TREE	NEARESTNEIGHBOUR	NN GENERALIZED	SVM
Qualität	++	+	+++	++	+	+++
Generierung	+++	++	+	+++	+++	+
Evaluation	++	+++	++	+	+++	++

Tabelle 5.6: Bewertung und Vergleich existierender Verfahren

RESTNEIGHBOUR und die inkrementellen Offline-Learner NN GENERALIZED und NAIVE BAYES erzeugen pro Iteration konstant – mit 0 bis 10 ms – sehr geringen Aufwand. Im Gegensatz dazu erhöht sich der Aufwand der Offline-Lernverfahren mit der Menge der Instanzen in der Trainingsmenge. Trotzdem liegen innerhalb der Kategorie der Offline-Lernverfahren erhebliche Unterschiede zwischen dem Zeitbedarf der verschiedenen Verfahren. Während der Zeitbedarf von J48 ebenfalls unterhalb der 10 ms ist, liegt der Aufwand von SVM und NBTREE um den Faktor 10 bis 100 darüber.

Evaluation

Insgesamt kann festgestellt werden, dass alle herangezogenen Verfahren in den betrachteten Szenarien keine Probleme hatten, die geforderten Rahmenbedingungen für den Zeitbedarf bei der Auswertung einer einzelnen Instanz einzuhalten. Je nachdem welche Größen der Informationsgrundlage bedarfsweise skaliert werden müssen, werden jeweils diejenigen Verfahren erhöhten Aufwand erzeugen, welche von diesen Größen abhängig sind. Daher muss bei einer Skalierung des Problems auf die geeignete Auswahl von Lernverfahren geachtet werden. NEARESTNEIGHBOUR zeigte in den getesteten Größenordnungen keine nennenswerte Verzögerung bei der Auswertung. Für Szenarien, welche vergleichbare Mengen an Sensoren und Trainingsinstanzen aufweisen, wären daher Verfahren der Kategorie Lazy-Learner prinzipiell auch geeignet.

In fast allen Verfahren ist die Anzahl von Sensoren eine Größe, welche den Zeitbedarf der Auswertung maßgeblich beeinflusst. Daher sei an dieser Stelle wieder auf die Anforderung an die Suche verwiesen, möglichst nur relevante Sensoren zu identifizieren und im Suchergebnis zu repräsentieren.

Qualität

Die angewendeten Verfahren nutzen implizit eine Art Mehrheitsentscheid. Einzelne fehlerhafte oder fehlende Aussagen können gut durch andere Aussagen ergänzt werden. Dies ist meist bei den Darstellung mit bis zu 25% fehlenden oder fehlerhaften Daten zu erkennen. Obwohl hier schon einige Informationen fehlen oder abweichen, können relativ gute Entscheidungen getroffen werden. Wächst die Zahl der fehlerhaften Aussagen jedoch über eine kritische Menge, so sinkt die Qualität deutlich. Dies wird deutlich wenn man den Verlauf dieser Eigenschaften von 0% über die 25% hin zu der 50% Darstellung betrachtet. Bei 50% verzeichnen alle Verfahren einen vergleichsweise deutlichen Einbruch in ihrer Genauigkeit.

Die Geschwindigkeit, mit der die Verfahren ein Konzept erlernen, beziehungsweise die Form der Lernkurve bis zum Erreichen einer hohen Klassifikationsgenauigkeit ist ebenfalls ein Aspekt in der Bewertung der Qualität.

Für die bisher betrachteten Verfahren stellen Konzeptänderungen eine große Herausforderung dar. Je später ein solcher Fall eintritt, desto deutlicher verlangsamt sich der Anstieg der Lernkurve durch das Over-Training. Da in den angestrebten Szenarien auch über einen längeren Zeitraum hinweg Konzeptänderungen auftreten können, muss dieses Problem geeignet behandelt werden. In diesem Zusammenhang bieten sich folgende Möglichkeiten an:

- Kombination aus einem Meta-Verfahren zur Berücksichtigung von Konzeptänderungen und einem darin gekapselten Offline-Lernverfahren (siehe Abschnitt 5.3.3).
- Online-Lernverfahren (siehe Abschnitt 5.3.3).

Beide Möglichkeiten werden im Folgenden behandelt.

5.5 Umsetzung eines Fensterverfahrens zur Evaluierung von Offline-Lernverfahren

Aus Abschnitt 5.3.2 wird deutlich, dass Offline-Verfahren alleine nicht für den Einsatz in dem angestrebten Anwendungsszenario geeignet sind. Zum einen ist für die Skalierbarkeit des Systems eine Limitierung der Anzahl der Trainingsinstanzen erforderlich, zum anderen kann auf diese Weise ein Mechanismus zur Behandlung von Konzeptänderungen geschaffen werden.

Ein wesentlicher Ansatz der in diesem Zuge angesprochen wurde, besteht in der Kapselung durch ein Meta-Verfahren, welches das enthaltene Offline-Lernverfahren (den *Basisklassifikator*) durch Mechanismen wie Fenster-, oder Gewichtungsverfahren ergänzt.

Im Rahmen der zuvor betrachteten Offline-Verfahren existiert jedoch keine geeignete Umsetzung derartiger Verfahren. Im Zuge dieser Arbeit wurde ein entsprechendes Fensterverfahren entwickelt und umgesetzt, welche sich mit den betrachteten Offline-Verfahren kombinieren lässt.

5.5.1 Grundprinzipien einer Fensterverwaltung

Ein Fensterverfahren beruht auf der Annahme, dass jeweils nur eine bestimmte Menge an zuletzt erhaltenen Trainingsinstanzen gültig ist, beziehungsweise auf dem aktuellen Konzept beruht. Jene Trainingsinstanzen, die für die Modellgenerierung zur Verfügung stehen bzw. der damit beanspruchte Speicherplatz werden im Folgenden als Fensterinhalt oder Fenstergröße bezeichnet. Die Abbildung 5.25 veranschaulicht das Prinzip eines Fensterverfahrens.

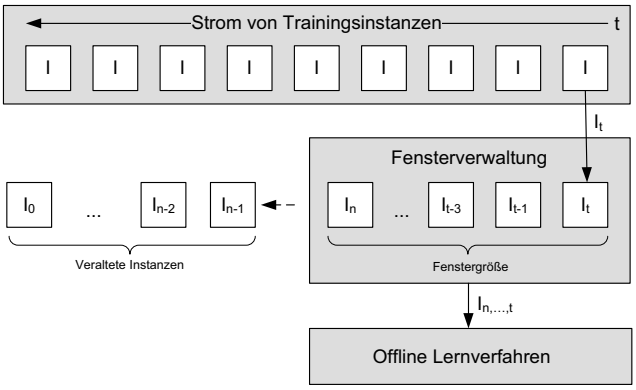


Abbildung 5.25: Mechanismus der Fensterverwaltung [WK96].

Welche und wie viele Trainingsinstanzen zur Modellgenerierung genutzt werden sollen, bestimmt die Fensterverwaltung. Damit besitzt die Fensterverwaltung einen entscheidenden, wenn auch nur mittelbaren Einfluss auf die Klassifikation bzw. das Zielkonzept sowie auf die Genauigkeit des Systems.

Hierzu erhält die Fensterverwaltung bei jedem Trainingsschritt eine neue Instanz. Die Fensterverwaltung hat nun die Aufgabe zu entscheiden, ob:

1. das Fenster um ein Element (die neue Instanz) *vergrößert* werden soll ($n_t = n_{t-1} + 1$).
2. die Fenstergröße n *gleichbleibt*. Dazu wird das letzte Element I_n, entfernt und das neue Element I_t eingefügt.
3. oder aber das Fenster um eine bestimmte Menge m Instanzen *verringert* werden soll.

Eine kleine Fenstergröße erhöht die Geschwindigkeit der Anpassung des Systems. Dies ist vorteilhaft, um Konzeptänderungen möglichst optimal zu behandeln. Eine große Fenstergröße beschreibt ein Konzept mit einer potentiell höheren Genauigkeit während stabiler Phasen, ohne Konzeptänderungen. Generell lassen sich mit diesem Ansatz vier Verwaltungstypen für Instanzen nachbilden:

- **Full-Memory:** Speichert alle Instanzen.
- **No-Memory:** Behält nur die aktuelle Instanz.
- **Fixed-Size:** Hält eine feste Menge von Instanzen vor.
- **Adaptive-Size:** Ermittelt eine variable Fenstergröße.

Während auf der einen Seite Full-Memory direkt das Offline-Verfahren transparent (d.h. das Fensterverfahren hat keine Auswirkung auf das Ergebnis) nutzbar macht, nutzt der No-Memory Ansatz jeweils nur die neue Instanz für die Generierung des Modells und liefert bei Batch-Learnern entsprechend unzureichende Ergebnisse im Hinblick auf die Genauigkeit. Die letzten beiden Ansätze Fixed-Size und Adaptive-Size erlauben einen Vergleich aller Lernverfahren unter Beschränkung der Trainingsinstanzen und dessen Auswirkungen auf das Lernverhalten.

Fixed-Size

Fixed-Size bietet die Möglichkeit eine Obergrenze für die Anzahl der Instanzen zu setzen, um beispielsweise den, über die Trainingsschritte hinweg, stetig anwachsenden Speicherbedarf zu begrenzen. Dies beinhaltet jedoch die Angabe einer vorweg fest definierten Obergrenze. Die Anzahl der Instanzen, welche benötigt werden, um ein Konzept zu beschreiben, variiert jedoch mit der Komplexität des Konzeptes, welche sich (gerade in Anbetracht unbekannter Nutzerkonzepte und unbekannter Menge verfügbarer Sensoren) nur schwer vorhersagen lässt.

Adaptive-Size

Die vierte Variante setzt daher keine feste Obergrenze voraus, sondern kann die Fenstergröße bedarfsweise anpassen. Hierzu muss jedoch bestimmt werden, welche Menge von Trainingsinstanzen gehalten werden soll. Eine exakte Beantwortung der Frage wäre nur mit dem vollständigen Wissen über das unbekannte Konzept möglich. Entsprechend gilt es diesen Wert anhand anderer Parameter zu ermitteln [KR98]. Eine dynamische Anpassung der Fenstergröße bietet eine höhere Flexibilität im effizienten Umgang mit unterschiedlichen Kontextdimensionen und Anwendungsszenarien. Auf der anderen Seite beinhaltet ein solcher Ansatz auch ein erhöhtes Risiko für Fehleinschätzungen. Die Analyse des Effekts von Konzeptänderungen und die dazugehörige Abbildung 5.22 verdeutlichten, dass mehr Trainingsinstanzen nicht immer auch eine höhere Genauigkeit bewirken. Die Menge der Instanzen, im Folgenden als Fenstergröße bezeichnet, sollte der aktuellen Lernsituation angepasst sein:

- **Lernphase:** Vergrößerung des Fensters.
- **Stabiles Konzept:** Beibehaltung der Fenstergröße.
- **Konzeptwechsel:** Reduzierung der Fenstergröße.

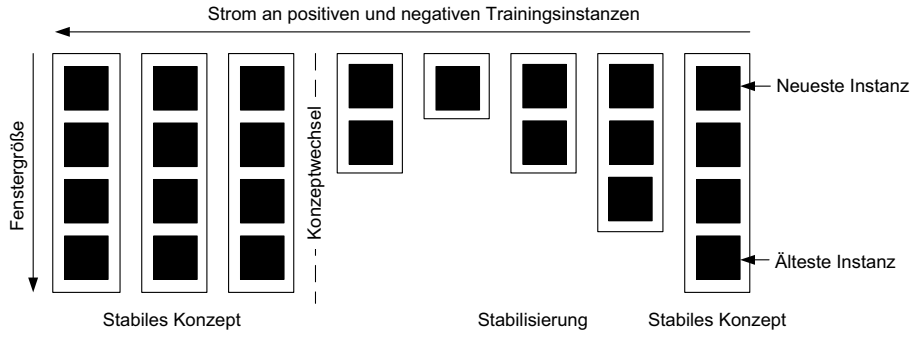


Abbildung 5.26: Veränderung der Fenstergröße je nach Lernsituation

Dieses Verhalten ist exemplarisch in Abbildung 5.26 dargestellt. Die Fensterverwaltung hat somit nicht nur einen Einfluss auf die Genauigkeit des Systems, sondern sie regelt ebenso den Speicherbedarf für die Speicherung der Trainingsinstanzen und des Modells. Somit ist bei einer Einschätzung der Performance einer solchen Verwaltung zusätzlich der Vergleich zwischen dem Speicherbedarf und der Genauigkeit zu betrachten.

5.5.2 Verwandte Arbeiten

MetaL

MetaL [Wid97] ist ein Ansatz für ein Meta-Lernverfahren zur Erweiterung eines Basisklassifikators um Funktionen zur Berücksichtigung von zeitlichen Aspekten (*Context Clues*). MetaL nutzt hierzu eine konstante Fenstergröße. Dieser Ansatz sieht vor, anhand statistischer Methoden eine Menge von Instanzen aus der Trainingsmenge zu bestimmen, welche Abhängigkeiten zur der Klassifikationsinstanz aufweisen. Diese Menge von Instanzen wird dynamisch zum Aufbau eines Modells durch den Basisklassifikator genutzt, welcher schließlich die Klassifikationsinstanz auswertet.

MetaL(B) (Meta-Learning - Bayes) bedient sich dem Bayes Klassifikator als Basis, während MetaL(IB) (Meta-Learning - Instance Based) den IB2-Algorithmus als Basisklassifikator nutzt (IB2 wird später in Abschnitt 5.6.1 erläutert). Weiterhin bietet MetaL(IB) zusätzliche Mechanismen wie beispielsweise *Exemplar Weighting* (MetaL(IB)-EW) oder *Feature Weighting* (MetaL(IB)-FW). Diese Mechanismen nehmen zusätzlich Einfluss auf die Distanzfunktion von IB2 [MM00b].

Learn++.NSE

Ein weiterer interessanter Ansatz für ein Meta-Verfahren zur Erweiterung eines (offline) Basisklassifikators um die Eigenschaften eines Online-Learners wird durch das Verfahren *Learn++.NSE* vorgestellt [KMP07]. Dieser Ansatz basiert sowohl auf der Gewichtung von Instanzen, als auch auf einer gewichteten Mehrheitsentscheidung über mehrere Modelle hinweg (im Weiteren bezeichnet als *Gesamtmodell*). In jeder Iteration werden alle Instanzen auf das aktuelle Gesamtmodell angewandt, wobei die Gewichte der Instanzen angepasst werden, indem bei fallender Genauigkeit die Gewichte der korrekt klassifizierten Trainingsinstanzen erhöht werden. Alle im Gesamtmodell enthaltenen Modelle werden anschließend anhand der Trainingsinstanzen ausgewertet, wobei fehlerhafte Entscheidungen mit der Gewichtung der jeweiligen Instanz verrechnet und zu einer Aussage über die aktuelle Genauigkeit des jeweiligen Modells aufsummiert werden. Fällt die Genauigkeit des letzten Modells unter einen Grenzwert, so wird ein neues Modell erstellt und dem Gesamtmodell hinzugefügt. Schließlich wird die Gewichtung für die Mehrheitsentscheidung des Gesamtmodells anhand der Genauigkeit der einzelnen Modelle angepasst, wobei die Modelle mit höherer Genauigkeit stärker gewichtet werden.

Dieser Ansatz hat die Eigenschaft, dass er bei einer möglichen Konzeptänderung sofort ein neues Modell erstellen und verwenden kann. Fällt die Genauigkeit des neuen Modells jedoch schlechter aus als die der vorangegangenen, wird die Gewichtung des neuen Modells reduziert und die der alten Modelle angehoben.

Der Ansatz ist stark auf die Optimierung der Qualität ausgerichtet, benötigt jedoch durch die Verarbeitung mehrerer Modelle einen deutlich erhöhten Aufwand. Ebenso ist die Menge der Modelle und Trainingsinstanzen nicht limitiert. Um diese Anforderung zu erfüllen, könnte möglicherweise das Entfernen von Instanzen und Modellen mit niedrigen Gewichten hinzugefügt werden.

Ansatz von FLORA 2

Ein Ansatz für eine *Window Adjustment Heuristic* (WAH) wurde von Widmer und Kubat 1996 vorgestellt und in das Online-Lernverfahren FLORA 2 integriert [WK96]. Diese WAH (siehe Abbildung 5.27) bedient sich einerseits der aktuellen Klassifikationsgenauigkeit und andererseits syntaktischen Eigenschaften des Modells, welche Aussagen über die Stabilität des Modells zulassen. Damit greift dieser Ansatz auf spezielle Eigenschaften von FLORA zurück und lässt sich nicht direkt als Ansatz für ein Meta-Verfahren anwenden oder auf andere Arten von Modellen übertragen.

IF $StabilitaetModel < a$ OR ($Genauigkeit < b$ AND $GenauigkeitAbfallend$)
  THEN $Konzeptdrift$
ELSE IF $StabilitaetModel > 2 * c$ AND $Genauigkeit > b$
  THEN $Sehr\ stabiles\ Modell$
ELSE IF $StabilitaetModel > c$ AND $Genauigkeit > b$
  THEN $Stabiles\ Modell$
ELSE $Lernphase$

a, b, c benutzerdefinierte Grenzwerte.

Abbildung 5.27: WAH-Ansatz von Widmer und Kubat

Liegt ein *sehr stabiles* Modell vor, so wird zusätzlich die Fenstergröße in jedem Lernschritt um eine Instanz verkleinert, bis nur noch ein *stabiles* Modell vorliegt. Findet sich ein ausreichend stabiles Zielkonzept, wird die Größe des Fensters konstant gehalten. Tritt jedoch keine der genannten Situationen ein, dann geht die WAH von einer bisher zu geringen Stabilität des Zielkonzeptes aus und vergrößert das Fenster um eine Einheit. Diese Beschränkung der Fenstergröße, verringert die Gefahr von Over-Training und führt zu einer verbesserten Reaktionsfähigkeit bei Konzeptänderungen.

Die Grenzwerte der WAH des FLORA-Systems bleiben über die Gesamtlaufzeit hinweg konstant. Zur Festlegung dieser Grenzwerte wird ein hohes domänenspezifisches Vorwissen benötigt oder diese Werte müssen zuvor experimentell ermittelt werden [WK96].

Bei der von Widmer und Kubat selbst durchgeführten Betrachtung der Funktionsweise ihres Verfahrens hat sich jedoch herausgestellt, dass die WAH keineswegs optimale Verhaltensweisen aufzeigte [WK96].

Ansatz von Klingenberg und Renz

Klingenberg (1998) [Kli98] beschäftigt sich mit der Anwendbarkeit von Lernsystemen zur Informationsfilterung von Texten. Die thematische Verwandtschaft zu dieser Arbeit findet sich in der speziellen Ausrichtung auf Online-Lernverfahren im Zusammenhang mit Konzeptwechseln. Er analysiert ebenfalls verschiedene Indikatoren zur Erkennung der Wechsel und integriert diese in den Entwurf einer Fensterverwaltungsheuristik.

Die von Klingenberg und Renz vorgestellte WAH nutzt andere Indikatoren zur Erkennung von Konzeptänderungen und eine Reihe von nutzerspezifischen Faktoren. Diese Faktoren werden mit den zuletzt ermittelten Klassifikationsgenauigkeiten des Modells verrechnet.

IF $(Genauigkeit_t < \mu_{Genauigkeit} - a * \sigma_{Genauigkeit})$ AND $(Genauigkeit_t < b * Genauigkeit_{t+1})$
THEN $Konzeptshift$
ELSE IF $(Genauigkeit_t < \mu_{Genauigkeit} - a * \sigma_{Genauigkeit})$
THEN $Konzeptdrift$
ELSE $Lernphase$

a, b, c	benutzerdefinierte Grenzwerte.
μ	bezeichnet den durchschnittlichen Wert.
$sigma$	bezeichnet die Standardabweichung.

Abbildung 5.28: WAH-Ansatz von Klingenberg und Renz

Dieser Ansatz, wie er in Abbildung 5.28 dargestellt wird, ist bei der Wahl der Faktoren weniger auf Domänenwissen ausgelegt, als auf das relative Empfinden des Nutzers, auf eine Kontextverschiebung zu reagieren.

Diese höhere Domänenunabhängigkeit birgt allerdings den Nachteil eines häufigeren Fehlverhaltens des Systems. Wenn beispielsweise durch ein stabiles Modell eine sehr hohe durchschnittliche Genauigkeit, sowie eine geringe Standardabweichung ermittelt werden, reichen bereits kleinste Änderungen, um anhand dieses Verfahrens Konzeptdrift zu signalisieren.

Zudem sieht dieser Ansatz vor, Konzeptshifts zu behandeln, indem alle Trainingsinstanzen entfernt werden. Dieses Vorgehen beinhaltet allerdings auch das Risiko, dass das gesamte Modell im Falle von Fehleinschätzungen verloren geht. Allerdings kann dieser Ansatz im Falle eines stabilen Konzeptes das Fenster vergrößern, was einerseits zu erhöhtem Speicherbedarf führt, andererseits auch wieder die Gefahr des Over-Trainings birgt.

5.5.3 Erweiterung des ausgewählten Ansatzes

Der Ansatz zur Bestimmung der Fenstergröße, welcher im Rahmen dieser Arbeit umgesetzt wurde, nimmt die Ideen von Widmer und Kubat sowie von Klingenberg und Renz auf und erweitert diese.

Zur Bestimmung der Fenstergröße kann sich die Fensterverwaltung nur auf bereits verarbeitete Trainingsinstanzen und Aussagen über deren Einfluss auf die Genauigkeit des Modells stützen. Anhand verschiedener Parameter kann die aktuelle Lernphase beschrieben werden. Welche Parameter hierzu bestimmt werden, wird in Abschnitt 5.5.3 erläutert. Je nach detektierter Lernphase wird entschieden, wie sich die Fenstergröße verhalten soll.

Zudem ist es notwendig, eine bestimmte Menge von Instanzen in die Ermittlung der Lernphase einzubeziehen. Die Größe dieser Menge ist ausschlaggebend für die Reaktionsgeschwindigkeit der Fensterheuristik auf Konzeptänderungen. Die Menge, um die die Fenstergröße, im Falle einer detektierten Konzeptänderung, reduziert wird, hat ebenfalls einen Effekt auf das Lernverhalten.

Parameter zur Bestimmung der aktuellen Lernphase

Als Parameter zur Ermittelung der aktuellen Lernphase werden die aktuelle Klassifikationsgenauigkeit, sowie deren Standardabweichung und die Shannon-Entropie herangezogen. Die Parameter werden in diesem Abschnitt erläutert.

Aus der Menge aller Trainingsinstanzen $X = \{x_1, x_2, .., x_s\}$ wird nur ein Fenster über die zuletzt hinzugefügten Trainingsinstanzen $F = \{x_{s-m}, x_{s-m+1}, \ldots, x_s\}$ betrachtet. Je nachdem ob das zu betrachtende Fenster größer ist als

die Anzahl der Trainingsinstanzen, welche bis zu dem betrachteten Zeitpunkt t gesammelt wurden, wird entweder die Fenstergröße betrachtet oder die Anzahl der bisher gesammelten Trainingsinstanzen $s = |X|$:

$$n = \begin{cases} m & falls \;\; m < s \\ s & falls \;\; s \leq m \end{cases} \tag{5.5}$$

Aktuelle Klassifikationsgenauigkeit

Vor der Integration in das Modell wird jede Trainingsinstanz x mit dem aktuell vorliegendem Modell $f()$ klassifiziert und mit dem zu der Trainingsinstanz assoziierten Feedback $h()$ verglichen. Die Funktion $hit()$, welche für diese Auswertung genutzt wird, ist wie folgt definiert:

$$hit(x) = \begin{cases} 1 & falls \;\; f(x) = h(x) \\ 0 & falls \;\; f(x) \neq h(x) \end{cases} \tag{5.6}$$

Um anhand dieser binären Funktion eine Aussage über die durchschnittliche Klassifikationsgenauigkeit des Modells zu erhalten, müssen mehrere dieser binären Aussagen miteinander verrechnet werden.

Die Klassifikationsgenauigkeit acc für den Zeitpunkt t lässt sich wie folgt berechnen:

$$acc_t = \frac{1}{n} \sum_{i=1}^{n} \quad hit(x_{t-i}) \tag{5.7}$$

Die Anzahl n der hierbei betrachteten Instanzen hat einen Einfluss auf die Anpassungsgeschwindigkeit des ermittelten Durchschnittswertes an den realen Durchschnittswert. Je nachdem wie groß dieser Parameter gewählt wird, desto *träger* reagiert das System.

Es wurde zudem eine Gewichtung der Genauigkeit eingeführt, so dass die Genauigkeit der letzten Iterationen höher gewichtet wurde, als die weiter zurückliegender. Diese Gewichtung reduziert die Trägheit der Funktion acc_t.

Es wurden in verschiedenen Tests mehrere Formen der Gewichtung getestet. Verglichen wurden dabei die lineare, exponentielle und logarithmische Verteilung der Gewichte. Die besten Ergebnisse wurden von der logarithmischen Verteilung der Gewichte erzielt, dicht gefolgt von der linearen, während eine exponentielle Verteilung zu schnelle Reaktionen auf einzelne Fehlentscheidungen verursachte.

Die Funktion für die logarithmisch gewichtete Bestimmung lässt sich wie folgt beschreiben:

$$accWeighted_t = \frac{1}{\sum_{i=1}^{n} \ln(i+1)} \sum_{i=1}^{n} (hit(x_{t-i}) * \ln(i+1)) \tag{5.8}$$

Standardabweichung

Die Standardabweichung ist ein Maß für die Streuung einer Variablen um ihren Mittelwert. In diesem Fall wird die Standardabweichung der Klassifikationsgenauigkeit betrachtet (deren Bestimmung zuvor beschrieben wurde):

$$meanAcc_t = \frac{1}{n} \sum_{i=1}^{n} accWeighted_{t-i} \qquad stdErr_t = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (accWeighted_{t-i} - meanAcc_t)^2} \tag{5.9}$$

Shannon-Entropie

Gerade zu Beginn oder nach einem Konzeptwechsel ist die Entscheidung eines Lernverfahrens von hoher Unsicherheit und Zufall geprägt. Die von der Fensterverwaltung genutzte Größe zur Erkennung eines solchen Verhaltens basiert auf der Entropie nach Shannon [Sha01]. Die Shannon-Entropie ist ein Maß für die Menge an Zufallsinfor-

mationen, die in einer Folge von Informationen steckt. Die Shannon-Entropie H einer Informationsfolge I über dem Alphabet A ist dann wie folgt definiert.

$$H(I) = -\sum_{i=1}^{|A|} p_i * \log_{|A|}(p_i) \tag{5.10}$$

p_j beschreibt die Wahrscheinlichkeit des Auftretens des j-ten Zeichens des Alphabets in der Informationsfolge.

Ist die Shannon-Entropie nahe an 1 so kann die Informationsfolge als zufällig bezeichnet werden, da jedes Zeichen mit gleicher Wahrscheinlichkeit auftritt [HRSV00]. Die Fensterverwaltung bestimmt die in der Folge von Klassifikationsergebnissen enthaltene Entropie. Aufgrund eines binären Alphabetes der Funktion $hit()$ vereinfacht sich die oben stehende Formel wie folgt.

$$H(I) = -p_0 * \log_2(p_0) - p_1 * \log_2(p_1) \quad p_0 = 1 - p_1 \quad H(I) = -p_1 * \log_2(p_1) - (1 - p_1) * \log_2(1 - p_1) \tag{5.11}$$

Die Variable p_1 lässt sich durch das Abzählen der Fälle von $hit(X) = 1$ im Fenster ermitteln und entspricht daher dem Ergebnis der Funktion $acc()$.

$$shannon_t = H(I) = -acc_t * \log_2(acc_t) - (1 - acc_t) * \log_2(1 - acc_t) \tag{5.12}$$

Bei der Umsetzung dieser Funktion müssen schließlich noch die Fälle $acc_t = 0$ und $acc_t = 1$ geeignet behandelt werden – die Entropie ist in diesen Fällen 0.

Angepasstes Verfahren zur Fensterverwaltung

Im Rahmen dieser Arbeit wurde die WAH-Ansatz von Widmer und Kubat [WK96] herangezogen und erweitert. Die Funktionsweise der erweiterten Window Adjustment Heuristic (siehe Abbildung 5.29) basiert grundlegend auf den zuvor angeführten Parametern, der Klassifikationsgenauigkeit, sowie der Shannon-Entropie. Deren Aussage werden zur Bestimmung verschiedener Phasen des Lernens genutzt. Diese Phasen werden unter anderem durch die Grenzwerte a für die Shannon-Entropie und b für die Genauigkeit beschrieben.

IF $(shannon_t > a)$
  THEN $Lernphase$
ELSE IF $acc_t < b$ AND $acc_t < (acc_{t-1} - stdErr_t)$
  THEN $Konzeptdrift$
ELSE IF $acc_t < b$
  THEN $Lernphase$
ELSE $Stabiles\ Modell$

a, b benutzerdefinierte Grenzwerte.

Abbildung 5.29: Erweiterung des WAH-Ansatzes

Die Bestimmung der Shannon-Entropie läuft ungeachtet der Reihenfolge der Trainingsinstanzen ab. Daher ist die Aussage der Shannon-Entropie hinsichtlich auftretender Konzeptdrifts zwar nutzbar, jedoch relativ träge. Die Entropie eignet sich jedoch gut, um die instabilen Phasen (welche beispielsweise gerade am Anfang oder nach einem Konzeptdrift auftreten) von beabsichtigten Konzeptdrifts zu differenzieren.

Die nachfolgende Bestimmung einer Konzeptänderung wird ähnlich wie bei dem Ansatz von Klingenberg und Renz durchgeführt. Hierbei wurde jedoch die Bestimmung von Konzeptshifts ausgelassen. Je nach Ausprägung des Konzeptdrifts ist zu erwarten, dass in den nachfolgenden Iterationen weitere Schritte zur Behandlung von Konzeptdrift erfolgen. Bei einem Konzeptshift würden demnach mehrere Konzeptdrift-Behandlungen, bei denen sukzessive die Fenstergröße reduziert werden würde, die Behandlung von Konzeptshift ersetzen.

Aus dem Ansatz von Widmer und Kubat wurde die Berücksichtigung stabiler Modelle übernommen. Indem die Fenstergröße bei stabilen Modellen unverändert bleibt, wird ein Over-Training vermieden.

5.5.4 Wahl der Parameter

Die Fensterverwaltung der neuen WAH benötigt die Bestimmung zweier grundlegender Parameter (siehe Verfahren in Abbildung 5.29):

- **Grenzwert für die Shannon-Entropie (a):** Je niedriger dieser Wert gewählt wird, desto eher findet die Behandlung von Konzeptänderungen auch in instabilen Phasen statt.
- **Grenzwert für die Genauigkeit (b):** Je höher dieser Wert gewählt wird, desto stärker versucht das System die entsprechende Genauigkeit durch eine Vergrößerung des Fensters zu erreichen und desto empfindlicher reagiert die Konzeptdrift-Bestimmung auf Schwankungen in der Genauigkeit.

Dazu kommen zwei weitere Parameter, welche die Funktionsweise der WAH beeinflussen:

- **Fensterreduktionsrate:** Je höher dieser Wert gewählt wird, desto ausgeprägter findet die Behandlung von Konzeptänderungen statt.
- **Anzahl der Elemente zur Berechnung der Genauigkeit:** Je höher dieser Wert gewählt wird, desto träger reagiert die Bestimmung der Genauigkeit auf Schwankungen.

Da kein analytisches Verfahren zur Festlegung der Grenzwerte in Abhängigkeit zum betrachteten Szenario existiert, wurde über eine Reihe von Tests ermittelt, dass ein Grenzwert für die Shannon-Entropie bei $a = 0.8$ und ein Grenzwert von $b = 0.95$ für die Genauigkeit für die betrachteten Szenarien sehr ordentliche Resultate erzielte. Die Fensterreduktionsrate wurde abhängig von der Differenz zwischen der aktuellen Genauigkeit und der durchschnittlichen Genauigkeit gewählt $n_t = n_{t-1} * (meanAccWeighted - accWeighted)$. Durch die logarithmische Gewichtung der Elemente zur Berechnung der Genauigkeit, ist die Anzahl nicht so ausschlaggebend wie bei einer gleichmäßigen Gewichtung. Eine Anzahl von 25 Elementen zur Berechnung der Genauigkeit lieferte hierfür gute Resultate.

5.5.5 Auswertung

Die Auswertung findet durch einen Vergleich mit den Ansätzen ohne Fenstermechanismus statt. Die Daten welche zur Auswertung genutzt werden, wurden analog zu denen aus Abschnitt 5.4 generiert. Als Basisklassifkator wurden die Verfahren NAIVE BAYES, NEARESTNEIGHBOUR und SVM herangezogen, welche schon bei den vorangegangenen Analysen relativ gute Eigenschaften im Zusammenhang mit Konzeptänderungen aufwiesen.

Als Szenario wurden die gleichen Parameter wie für die Analyse der Auswirkung von Konzeptänderungen in Abschnitt 5.4.7 ausgewählt.

Feste Fenstergröße

Zu Beginn wurde der Ansatz einer Fensterverwaltung mit fester Fenstergröße getestet. Wie in Abbildung 5.30 zu erkennen ist, hat die Nutzung einer Fensterverwaltung meist einen positiven Einfluss auf die Klassifikationsgenauigkeit im Zusammenspiel mit dem Auftreten von Konzeptänderungen.

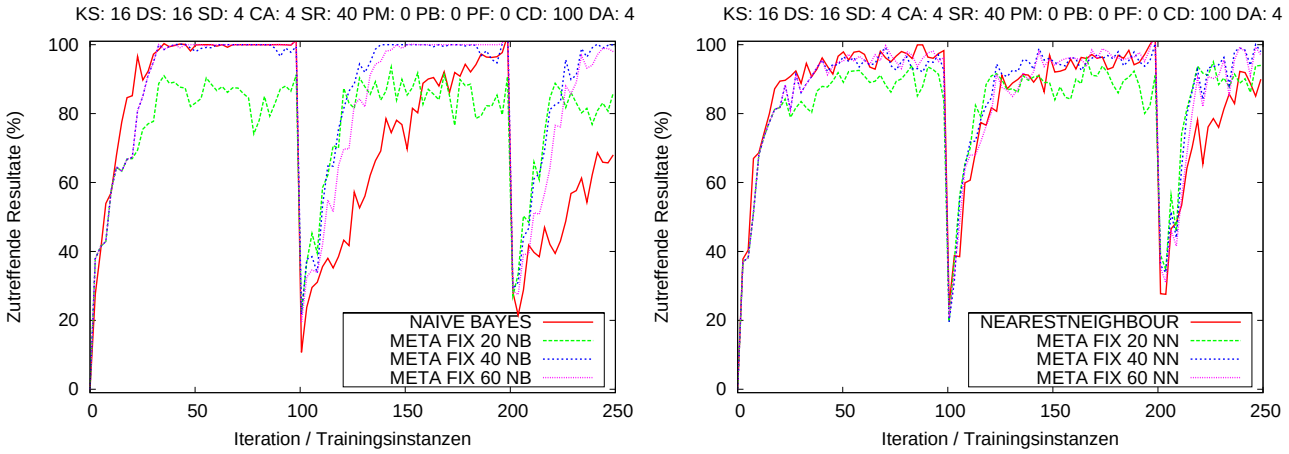
Die Größe des Fensters ist von zentraler Bedeutung. Eine Fenstergröße von 20 Instanzen führt in dem ausgewählten Szenario stellenweise zu einer vergleichsweise schlechteren Genauigkeit, während eine Fenstergröße von 40 bis 60 Instanzen im Allgemeinen zu sehr guten Resultaten führt. Dieses Verhalten ist besonders deutlich bei NAIVE BAYES zu erkennen.

Die benötigte Fenstergröße ist im Wesentlichen von der Komplexität des Konzeptes und der Aussagekraft einzelner Instanzen abhängig. Diese Parameter sind im Voraus meist nicht bekannt. Daher ist die Festlegung einer geeigneten Fenstergröße ohne konkretes Domänenwissen nur schwer möglich.

Variable Fenstergröße

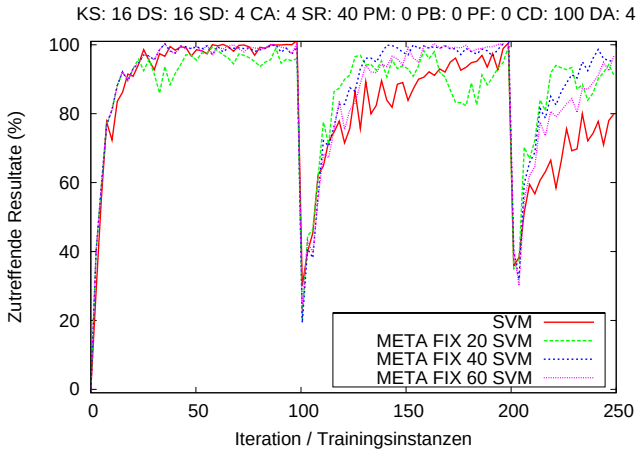
In der folgenden Abbildung 5.31 wird der im Rahmen dieser Arbeit weiterentwickelte und implementierte Ansatz zur dynamischen Anpassung der Fenstergröße (WAH) betrachtet.

Zum einen kann die Fenstergröße ohne konkretes Domänenwissen dynamisch bestimmt werden. Zum anderen kann die Fenstergröße im Falle einer Konzeptänderung reduziert werden, um den Anstieg der Lernkurve in den darauf folgenden Schritten zu optimieren. Der letzte Aspekt wird beispielsweise im Zusammenspiel mit NAIVE



(a) NAIVE BAYES

(b) NEARESTNEIGHBOUR



(c) SVM

Abbildung 5.30: Vergleich verschiedener, statisch festgelegter Fenstergrößen

BAYES deutlich. In Abbildung 5.31 a) ist der Anstieg der Lernkurve nach einer Konzeptänderung deutlich schneller als bei dem statischen Ansatz aus Abbildung 5.30 a).

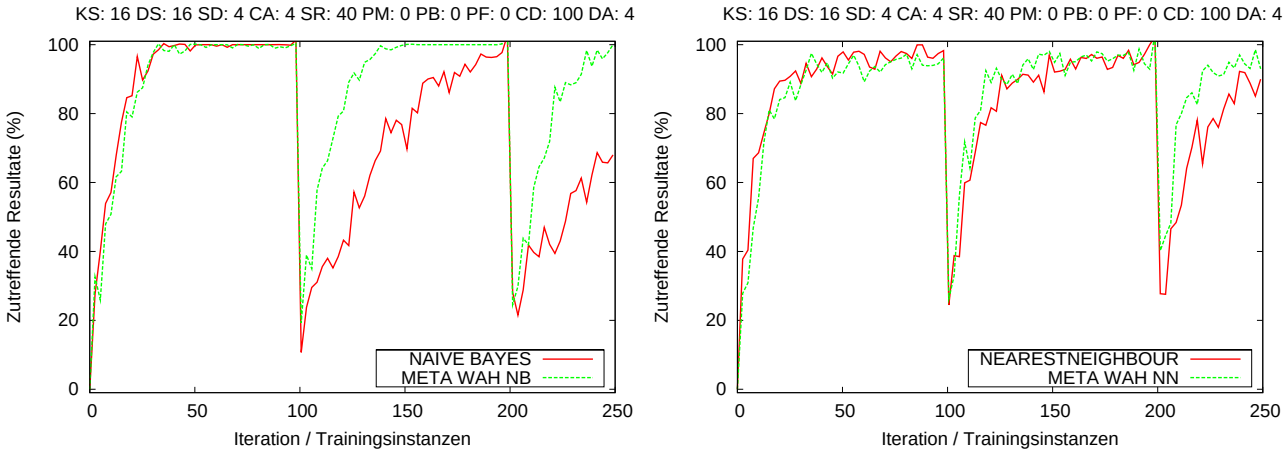
Beobachtete Größen und Eigenschaften der Fensterverwaltung

Die WAH nutzt als Grundlage für die Modifikation der Fenstergröße die aktuelle Genauigkeit des Modells $accWeighted_t$, die Shannon-Entropie $shannon_t$, sowie die Standardabweichung $stdErr_t$. Wie sich diese Werte im Laufe der Trainingsphase verhalten, ist in Abbildung 5.32 dargestellt.

Die Genauigkeit wird in jedem Iterationsschritt über die Funktion $accWeighted$ bestimmt. Ein Wert von 1.0 würde einem Wert von 100% für die Trefferquote entsprechen, wobei hierfür nur die vorangegangenen 25 Trainingsinstanzen betrachtet werden. Gut zu erkennen, sind jeweils die Einbrüche in der Genauigkeit beim Auftreten von Konzeptänderungen.

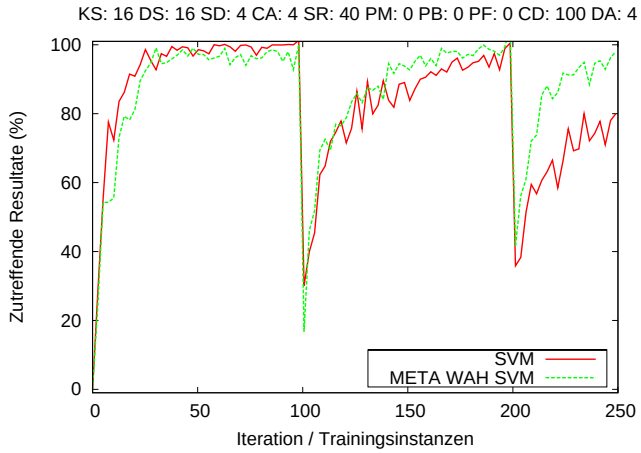
Die Shannon-Entropie verhält sich fast entgegengesetzt zur Genauigkeit. Durch das Heranziehen der Shannon-Entropie, ist es möglich eine übermäßige Fensterreduktion zu verhindern. Sofern das Modell instabil ist, wird auf eine Fensterreduktion verzichtet. Statt einer weiteren Fensterreduktion wird hierdurch eine Lernphase, beziehungsweise eine Vergrößerung des Fensters durchgeführt.

Die Standardabweichung wird herangezogen, um Schwankungen in der Genauigkeit auszugleichen und so irrtümliche Behandlungen von Konzeptänderungen zu vermeiden.



(a) NAIVE BAYES

(b) NEARESTNEIGHBOUR



(c) SVM

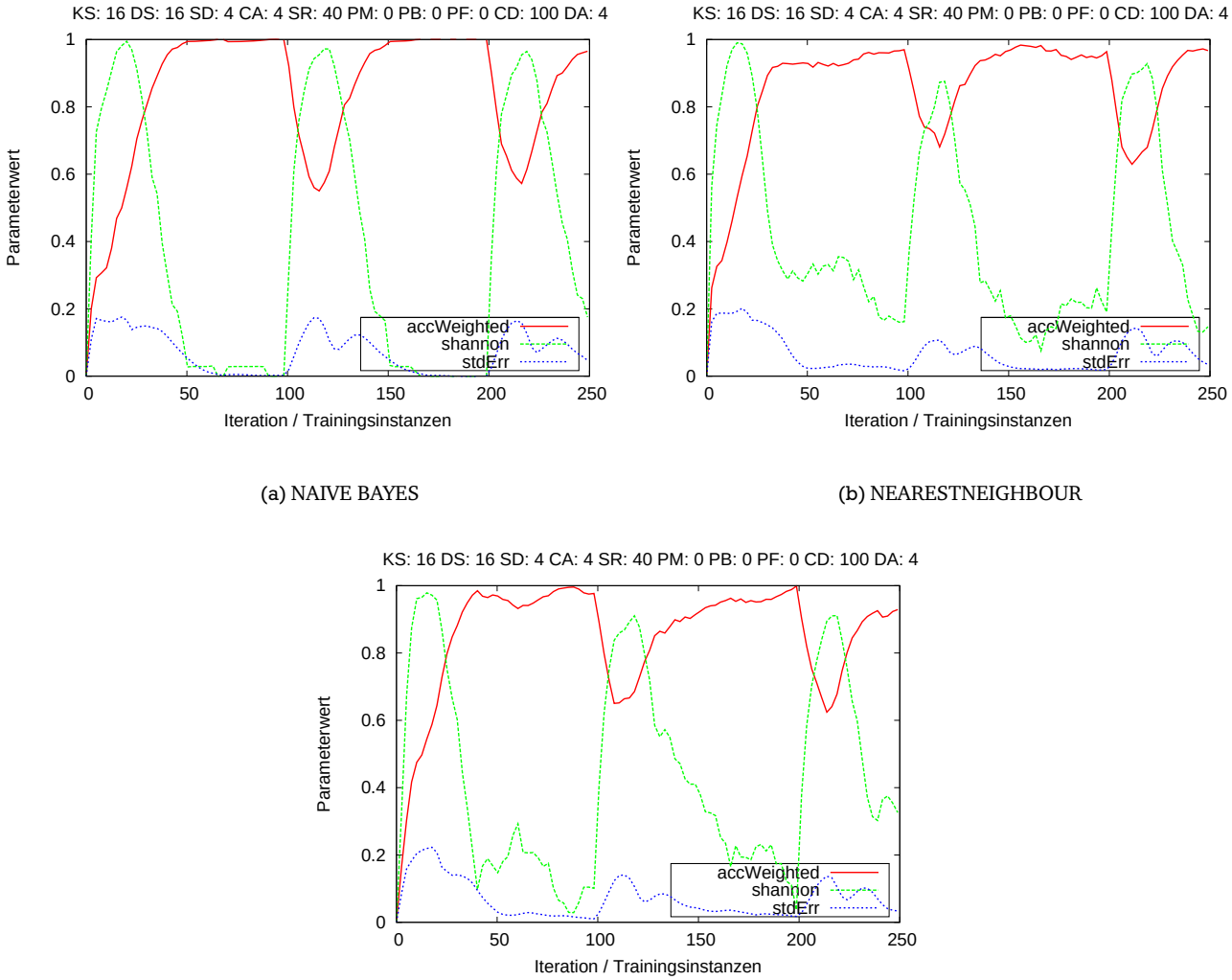
Abbildung 5.31: Vergleich Klassifikationsgenauigkeit mit WAH

Im Vergleich zwischen den verschiedenen Verfahren lässt sich gut ein Unterschied in den betrachteten Werten erkennen. Im Allgemeinen wird festgestellt, dass die Schwankungen deutlicher ausfallen, je ungenauer die Resultate der jeweiligen Verfahren sind. Je höher die Standardabweichung ausfällt, desto größer ist Toleranz der WAH gegenüber einzelnen fehlerhaften Entscheidungen. Langsam steigende Lernkurven führen zu größeren Fenstern. Ein starker Einbruch der Genauigkeit, führt zu einer drastischeren Reduktion der Fenstergröße, während eine höhere Shannon-Entropie weitere Reduktionen verhindert.

Werden diese Werte auf den Ansatz der WAH, welches in Abbildung 5.29 skizziert wurde und den Parametern aus Abschnitt 5.5.4 angewendet, so lässt sich das Verhalten der Fenstergröße in 5.33 a) nachvollziehen.

Wird der Aufwand zur Generierung in Abbildung 5.33 b) betrachtet, so wird gerade bei NEARESTNEIGHBOUR und NAIVE BAYES eine leichte Steigerung des Aufwandes deutlich. Dieser Aufwand, der sich auf ungefähr 10*ms* beläuft ist über den Lernzeitraum hinweg relativ gleichbleibend.

Inkrementelle Lernverfahren unterstützen im Allgemeinen nur das Hinzufügen neuer Instanzen, nicht aber das Entfernen. In Fällen in denen die Fenstergröße beibehalten oder reduziert werden soll und somit mindestens ein Element aus dem Modell entfernt werden muss, ist die Erstellung eines neuen Modells erforderlich, wodurch ein nicht unerheblicher Zusatzaufwand entsteht. Dieser Zusatzaufwand bleibt jedoch bei einer Limitierung der Trainingsinstanzen und über den gesamten Lernverlauf hinweg betrachtet im Durchschnitt innerhalb dieser 10ms. Zur



(a) NAIVE BAYES

(b) NEARESTNEIGHBOUR

(c) SVM

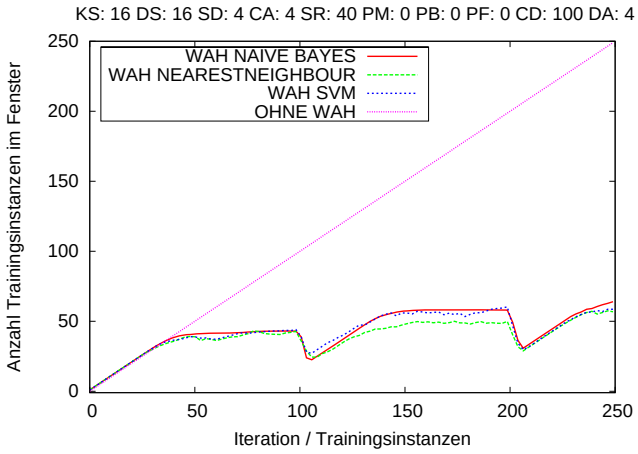
Abbildung 5.32: Parameter der WAH

Bestimmung der aktuellen Genauigkeit wird bei jeder Generierung des Modells auch eine Auswertung angestoßen, die einen weiteren Zusatzaufwand erzeugt.

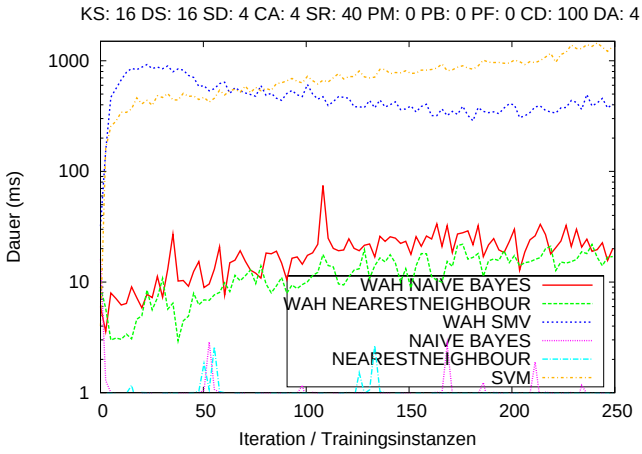
Bei SVM wird jedoch auch der Nutzen der WAH hinsichtlich des Aufwandes zur Generierung deutlich. Durch die Limitierung der Trainingsinstanzen wird der Aufwand zur Modellgenerierung unabhängig von der Anzahl der global zur Verfügung stehenden Trainingsinstanzen.

In der Abbildung 5.33 c) wird der Speicherbedarf für das jeweilige Modell betrachtet. Bei NEARESTNEIGHBOUR und SVM wird ebenfalls eine deutliche Reduktion des Speicherbedarfs durch die WAH deutlich. Dagegen ist die Größe des Modells von NAIVE BAYES unabhängig von der Anzahl der Instanzen, da in einem solchen Modell nur die Auftretenshäufigkeit von Sensorwerten im Bezug auf Kontextklassen vorgehalten wird.

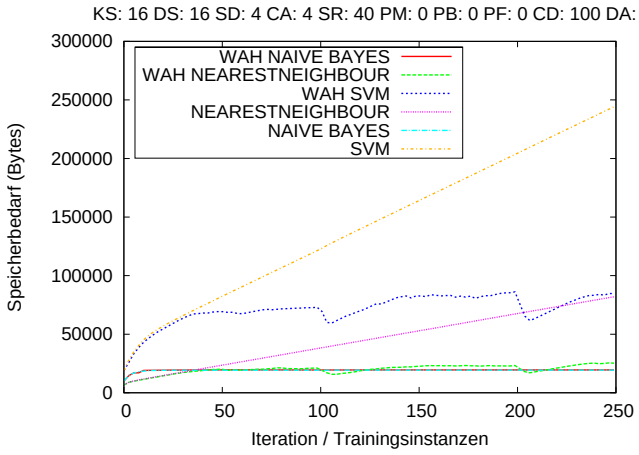
Bei Offline-Verfahren wird zusätzlich zu dem Speicherbedarf für die Modelle auch weiterer Speicherplatz zum Vorhalten der Instanzen benötigt, da bei jeder Iteration alle Trainingsdaten benötigt werden. Wird an dieser Stelle das beschriebene Meta-Verfahren angewendet, so können einerseits alle Instanzen außerhalb dieses Fensters entfernt werden. Andererseits wird bei den inkrementellen Ansätzen in Kombination mit dem beschriebenen Meta-Verfahren ebenso eine Speicherung aller Trainingsinstanzen im Fenster notwendig. Da inkrementelle Verfahren nur das Hinzufügen neuer Instanzen erlauben, jedoch nicht das Entfernen, wird bei dem Entfernen eines Elements eine komplette Neugenerierung des Modells notwendig.



(a) Fenstergröße



(b) Aufwand Generierung



(c) Speicherbedarf

Abbildung 5.33: Eigenschaften der Fensterverwaltung

5.5.6 Fazit

Wie in den Abbildungen zu sehen ist, liefern die betrachteten Verfahren (SVM, NAIVE BAYES, NEARESTNEIGHBOUR) anfangs eine ähnliche Trefferquote, sowohl mit oder ohne Fensterverfahren. Nach dem Auftreten einer Konzeptänderung gehen diese Werte jedoch auseinander. Je später eine Konzeptänderung auftritt, desto größer ist der Unterschied in der Qualität, welche in der Kombination mit einem Fensterverfahren erzielt werden kann. Ohne konkretes Vorwissen ist die Bestimmung der Fenstergröße nur schwer möglich. Durch den vorgestellten Ansatz der WAH ist es möglich die Fenstergröße dynamisch an das Szenario, die Komplexität des Konzepts und an das Auftreten von Konzeptänderungen anzupassen. An dieser Stelle ist zu erwähnen, dass weitere Resultate für J48, NB TREE und NN GENERALIZED in Kombination mit WAH ebenfalls deutlich besser ausgefallen sind als ohne.

Die Fenstergröße beziehungsweise die Größe der Trainingsmenge ist ebenfalls ein wichtiges Merkmal. Die Anzahl von Trainingsinstanzen bei Ansätzen ohne limitierenden Faktor steigt linear. Bei Anwendungsszenarien, bei denen regelmäßig mit Feedback zu rechnen ist, steigt der Speicherbedarf zur Speicherung der Traininginstanzen und je nach Verfahren auch der des Modells stetig an. Bei dem Einsatz von Batch-Learnern steigt der Aufwand nach jedem Iterationsschritt ebenfalls stetig an.

Die Limitierung der Trainingsinstanzen verbessert die Anwendbarkeit von Batch-Verfahren und Lazy-Learnern, indem das Problem der Skalierbarkeit in der Anzahl der Instanzen behandelt wird.

Fensterverfahren ermöglichen es Offline-Lernverfahren auf Effekte einzugehen, welche durch die iterative Herangehensweise und zeitbehaftete Trainingsinstanzen verursacht werden.

Allerdings wird bei einem Fensterverfahren ein bereits bestehendes Lernverfahren gekapselt, wodurch das Fensterverfahren nur bedingt auf das Verhalten des Basisklassifikators eingreifen kann. Online-Lernverfahren sind eigenständige Lernverfahren, welche direkt auf die Verarbeitung von Strömen von Instanzen konzipiert wurden. Somit behandeln diese Verfahren implizit die Herausforderungen, welche durch zeitliche Änderungen innerhalb des zu erlernenden Konzeptes entstehen können.

Es existiert eine Reihe von Ansätzen in der Literatur, welche im Folgenden aufgezeigt werden. Nur für wenige dieser Ansätze sind Implementierungen verfügbar und viele dieser Ansätze lassen sich nur eingeschränkt in den angestrebten Anwendungsszenarien nutzen (viele unterstützen lediglich eine binäre Klassifikation oder sind beschränkt auf diskrete oder kontinuierliche Attribute). Das WEKA-Projekt beinhaltet seit kurzem die Erweiterung *Massive Online Analysis* (MOA - `http://www.cs.waikato.ac.nz/~abifet/MOA/`), welche den bisher auf Offline-Verfahren fokussierten Ansatz erweitert und um Aspekte aus dem Bereich Online-Learning (wie beispielsweise die iterative Erstellung und Analyse von Modellen) erweitert. Allerdings sind hier ebenfalls noch keine nativen Online-Verfahren enthalten. Um einen Vergleich mit Online-Lernverfahren zu ermöglichen, ist die Auswahl eines geeigneten Ansatzes für ein Online-Lernverfahren, dessen Erweiterung zum Einsatz in dem angestrebten Anwendungsszenario, sowie dessen Umsetzung erforderlich. Die Arbeiten hierzu werden ebenfalls in diesem Abschnitt dargestellt.

5.6.1 Verwandte Arbeiten

Es existiert eine Reihe von Online-Lernverfahren, welche unterschiedliche Ansätze verfolgen. Zum besseren Vergleich der Ansätze wird zunächst eine Kategorisierung der betrachteten Ansätze durchgeführt.

Um zeitliche Aspekte zu berücksichtigen, basieren Online-Learner auf einer Art *unvollständiger* Datenhaltung. Der Instanzenspeicher ist vergleichbar mit dem Ansatz aus Abschnitt 5.5.1 der Fensterverfahren. Er beschreibt die Eigenschaft des Lernverfahrens, Instanzen zu speichern und für zukünftige Iterationen heranzuziehen. Eine weitere Kategorisierung von Online-Lernverfahren kann anhand des Merkmals *Ablauf der Modellgenerierung* (siehe auch Abschnitt 5.3.2) durchgeführt werden. Die Kategorisierung der betrachteten Verfahren ist in der Tabelle 5.7 dargestellt.

Verfahren	Instanzenspeicher	Modellgenerierung
IB2/IB3	partiell	inkrementell
FAVORIT	partiell	inkrementell
AQ-PM	partiell	batch
STAGGER	-	inkrementell
WINNOW	-	inkrementell
FLORA 1-4	partiell	inkrementell
LAIR, HILLARY	partiell	inkrementell

Tabelle 5.7: Kategorisierung von Online-Lernverfahren. In Anlehnung an [Mal04]

IB2/IB3

Das Lernverfahren *Instance Based 2* (IB2, nicht zu verwechseln mit IBk-2) entspricht dem Ansatz von NEAREST-NEIGHBOUR, jedoch mit dem Unterschied, dass nur Instanzen gespeichert werden, deren Klassifikation anhand des bisherigen Datensatzes nicht korrekt war. So tendiert IB2 dazu, nur Trainingsinstanzen auf den Klassengrenzen zu speichern und mit diesen ein Klassifikationsmodell aufzubauen [AAK91]. Im Falle einer Konzeptänderung verbleiben jedoch auch *veraltete* Instanzen, weshalb auch hier im Laufe der Zeit Verzögerungen in der Adaption von Konzeptänderungen zu erwarten sind. IB3 verwendet zusätzlich einen Zähler pro Instanz, welcher aussagt, wie häufig eine Instanz an einer richtigen oder falschen Bewertung beteiligt war. Dies ermöglicht Instanzen, welche durch fehlerhaften Sensorwerten oder Konzeptänderungen entstanden sind zu identifizieren und zu entfernen.

FAVORIT

FAVORIT [KK92] erweitert den Ansatz eines Entscheidungsbaums, um eine zusätzliche Gewichtung der Instanzen, welche im Laufe der Zeit sinkt. Fällt diese Größe unter einen vom Nutzer definierten Grenzwert, so wird die Instanz aus dem Konzept entfernt. Erlernt das System jedoch ein bereits enthaltenes Beispiel nochmals, so wird dessen Gewichtung inkrementiert.

AQ-PM

Der Lernalgorithmus AQ-PM [MM00b] ist ein *temporaler Batch-Learner*. Dieses Modell besteht aus Entscheidungs-Regeln, welche die Instanzen, welche im Instanzenspeicher enthalten sind überdecken. Ändert sich die Menge an Instanzen im Instanzenspeicher, so wird das Modell neu aufgebaut. Eine zusätzliche Erweiterung gegenüber FAVORIT fand in der Gewichtungsfunktion statt, welche auch die Gewichte von positiven Instanzen aus der Nachbarschaft der Trainingsinstanz erhöht.

STAGGER

STAGGER [SG86] besitzt keine Möglichkeit Instanzen zu speichern. Stattdessen nutzt STAGGER *Beschreibungselemente*, um einzelne Aussagen im Konzept zu speichern. Diese Aussagen können verknüpft werden und bilden somit Regeln. Entscheidungen über den Regelaufbau oder Regelveränderungen werden auf Basis statistischer Größen realisiert. Hierzu werden Zähler für die Regeln gehalten, welche je nach zutreffenden Trainingsinstanzen verändert werden.

WINNOW

Bei WINNOW [Lit88] erhält jeder vorkommende Attributwert in den Trainingsinstanzen eine Gewichtung entsprechend seiner relativen Häufigkeit im System. WINNOW führt eine binäre Entscheidung anhand einer Hyperebene im Raum durch (vergleichbar zu SVM). Eine hinzuzufügende Trainingsinstanz wird gegen das aktuelle Modell getestet. Entspricht das Ergebnis nicht dem der Trainingsinstanz, so werden alle Gewichte, welche bei der Entscheidung herangezogen wurden, modifiziert (vergleichbar mit der Herangehensweise von NEURALNET).

FLORA

FLORA [KW96] nutzt ein Fensterverfahren zur partiellen Speicherung von Trainingsinstanzen. Auf der Basis von Instanzen innerhalb dieses Fensters werden Regeln generiert. Diese Menge Regeln werden durch Funktionen zur Generalisierung auf wenige Aussagekräftige Regeln reduziert. Die bisher existierenden FLORA-Verfahren erlauben eine binäre Klassifikation auf der Basis von diskreten Werten. Tritt eine Konzeptänderung ein, so werden diejenigen Regeln, welche auf eine Trainingsinstanz zutreffen, jedoch nicht der Kontextklasse der Trainingsinstanz entsprechen, modifiziert und zu der Kontextklasse der Trainingsinstanz zugeordnet.

LAIR, HILLARY

LAIR [Mal04, EW91] ist ein Lernsystem mit einem partiellen Speicher für Instanzen. Jedoch ist die Speicherausprägung minimal, da LAIR nur das erste positive Beispiel ablegt. Basierend auf neu hinzukommenden Trainingsinstanzen wird dieses erste Beispiel erweitert und dient als Grundlage zur Klassifikation. Das Lernsystem HILLARY hingegen nutzt einen größeren Instanzenspeicher, in dem es nur negative Trainingsinstanzen speichert. Positive Instanzen werden nicht beibehalten, sondern nur in das Konzept integriert. Überdeckt sich im weiteren Lernverlauf das ausschließlich durch positive Beispiele entstandene Konzept mit negativen Instanzen, so werden diese wieder aus dem Speicher entfernt [IWL88].

5.6.2 FLORA - Grundlage und Entwicklungsstand

Als Grundlage für die Umsetzung eines geeigneten Online-Lernverfahrens wird das FLOating Rough Approximation-Verfahren herangezogen [WK96]. Im Bereich der Online-Lernverfahren ist FLORA eines der am weitesten fortgeschrittenen Verfahren und wird in vielen Arbeiten als Referenzverfahren herangezogen. Von FLORA wurden bisher vier Versionen (FLORA 1-4) vorgestellt.

Basiskonzept - FLORA 1

Das Basiskonzept basiert auf logischen Regeln, welche dynamisch zwischen Regelmengen unterschiedlicher Aussagen verschoben werden können. Zudem verfügt FLORA über die in Kapitel 5.4.6 beschriebenen Eigenschaften

der Erweiterbarkeit und des Backtracings. Durch das Verschieben einzelner Regeln kann das Verhalten des Modells schnell angepasst werden.

Flexible Fenstergröße - FLORA 2

FLORA 2 erweitert das Konzept um ein Verfahren für eine dynamische Anpassung der Fenstergröße. Dieses Verfahren wurde bereits in Abschnitt 5.5.2 beschrieben. Die Fenstergröße passt sich der Klassifikationsgenauigkeit des Modells an. In stabilen Phasen wird die Fenstergröße konstant gehalten, um ein Over-Training zu vermeiden. In Situationen, in denen die Klassifikationsgenauigkeit deutlich zurückgeht, wird das Fenster verkleinert, um eine schnelle Anpassung an das Zielkonzept zu ermöglichen.

Langzeitgedächtnis - FLORA 3

Ein weiteres Problem von Konzeptänderungen ist, dass sie in der Anfangsphase nur schwer von einem *falschen* Feedback unterschieden werden können. Das *Langzeitgedächtnis* von FLORA 3 kann dieses Problems lösen. Eine neue Aussage des Nutzers wird vom Modell sofort übernommen, das alte Konzept bleibt aber als Backup gespeichert. Sollten nachfolgende Aussagen besser dem alten Konzept entsprechen, kann zu diesem zurückgesprungen werden.

Fehler im Feedback - FLORA 4

FLORA 4 beinhaltet zusätzliche Mechanismen, um Fehler in den Trainingsdaten und fehlerhaftes Feedback zu berücksichtigen. Während FLORA 1-3 sehr schnell auf Änderungen in der Informationsgrundlage reagieren, wurde bei FLORA 4 eine überschnelle Reaktion auf einzelne widersprüchliche Instanzen vermieden. Daher ist FLORA 4 deutlich robuster gegenüber diesen Effekten. Auf der anderen Seite ist jedoch die Reaktionsgeschwindigkeit gegenüber Konzeptänderungen herabgesetzt.

5.6.3 FLORA-MC - Eine Erweiterung der FLORA Familie

Der in der Literatur vorhandene Ansatz von FLORA verfügt nur über die Funktionalität, binär zu klassifizieren und diskrete Werte zu verarbeiten. Im Rahmen dieser Arbeit wurde der Ansatz von FLORA erweitert und als WEKA-kompatibles Lernverfahren mit der Bezeichnung FLOating Rough Approximation-Multivariate Classification (FLORA-MC) entworfen und umgesetzt. FLORA-MC verfügt als Erweiterung sowohl über eine zusätzliche Strategie der dynamischen Fensterverwaltung, als auch über die Möglichkeit, kontinuierliche Wertebereiche zu verarbeiten und Konzepte mit beliebig vielen Zielsituationen abzubilden.

Der Ansatz von FLORA wurde als Grundlage für FLORA-MC übernommen. FLORA arbeitet auf Regeln, welche aus sogenannten *Beschreibungselementen* (*Description Items* - DI) bestehen. Beschreibungselemente bestehen aus einzelnen verketteten, beziehungsweise UND-verknüpften Aussagen. Im Laufe der Lernphase werden DIs anhand der Trainingsinstanzen erstellt. Für jede Kontextklasse k werden drei Mengen zur Speicherung von DIs angelegt:

- **$ADES_k$:** Enthält alle DIs, deren Aussage *für* die Kontextklasse k spricht.
- **$NDES_k$:** Beinhaltet DIs, deren Aussage *gegen* die Kontextklasse k spricht.
- **$PDES_k$:** Enthält DIs, deren Aussage nicht klar den anderen beiden Gruppen zugeordnet werden kann (beispielsweise durch widersprüchliche Aussagen). In das PDES werden keine DI auf direktem Wege hinzugefügt, sondern DI aus dem ADES oder NDES werden gegebenenfalls dorthin verschoben.

In der Abbildung 5.34 wird die Architektur von FLORA-MC gezeigt. Gegenüber der FLORA Familie nutzt FLORA-MC jeweils eine Kombination aus ADES, PDES, NDES je Kontextklasse.

Die wesentlichen Mechanismen von FLORA bestehen aus dem Lernprozess und dem Prozess, Wissen gegebenenfalls wieder zu verwerfen. In der weiteren Diskussion des FLORA-Verfahrens und deren Erweiterung wird die Terminologie von [WK96] weitestgehend übernommen.

Lernprozess

Ein DI wird nur solange in den oben genannten Mengen gehalten, wie die Aussage des DI durch Trainingsinstanzen im Fenster *gestützt* wird. Beispielsweise besteht eine Aussage im ADES nur solange wie sie durch mindestens eine Instanz im Fenster beschrieben wird ($AP_i > 0$). Wird dieser Zähler auf null herabgesetzt ($AP_i = 0$), so wird das

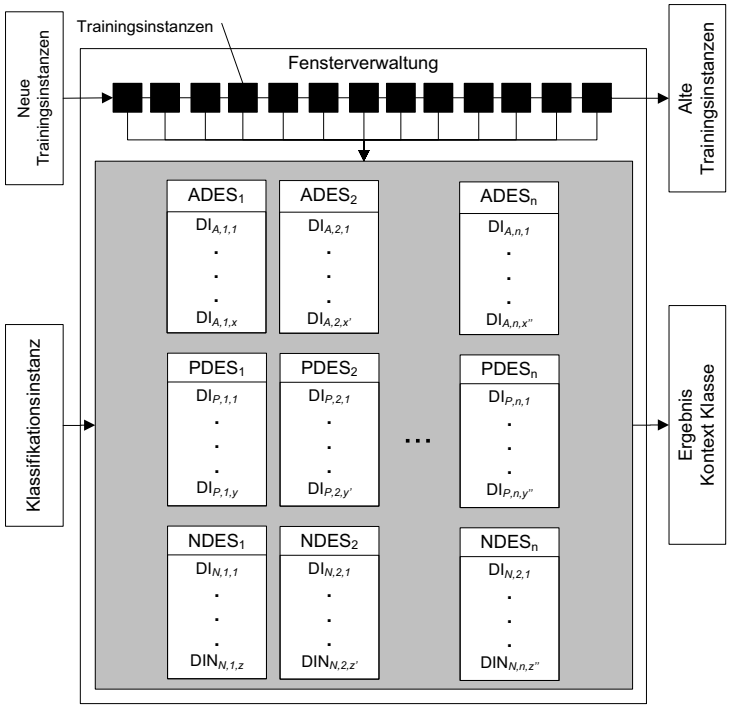


Abbildung 5.34: Architektur von FLORA-MC

entsprechende Beschreibungselement entfernt. Entsprechende Zähler existieren für das PDES (PP_i, PN_i) und das NDES (NN_i).

Der Lernprozess wird durch die Anwendung einer neuen Instanz auf die Funktion *learn()* angestoßen. Die Funktion *learn()* wird in Abbildung 5.35 dargestellt.

Die Funktion *learn(I, j, $ADES_i$, $PDES_i$, NDES)* erhält in Schritt 1 die neue Instanz, den Klassenwert, sowie Zeiger auf ADES, PDES und NDES als Parameter übergeben.

In Schritt 2 wird über die *match* Funktion überprüft, ob aktuell bestehende DIs aus dem ADES-Set bereits die Instanz beschreiben[9]. Wenn dies nicht der Fall ist, wird in Schritt 3 versucht eine Generalisierung mit einem der DIs aus dem ADES durchzuführen (siehe Abschnitt 5.6.3). Ist dies ebenfalls nicht möglich, wird in Schritt 4 die Instanz als ein neues DI in das ADES eingefügt. Die Funktion *include()* erstellt hierzu ein entsprechendes Beschreibungselement. Es setzt die Zähler, welche als zweiter Parameter übergeben werden und speichert das neu erstellte DI in jener Menge, welche als dritter Parameter angegeben wurde.

In den nächsten Schritten werden die Inhalte der PDES und NDES Mengen angepasst. Dies beinhaltet die Erhöhung der Zähler passender DIs aus dem PDES in Schritt 6, sowie das Verschieben passender DIs aus der Menge aller NDES in das jeweils entsprechende PDES. In der *delete()* Funktion wird das übergebene Beschreibungselement aus der Menge entfernt, welche als zweiter Parameter übergeben wurde.

In Schritt 7 wird die Erweiterungsfunktion für alle anderen Klassenwerte durchgeführt, jedoch mit vertauschten Zeigern auf die ADES und NDES Mengen. Dies hat zur Folge, dass Aussagen die *für* einen bestimmen Klassenwert gleichzeitig auch als Aussagen *gegen* alle anderen Klassenwerte genutzt werden.

Verwerfen von Wissen

Aussagen, welche nicht mehr durch Instanzen gestützt werden und folglich veraltet sind, müssen aus der Wissensbasis also den verschiedenen *Description Element Sets* entfernt werden. Die Funktion ***forget()*** (siehe Abbildung 5.36) wird hierzu aufgerufen um die Zähler aller Beschreibungselemente (DI), welche von der entsprechenden Instanz gestützt werden, zu reduzieren.

[9] match: Alle diskreten Werte der Trainingsinstanz stimmen mit jenen des DI überein und alle kontinuierlichen Werte der Trainingsinstanz liegen in den numerischen Intervallen des DI.

Notationen:

$ADES$	Menge aller ADES-Mengen
$PDES$	Menge aller PDES-Mengen
$NDES$	Menge aller NDES-Mengen
I	Neue Trainingsinstanz
AP_j, i	Anzahl von Instanzen welche $DI_{A,j,i}$ stützen
PP_j, i	Anzahl von Instanzen (*für* die jeweilige Kontextklasse j) welche $DI_{P,j,i}$ stützen
PN_j, i	Anzahl von Instanzen (*gegen* die jeweilige Kontextklasse j) welche $DI_{P,j,i}$ stützen
NN_j, i	Anzahl von Instanzen welche $DI_{N,j,i}$ stützen

```
FUNCTION learn(I)
  extend(I,getClass(I),ADES, PDES, NDES)        //Schritt 1: Ermitteln des Klassenwertes j; Starten der Erweiterungsfunktion
  FOR i := 1 TO |NDES| DO                       //Schritt 7: Umgekehrte Anwendung der Erweiterungsfunktion auf die Sets anderer Klassenwerte
    IF i != j THEN
      extend(I,i,NDES, PDES, ADES)

FUNCTION extend(I,j,ADES, PDES, NDES)
  match := false
  FOR i := 1 to |DI_A,j| DO                     //Schritt 2: Abgleich mit ADES-SET
    IF match(I, DI_A,i) THEN
      AP_j,i := AP_j,i + 1
      match := true

  IF NOT match THEN                             //Schritt 3: Generalisieren mit ADES-SET
    generalized := generalize(I, ADES_j, NDES_j)

  IF NOT match AND NOT generalized THEN         //Schritt 4: Hinzufügen zu ADES-SET
    include(I, AP_j,i = 1, ADES_j)

  FOR i := 1 TO |PDES_j| DO                     //Schritt 5: Erweitern der Zähler im PDES-SET
    IF match(I, DI_P,j,i) THEN
      PP_j,i := PP_j,i + 1

  FOR i := 1 TO |NDES_j| DO                     //Schritt 6: Verschieben von Aussagen aus NDES-SETS
    IF match(I, DI_N,j,i) THEN
      delete(DI_N,j,i, NDES_j)
      include(DI_N,j,i, {PN_i = NN_i, PP_i = 1}, PDES_j)
```

Abbildung 5.35: FLORA-MC Funktion: *learn()*

Die Vorgehensweise der Funktion *forget()* ist dabei ähnlich zu der zuvor beschriebenen Funktion *learn()*. In der Funktion *reduceADES* werden die Inhalte aller Elemente aus dem ADES überprüft und behandelt. Wird ein DI aus dem ADES von keiner Instanz gestützt, beziehungsweise geht der Zähler auf null, wird das DI entfernt. Im nächsten Schritt werden die Zähler der DI aus dem PDES angepasst. Bestehen nach dem Entfernen einer Instanz keine widersprüchlichen Aussagen mehr, so kann ein Beschreibungselement in die NDES Menge verschoben werden.

Danach wird die Menge aller anderen NDES verarbeitet, wobei die Funktion *reduceNDES* analog zur Funktion *reduceADES* vorgeht, mit dem Unterschied, dass Elemente aus dem PDES, welche keine Widersprüche mehr aufweisen, anschließend in das jeweilige ADES verschoben werden.

Beispielhafte Darstellung eines Ablaufs

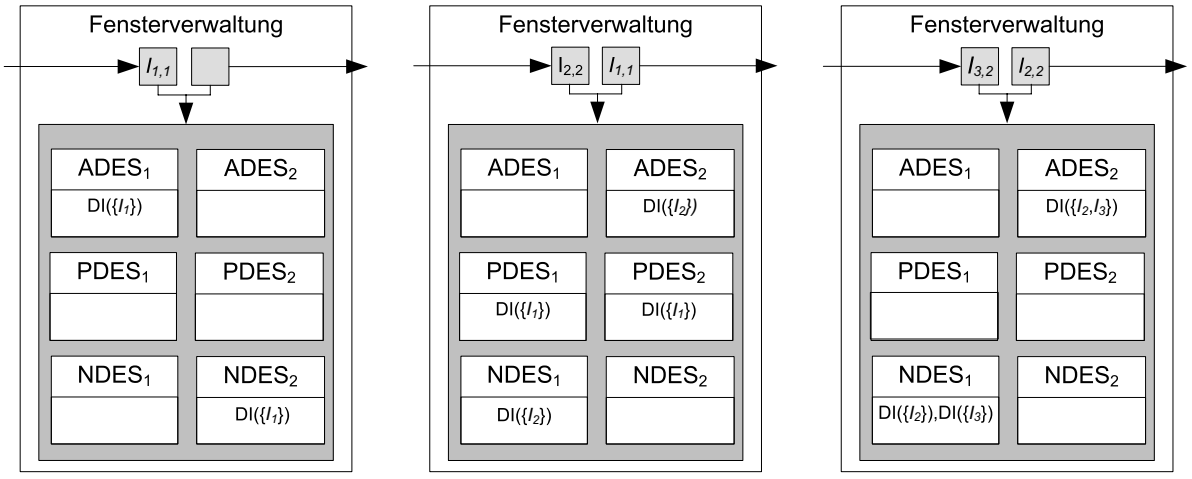
In Abbildung 5.37 wird ein vereinfachter Ablauf der Lernfunktion dargestellt. Dabei wird angenommen, dass die Fensterverwaltung eine feste Fenstergröße von zwei Elementen vorsieht.

```
FUNCTION forget(I)
  j = getClass(I)
    reduceADES(I,j,NDES, PDES, ADES)
  FOR i := 1 TO |NDES| DO
    IF i != j
      reduceNDES(I,i,NDES, PDES, ADES)
```

```
FUNCTION reduceADES(I,j,ADES, PDES, NDES)
  FOR i := 1 TO |ADES_j| DO
    IF match(I, DI_{A,j,i}) THEN
      AP_{j,i} := AP_{j,i} − 1
      IF AP_{j,i} = 0 THEN
        delete(DI_{A,j,i}, ADES_j)
  FOR i := 1 TO |PDES_j| DO
    IF match(I, DI_{P,j,i}) THEN
      PP_{j,i} := PP_{j,i} − 1
      IF PP_{j,i} = 0 THEN
        delete(DI_{P,j,i}, PDES_j)
        include(DI_{P,j,i}, NN_{j,i} = PN_{j,i} , NDES_j)
```

```
FUNCTION reduceNDES(I,j,ADES, PDES, NDES)
  FOR i := 1 TO |NDES_j| DO
    IF match(I, DI_{N,j,i}) THEN
      NN_{j,i} := NN_{j,i} − 1
      IF NN_{j,i} = 0 THEN
        delete(DI_{A,j,i}, ADES_j)
  FOR i := 1 TO |PDES_j| DO
    IF match(I, DI_{P,j,i}) THEN
      PN_{j,i} := PN_{j,i} − 1
      IF PN_{j,i} = 0 THEN
        delete(DI_{P,j,i}, PDES_j)
        include(DI_{P,j,i}, AP_{j,i} = PP_{j,i} , ADES_j)
```

Abbildung 5.36: FLORA-MC Funktion: *forget()*



(a) 1. Iteration (b) 2. Iteration (c) 3. Iteration

Abbildung 5.37: Beispiel: Ablauf des Lernverfahrens

In jeder Iteration wird eine neue Instanz $I_{Nr,Klassenwert}$ hinzugefügt. In der ersten Iteration wird durch die Lernfunktion ein neues Beschreibungselement (DI) erstellt[10]. Dieses DI wird in das ADES des Klassenwertes – also $ADES_1$ – sowie in das NDES aller anderen Klassenwerte hinzugefügt. In diesem vereinfachten Fall existiert nur ein weiterer Klassenwert – entsprechend wird das DI in $NDES_2$ eingetragen.

In der zweiten Iteration wird eine weitere Instanz $I_{2,2}$ hinzugefügt. Angenommen die Aussage dieser neuen Instanz deckt sich mit der der vorangegangenen Instanz $I_{1,1}$ (match($I_{2,2}$,$DI(\{I_1\})$) = true). Dann wird das $DI(\{I_1\})$ vom $ADES_1$ in das $PDES_1$, sowie vom $ADES_2$ in das $PDES_2$ verschoben.

In der dritten Iteration wird die Instanz $I_{1,1}$ von der Instanz $I_{3,2}$ aus dem Fenster verdrängt. Durch die Funktion *forget()* werden die von $I_{1,1}$ gestützten Beschreibungselemente überprüft. Da die Beschreibungselemente $DI(\{I_1\})$ ausschließlich von $I_{1,1}$ gestützt wurden, werden diese nach dem Entfernen der Instanz ebenfalls entfernt. Da nun zu $ADES_2$ ein weiteres Beschreibungselement hinzugefügt wird, kann nun eine Generalisierung beider Instanzen durchgeführt werden. Ist diese Generalisierung möglich, so entsteht ein einzelnes neues Beschreibungselement, welches von beiden Instanzen I_2 und I_3 gestützt wird.

[10] Zu jedem DI werden jeweils die Menge der Instanzen, welche die Aussage des DI stützen, angegeben.

Generalisieren

Unter einer Generalisierung versteht man eine Art der Zusammenfassung oder Verallgemeinerung von Regeln, so dass die neu entstandene Regel bzw. das generalisierte Beschreibungselement mit mindestens genau so vielen Trainingsinstanzen wie zuvor übereinstimmt.

Regelreduzierung

In FLORA wird hierfür die *Dropping Condition Rule* von Michalski [Mic83] angewandt. Diese Generalisierungsfunktion entfernt mindestens einen Attributwert aus der Konjunktion des Beschreibungselements, so dass es mit der zu lernenden positiven Instanz übereinstimmt. Allerdings wurde die Durchführung der Generalisierung in FLORA 1-4 nur teilweise beschrieben. Die *Dropping Condition Rule* reduziert die Menge der Regeln im System, so dass nur noch die geringste mögliche Menge an Regeln gehalten wird. Eine Reduktion der Regeln kann zu einer Übergeneralisierung führen. Dies ist gerade in den frühen Phasen des Trainings der Fall, wenn die jeweiligen Beschreibungssets noch gering besetzt sind und dadurch eine Übergeneralisierung des jeweils anderen Sets nicht verhindert werden kann. Ein übergeneralisiertes Beschreibungselement wird beim Auftreten von Gegenbeispielen möglicherweise schnell wieder verworfen, womit die Aussagen der ursprünglichen Beschreibungselemente ebenfalls verloren gehen.

Konsistenzprüfung

Beim Generalisieren eines Beschreibungselements muss auf die Konsistenz der Aussagen zwischen dem NDES und dem ADES geachtet werden, um so die Widerspruchsfreiheit zwischen den beiden Sets zu wahren. Diese Überprüfung ist nach jedem Anwenden der Generalisierungsfunktion durchzuführen [WK96]. Zusammenfassend ist eine solche Funktion der Verallgemeinerung also genau dann erfolgreich, wenn das generalisierte Beschreibungselement des ADES-Sets bzw. des NDES-Sets kein Element des NDES- und PDES-Sets bzw. ADES- und PDES-Sets überdeckt.

Beispielhafter Ablauf der Generalisierung

Das folgende Beispiel verdeutlicht die Generalisierungsfunktion mittels der *Dropping Condition Rule* in Anlehnung an das STAGGER-Konzeptes [SG86] bei dem Übergang von der zweiten zur dritten Phase. Bei dem STAGGER-Konzept handelt sich um ein einfaches Konzept zum Vergleich von Online-Lernverfahren, bei dem den drei Attributen *Farbe* (rot, blau, grün), *Form* (dreieckig, rund, viereckig), *Größe* (klein, mittel, groß) der Wert *Positiv* oder *Negativ* zugeordnet werden soll. Im Zeitverlauf des Lernens, werden diese Konzepte über drei Phasen hinweg modifiziert. Das STAGGER-Konzept sieht folgende Konzepte für die einzelnen Phasen vor:

1. Phase: $Größe = klein \wedge Farbe = rot := Positiv$.
2. Phase: $Farbe = grün \vee Form = rund := Positiv$.
3. Phase: $Größe = (mittel \vee groß) := Positiv$.

Schritt	Element	Inhalt
1	ADES	$Farbe = grün \wedge Form = rund \wedge Größe = mittel$
	NDES	$Farbe = rot \wedge Form = viereckig \wedge Größe = mittel$
2	Instanz	$Farbe = grün \wedge Form = viereckig \wedge Größe = mittel$
3	ADES	$Farbe = grün \wedge Größe = mittel$
	NDES	$Farbe = rot \wedge Form = viereckig \wedge Größe = mittel$
4	Instanz	$Farbe = rot \wedge Form = dreieckig \wedge Größe = mittel$
5	ADES	$Farbe = grün \wedge Größe = mittel$
		$Farbe = rot \wedge Form = dreieckig \wedge Größe = mittel$
	NDES	$Farbe = rot \wedge Form = viereckig \wedge Größe = mittel$

Tabelle 5.8: Dropping Condition Rule in FLORA

Im ersten Schritt der Tabelle 5.8 wird ein Inhalt vorgegeben, wie er innerhalb der zweiten Phase sein könnte. Wird im nächsten Schritt eine weitere Instanz hinzugefügt, so ist in diesem Falle eine Generalisierung möglich.

Hierzu kann im dritten Schritt das Attribut $Form$ entfernt werden. Im vierten Schritt findet ein Konzeptwechsel in die dritte Phase des STAGGER-Konzeptes statt. Die Instanz $Farbe = rot \wedge Form = dreieckig \wedge Größe = mittel$ soll erlernt werden. Das im Falle einer Generalisierung resultierende Beschreibungselement des ADES-Sets wäre $Größe = mittel$. Dieses steht jedoch im Widerspruch zu dem Element aus dem NDES. Aus diesem Grund wird auf die Generalisierung verzichtet und ein neues Beschreibungselement hinzugefügt.

Auswahl des Beschreibungselements, Distanzbestimmung

Bei jeder Iteration wird lediglich eine Generalisierung initiiert. Das bedeutet, es wird maximal eine (erfolgreiche) Generalisierung der neuen Instanz mit einem weiteren Beschreibungselement durchgeführt. Hierzu ist die Entscheidung zu treffen, welches der Beschreibungselemente zur Generalisierung herangezogen werden soll, um möglichst geringe Änderungen an ihnen vornehmen zu müssen. Hierfür ist eine geeignete Distanzfunktion zwischen dem Beschreibungselement und der Instanz zu definieren.

Die Distanz zwischen einer Instanz und einem Beschreibungselement wird wie folgt ermittelt:

- Jedes diskrete Attribut, welches ungleich ist erhöht die Distanz um eins.
- Bei numerischen Attributen wird der normierte Abstand zwischen beiden Werten auf die Distanz addiert (die Verarbeitung numerischer Attribute wird im nachfolgenden Abschnitt genauer erläutert).
- Fehlende Sensorwerte werden wie diskrete Werte behandelt.

Verarbeitung numerischer Werte

Der ursprüngliche Ansatz von FLORA berücksichtigt nur diskrete Werte. Dieser Ansatz wurde erweitert, um auch numerische Werte verarbeiten zu können. Diese Erweiterung erlaubt die Verwendung von numerischen Intervallen innerhalb von Beschreibungselementen. Bei der Generalisierung müssen diese Intervalle berücksichtigt werden. Einzelne numerische Werte werden bei der Generalisierung zu Intervallen zusammengefasst. Ein Hauptproblem besteht in der Frage, wie gering der *Abstand* der einzelnen Werte zu einander sein darf, um bei der Generalisierung zu einem Intervall verbunden werden zu können. Auch hier besteht das Problem der möglichen Übergeneralisierung.

Das Problem besteht in dem fehlenden Wissen über die Größe des Wertebereichs eines numerischen Wertes. Dieses Wissen kann jedoch durch die Beschreibung über die Eigenschaften eines Sensors erhalten und genutzt werden (siehe Abschnitt 4.4).

Hierzu wurde eine Glättungsfunktion integriert, welche die Angabe eines Parameters $smooth \in \{0, 1\}$ benötigt, welcher abhängig von dem Wertebereich des jeweiligen Sensors gewählt wird. Die Generalisierung eines numerischen Wertes x wird wie folgt durchgeführt:

$$smoothing(smooth, n) = \sqrt{\frac{\ln(smooth)}{-\alpha * n}} \tag{5.13}$$

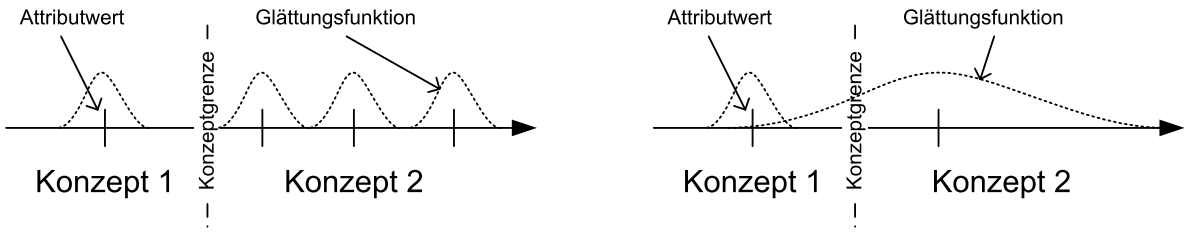
Der Parameter n ist die Anzahl der Instanzen, welche sich bereits durch Generalisierung auf dieses Intervall haben abbilden lassen. Der nutzerdefinierbare Wert α beeinflusst die Intervallbildung. Je niedriger dieser Wert gewählt wird, desto breiter sind die Intervalle, die durch die Glättungsfunktion erzeugt werden. Wie in Abbildung 5.38 dargestellt wurde, verursachen kleine Glättungswerte eine Reduktion der generalisierbaren Beschreibungselemente, wohingegen zu große Glättungswerte zu Übergeneralisierungen führen können. Im Weiteren wurde für $\alpha = 0.5$ gewählt.

Der Glättungswert beschreibt den Bereich um einen einzelnen Wert y, in welchem eine Generalisierung durchgeführt wird ($[y - smoothing(smooth, n), y + smoothing(smooth, n)]$). Wie in Abbildung 5.39 dargestellt, werden bei der Generalisierung mehrere einzelne Intervalle oder Werte zu einem gemeinsamen Intervall $[lb, ub]$ zusammengefasst.

Je mehr Elemente innerhalb eines Intervalls sind, desto größer ist der Parameter n und desto deutlicher ist die Abgrenzung des Intervalls an den Rändern. Dieses Vorgehen basiert auf der Annahme, dass wenn ein Intervall dicht besetzt ist, es unwahrscheinlich ist, dass weit entfernte Werte noch zu diesem Intervall zählen.

$$lb - smoothing(smooth, n) \leq x \leq ub + smoothing(smooth, n) \tag{5.14}$$

Nur wenn sich der numerische Wert x innerhalb dieses Intervalls befindet, kann er durch die Generalisierung mit diesem kombiniert werden.



(a) Zu großer Glättungswert (b) Zu kleiner Glättungswert

Abbildung 5.38: Auswirkung falscher Glättungswerte bei numerischen Attributwerten.

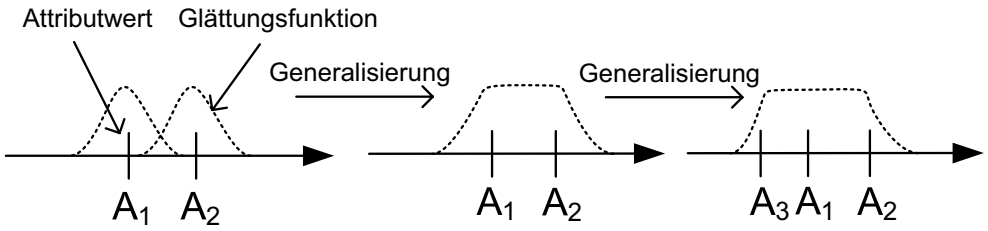


Abbildung 5.39: Generalisierung über numerische Attribute

Klassifikation

Das FLORA-Verfahren sieht zur Klassifikation die Auswertung der Regeln im ADES vor. Wird in FLORA kein passendes Beschreibungselement im ADES vorgefunden, so lautet das binäre Ergebnis von FLORA $false$. FLORA-MC besitzt mehrere ADES, welche Aussagen mit numerischen Intervallen beinhalten können. Zudem müssen auch Instanzen mit fehlenden Sensorwerten verarbeitet werden können.

Die Klassifikation von FLORA-MC sieht nun zwei Schritte zur Klassifikation vor. Im ersten Schritt wird überprüft, ob ein Beschreibungselement in der Menge aller ADES-Sets gefunden werden kann, welches die Klassifikationsinstanz beschreibt ($match(I) = true$). Ist dies der Fall, so wird der Klassenwert des jeweiligen ADES als Resultat zurückgegeben. Wenn kein Beschreibungselement gefunden werden kann, wird in einem zweiten Schritt das Beschreibungselement mit kleinster *Distanz* zur Instanz bestimmt.

Multiple Klassifikation

Je nach Anwendungsfall und Kontextdimension kann ein Kontextobjekt auch gleichzeitig mehrere gültige Kontextklassen haben. Diese Möglichkeit besteht in FLORA-MC. Enthält eine Instanz mehrere Klassenwerte, so kann die Lernfunktion das entsprechende Beschreibungselement in mehreren ADES-Sets gleichzeitig aufnehmen. Alle weiteren Schritte werden analog mit mehreren Beschreibungselementen durchgeführt, mit dem Unterschied, dass bei der Verarbeitung der NDES-Sets diejenigen, deren Klassenwerte in der Instanz enthalten sind ausgelassen werden. Bei der Klassifikation wird dann eine Liste von Kontextklassen als Resultat bestimmt.

Fensterverwaltung

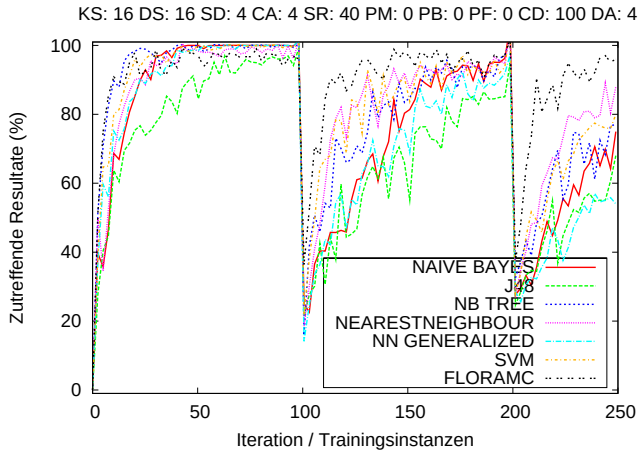
Die *Window Adjustment Heuristic* (WAH) von FLORA wurde bereits in Abschnitt 5.5 adressiert. Es diente als Grundlage für die erweiterte WAH, welche ebenfalls in dem Abschnitt behandelt wurde. Diese erweiterte WAH wird als Grundlage für die Fensterverwaltung in FLORA-MC herangezogen.

Im Gegensatz zur dem Meta-Ansatz ist bei FLORA die Fensterverwaltung direkt im Verfahren verankert. Der Vorteil dieses Ansatzes besteht darin (auf umgekehrtem Wege), von dem Lernverfahren wiederum direkt Einfluss auf die Fensterverwaltung ausüben zu können. FLORA-MC nutzt diesen Vorteil indem eine Fensterreduktion durchgeführt werden kann in welcher nun auch gezielt Elemente entfernt werden können, statt wie bei dem Meta-Ansatz

immer nur die ältesten Elemente aus der Fensterverwaltung zu entfernen. Hierbei können gezielt Instanzen entfernt werden, welche Widersprüche zum aktuellen Konzept aufweisen. Während der Überprüfung der Konsistenz innerhalb der Lernfunktion wird ein Zähler für die Anzahl der Widersprüche eingefügt. Entsteht bei der Integration der neuen Instanz ins ADES-Set ein Widerspruch der Beschreibungselemente zu jenen des NDES, so werden diejenigen Instanzen welche dieses Beschreibungselement des NDES stützen vorrangig entfernt. Somit entfernt das System bei der Erkennung eines Konzeptwechsels zunächst die Instanzen eines Konzeptes, welche im Lernablauf den letzten Widerspruch hervorgerufen haben und somit wahrscheinlich für jenen Wechsel verantwortlich sind. Ist über den Ansatz der gezielten Reduktion von Instanzen keine Auswahl weiterer Instanzen möglich (alle Zähler auf null), so wird nach dem normalen WAH-Ansatz bei den ältesten Instanzen fortgefahren.

5.6.4 Auswertung

Zur Evaluation von FLORA-MC wurde die gleiche Vorgehensweise (siehe Abschnitt 5.4.1) wie bei der Analyse bestehender Verfahren und des erweiterten WAH-Verfahrens angewandt. FLORA-MC wird mit eben jenen Verfahren verglichen.



(a) 4 betroffene Kontextklassen

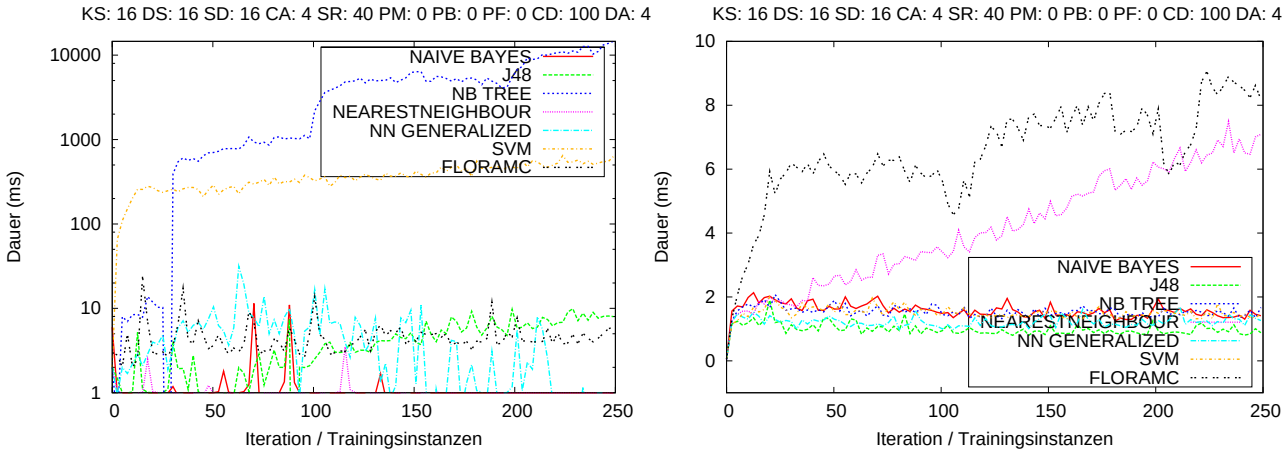
Abbildung 5.40: Klassifikationsgenauigkeit von FLORA-MC und anderen Verfahren unter Konzeptänderung

Bei der Analyse des Basisszenarios mit Konzeptänderungen (siehe Abbildung 5.40) schneidet FLORAMC im Vergleich mit allen anderen betrachteten Verfahren am besten ab. Es bietet sowohl eine schnelle Lernkurve zu Beginn, als auch eine hohe Qualität der Ergebnisse in der ersten Phase. Nach dem Auftreten der Konzeptänderungen weist es die besten Adaptionsraten auf.

In den Abbildungen 5.41 a) und b) wird der Aufwand bei der Anwendung von FLORA-MC mit denen anderer Verfahren verglichen. Die Modellgenerierung generiert vergleichsweise wenig Aufwand. Hierzu genügt es jeweils im ADES-Set der entsprechenden Kontextklasse und den anderen NDES-Sets zu überprüfen, ob eine der Regeln auf die neue Instanz angewendet werden kann. Ist dies der Fall, kann eine Generalisierung versucht werden, anderenfalls wird eine neue Regel hinzugefügt.

Zur Auswertung wird die Klassifikationsinstanz auf die Regeln aller ADES-Sets angewandt. Werden im ersten Schritt keine passenden Regeln gefunden, so wird im zweiten Schritt der Abstand zwischen der Klassifikationsinstanz und den Regeln der ADES und NDES-Sets berechnet. Dabei werden diejenigen Kontextklassen gewählt, bei welchen der minimale Abstand zu den Regeln des ADES-Sets geringer ist, als der minimale Abstand zu denen der NDES-Sets. Dies wird für jede der Kontextklassen durchgeführt. Durch den relativ großen Domänenraum führt der erste Schritt seltener zu passenden Regeln, wodurch der zeitaufwändigere zweite Schritt durchgeführt wird. Diese Vorgehensweise führt, wie in Abbildung 5.41 b) dargestellt, zu etwas größerem Aufwand bei der Modellauswertung. In dem unserem Beitrag [SHRS08] wird das Stagger-Konzept modifiziert und zum Vergleich mit anderen Verfahren herangezogen. Das Stagger-Konzept nutzt einen kleineren Domänenraum, worin FLORA-MC im Vergleich deutlich besser abschneidet. Durch weitgreifende Generalisierung der Regeln lassen sich Regeln aufbau-

en, welche größere Abschnitte des Domänenraumes beschreiben. Tests hierzu haben gezeigt, dass dies zu einer geringeren Anzahl von Regeln führte, welche häufiger im ersten Schritt zur Auswertung genutzt werden konnten und somit zu deutlich höheren Geschwindigkeiten bei der Auswertung führten. Im Gegenzug reduzierte sich jedoch deutlich die Genauigkeit des Modells und die Empfindlichkeit gegenüber Fehlern in den Trainingsdaten verstärkte sich.



(a) Modellgenerierung

(b) Modellauswertung (100 Durchläufe)

Abbildung 5.41: Eigenschaften der Modelle von FLORA-MC und anderen Verfahren

Das FLORA-MC-Verfahren wurde dazu entworfen schnell auf Konzeptänderungen zu reagieren. Wird eine Trainingsinstanz mit fehlerhaften Sensorwerten oder fehlerhaftem Feedback hinzugefügt, so kann dies zum sofortigen Verwerfen von Regeln und somit zum Verlust von Wissen und Genauigkeit führen. Wie die Abbildung 5.42 a) zeigt, kann dies im geringen Umfang noch ausgeglichen werden.

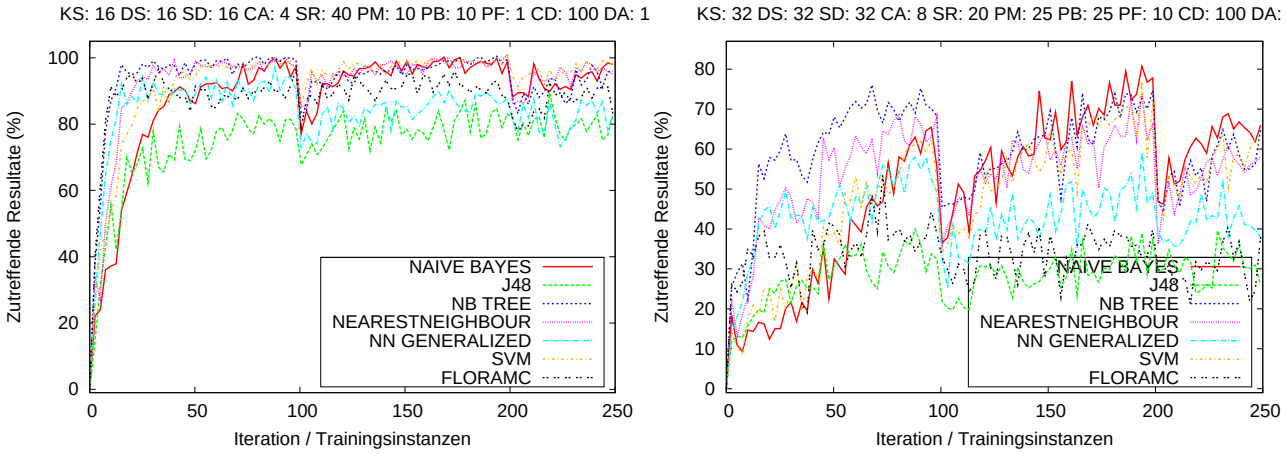
FLORA-MC greift zudem auf den Ansatz der WAH zurück, um Aussagen über die aktuelle Genauigkeit und Phase des Lernens zur Modellgenerierung zu nutzen. Dadurch ist dieses Verfahren jedoch auch abhängig von den Parametern und Einstellungen der Heuristik. Kommt es durch die Fehler innerhalb der Trainingsdaten nicht zu einer Stabilisierung des Modells, so werden weiterhin Instanzen zum Fenster hinzugefügt und im Falle einer Konzeptänderung kommt es nicht zu Reduktion des Fensters. Werden diese Parameter auf ein Szenario angepasst, bei dem (neben der Konzeptänderung) wenige Fehler in den Trainingsdaten zu erwarten sind, wie beispielsweise bei dem zuvor dargestellten Basisszenario und werden diese Parameter auf die effiziente Verarbeitung von Konzeptänderungen eingestellt, so reagiert dieses Verfahren empfindlich gegenüber einer Steigerung der Fehlerrate in den Trainingsdaten. In den Abbildungen 5.42 a) und b) werden hierzu die Szenarien aus Abschnitt 5.4.7 herangezogen und ausgewertet.

Es zeigt sich, dass zur Einstellung der Parameter der WAH Domänenwissen über die Qualität der zu erwartenden Informationen erforderlich ist. Werden dieselben Parameter wie bei dem Basisszenario angewendet, so reduziert sich die Qualität im Vergleich zu anderen Algorithmen.

Der Ansatz FLORA-MC zeigt seine Stärken in Szenarien, in denen weniger Fehler in der Informationsgrundlage zu erwarten ist. In diesen Umgebungen hat FLORA-MC eine hohe Klassifikationsgenauigkeit und die Fähigkeit selbst unter Konzeptänderungen deutlich schneller wieder eine Annäherung an das neue Konzept zu erreichen als viele andere Verfahren.

Es wurden noch weitere Optimierungen des Verfahrens durchgeführt, wie beispielsweise das gezielte Entfernen von widersprüchlichen Instanzen aus der Trainingsmenge oder ein Langzeitgedächtnis, welches verworfene Modelle bei *wiederkehrenden Konzepten* (siehe Abschnitt 5.3.1) heranziehen kann. Dieses Reaktivieren verworfener Modelle kann dazu beitragen, die Robustheit gegenüber fehlerhaftem Feedback oder irrtümlich indizierter Konzeptänderung zu erhöhen.

Wie gut die Verfahren mit einer Konzeptänderung umgehen können, hängt stark von der Größe des Domänenraums, sowie von der Komplexität und der Überdeckung der Konzepte ab. In dieser Arbeit wurden künstlich generierte Konzepte herangezogen, um das Verhalten auf bestimmte Eigenschaften analysieren zu können. Andere



(a) Mäßige Bedingungen

(b) Schwere Bedingungen

Abbildung 5.42: Klassifikationsgenauigkeit von FLORA-MC und anderen Verfahren bei einer Steigerung der Fehlerrate in den Trainingsdaten

Konzepte wie beispielsweise das in unserer Arbeit *Adapting the User Context in Realtime: Tailoring Online Machine Learning Algorithms to Ambient Computing* [SHRS08] modifizierte Stagger-Konzept, verfügt über deutlich weniger Sensoren und mehr Überdeckungen der Konzepte. Dies führt bei Konzeptänderungen verstärkt zu Konflikten. In diesem Zusammenhang ergeben sich noch größere Unterschiede in der Klassifikationsgenauigkeit zwischen klassischen Lernverfahren und FLORA-MC. FLORA-MC zeigt bei dem modifizierten Stagger-Konzept seine Stärken im Vergleich zu bestehenden Ansätzen.

5.7 Fazit

Dieses Kapitel stellte die Arbeiten im Bereich der Informationsauswertung vor. Hierzu wurden die Rahmenbedingungen des Gesamtszenarios auf diesen Bereich abgebildet. Anhand dieser Rahmenbedingungen wurden bestehende Ansätze zur Auswertung von Informationen betrachtet und der adaptive Ansatz unter dem Einsatz von überwachten Lernverfahren ausgewählt. Der im Rahmen dieser Arbeit entwickelte Kontextdienst und dessen iterativer Ansatz zur Anpassung des Modells an das Konzept des Nutzers wurde vorgestellt. Des Weiteren wurde auf die speziellen Herausforderungen eingegangen, vor welchen die Lernverfahren durch diese Vorgehensweise und das Gesamtsystem gestellt werden. Anhand dieser Herausforderungen wurden eine Reihe bestehender Lernverfahren verglichen, wozu eine Testumgebung diente, welche speziell für diesen Anwendungsfall konzipiert wurde. Durch die Testumgebung konnten das Laufzeitverhalten und die Qualität der Verfahren in Abhängigkeit zu den verschiedenen Eigenschaften der Informationsgrundlage bestimmt werden. Wie sich durch die Analysen zeigte, sind die Verfahren relativ robust gegenüber einigen Eigenschaften der Informationsgrundlage. Änderungen im Konzept der Nutzer oder in den Aussagen der Sensoren erzeugen Probleme, welche durch den Einsatz reiner Offline-Ansätze nicht ausreichend abgedeckt werden. Hierzu wurden weitere Ansätze aus den Bereichen der Meta-Lernverfahren und der Online-Learner herangezogen, ausgewählt, umgesetzt und erweitert. Die Analyse zeigte das Optimierungspotential durch den Einsatz dieser Verfahren, sowohl in der Qualität als auch im Speicherbedarf und im Laufzeitverhalten, in Abhängigkeit zu dem zugrundeliegenden Szenario.

6 Realisierung

Im bisherigen Teil der Arbeit wurde ein Kommunikationssystem als angestrebtes Anwendungsszenario betrachtet. Aus diesem Anwendungsszenario wurden Rahmenbedingungen abgeleitet und es wurden eine Architektur, sowie eine Reihe von Komponenten entworfen, welche diesen Rahmenbedingungen genügen. Um den Beweis zu erbringen, dass das Gesamtkonzept erfolgreich in ein funktionsfähiges Systems überführt werden kann, welches den Anforderungen genügt, wurde eine Umsetzung aller Ansätze durchgeführt.

Das Rahmenwerk ContextFramework.KOM stellt eine Implementierung der zuvor beschriebenen Ansätze als Proof-of-Concept dar und besteht aus einer Vielzahl von Komponenten. In diesem Kapitel werden ausgewählte Teile des Rahmenwerkes detaillierter beschrieben. Hierbei wurden verstärkt diejenigen Teile herausgegriffen, welche sich auf die wesentlichen Aspekte dieser Arbeit beziehen und deren Umsetzung erläutern.

Das Rahmenwerk ContextFramework.KOM besteht im Kern aus einer generischen, offenen Middleware. Diese Middleware erlaubt die verteilte Ausführung und Administration von Diensten, sowie die Kommunikation zwischen den Diensten. Die Umsetzung dieser Middleware und die Architektur des Rahmenwerkes werden in Abschnitt 6.1 erläutert.

Die Sensoren bilden einen Teil des Rahmenwerkes. Diese verteilt verfügbaren Informationsquellen wurden ebenfalls in Form von Diensten auf den Endgeräten umgesetzt und auf die jeweilige Plattform angepasst. In Abschnitt 6.2 wird die Umsetzung der Sensoren und der Beschreibung der Sensoren aufgeführt.

Der Kern des Systems beinhaltet zwei Dienste, welche grundlegend für die Informationsgewinnung und deren Auswertung benötigt werden: Das Sensorverzeichnis und der Kontextdienst.

Das Rahmenwerk erlaubt es zur Laufzeit, Sensoren beliebiger Art dynamisch anzubinden, um so automatisiert auf die verfügbaren Sensorinformationen zugreifen zu können. Diese Funktionen werden von dem *Sensorverzeichnis* erbracht und in Abschnitt 6.3 behandelt.

Eine lernende Inferenzlogik schließt aus den verfügbaren Informationen auf den aktuellen Kontext der im System registrierten Benutzer und kann dieses Wissen gezielt zur Unterstützung von Kommunikationsdiensten bereitstellen. Die Umsetzung des Dienstes zur Auswertung des Kontextes wird in Abschnitt 6.4 erläutert.

Das Ziel des Gesamtsystems besteht in der Unterstützung der Anrufverarbeitung durch die automatische Bestimmung und Nutzung des Kontextes. Die hierzu notwendige Integration des Rahmenwerkes in existierende Kommunikationssysteme wird in Abschnitt 6.5 beschrieben.

6.1 Plattformen und Architektur

Das System kombiniert eine Reihe von Technologien aus unterschiedlichen Bereichen zu einem Gesamtsystem mit vielfältigen Funktionen und Möglichkeiten. Der Kern des Systems basiert auf Java und ist somit auf unterschiedlichen Plattformen ausführbar. Die Sensoren müssen jedoch teilweise auf systemnahe Attribute oder Schnittstellen zurückgreifen. Diese Sensoren sind dann nur plattformabhängig einsetzbar. Welche Sensoren umgesetzt wurden und auf welchen Plattformen sie eingesetzt werden können, ist in Abschnitt 6.2 beschrieben.

Dienstorientierte Architektur

Wie in Abschnitt 3.3 erläutert wurde ist die Zielarchitektur des Systems dienstorientiert. Grundlegend besteht die Möglichkeit, einen Dienst jeweils durch ein einzelnes Java-Programm abzubilden.

Ein Dienst kann jedoch auch als ein Software-Packet (*Bundle*) betrachtet werden, welches neben anderen Bundles gleichen Aufbaus auf einer Zielplattform zur Ausführung gebracht wird. Der Vorteile eines gemeinsamen Dienste-Managements ist, dass das dynamische Nachladen weiterer Bundles und die dynamische Komposition von Diensten möglich sind.

Es existiert eine Reihe schwergewichtiger, teilweise kostenpflichtiger Systeme oder Rahmenwerke, wie beispielsweise WebSphere von IBM. Andere Systeme wie Tomcat[1] haben den Fokus auf Webservices zur direkten Interaktion mit Nutzern gelegt (dynamische Generierung von HTTP Seiten). Diese Systeme sind oftmals relativ schwergewichtig und eignen sich daher nur bedingt zum Einsatz auf ressourcenbeschränkten Endgeräten.

[1] `http://tomcat.apache.org/`

OSGI

OSGI[2] (früher *Open Services Gateway Initiative*) basiert auf Java und bietet die Möglichkeit Dienste dynamisch zur Laufzeit nachzuladen. OSGI sieht als Basis eine gemeinsame Spezifikation für OSGI-Bundles vor. Es existiert inzwischen eine Reihe von Implementierungen von OSGI-Rahmenwerken, welche diese Spezifikation erfüllen. Mit Concierge [RA07] wird eine sehr leichtgewichtige und effiziente Umsetzung angeboten, welche der Anforderung (5) aus Abschnitt 2.3 (Minimierung der Kosten und des Aufwands zur Umsetzung und Integration) entspricht. OSGI verfolgt einen generischen Ansatz zur Umsetzung von Diensten. Ein Dienst kann beispielsweise ein kompletter HTTP-Server sein. Er kann aber auch nur ein funktionaler Bestandteil davon sein oder aber auch nur eine Reihe von Java-Klassen beinhalten. Letzteres macht Sinn, da OSGI die Wiederverwendung von Code ermöglicht. Dies bedeutet ein Bundle kann, neben den von ihm selbst erbrachten Funktionalitäten, auch seine Klassen anderen Bundles innerhalb des OSGI-Rahmenwerkes zur Verfügung stellen. OSGI ermöglicht es, Java-Code in Form von Bundles aufzuteilen, nachzuladen und auszuführen. Durch die Nutzung eines Rahmenwerkes wie OSGI wird in der Regel implizit auch eine dienstorientierte Architektur forciert.

Die einzelnen Dienste von ContextFramework.KOM wurden im Rahmen des leichtgewichtigen Concierge OSGI Systems[3] implementiert.

6.1.1 Kommunikation zwischen den Diensten

Zur Auffindung der Sensoren wurden Technologien aus dem Bereich der *Service Lookup*-Protokolle herangezogen. Zur Kommunikation mit Sensor-Diensten über unterschiedliche Plattformen und Netzwerk-Technologien hinweg kommen Technologien aus dem Bereich der Kommunikations-Middleware zum Einsatz. Die verwendeten Technologien wurden hierzu in Abschnitt 4.2.2 eingeführt.

Adressierung von Diensten

Auf der Ebene der Konnektoren gilt es Dienste geeignet zu adressieren. Des Weiteren wird eine Gruppierung von Diensten benötigt, um Dienste zu kategorisieren und auf diesem Wege bestimmte Dienste des Rahmenwerkes von Sensoren zu unterscheiden. Das Sensorverzeichnis muss beispielsweise die Möglichkeit haben, eine Liste aller verfügbaren Sensoren zu erhalten.

In dieser Arbeit wird eine Adressierung der Dienste vergleichbar zu URLs oder *Java Package* Bezeichnungen durchgeführt. Dies ermöglicht sowohl eine direkte Adressierung einzelner Dienste als auch eine hierarchische Gruppierung von Diensten. Während sich einzelne Dienste des Rahmenwerkes mit „sn/services/[Bezeichner]" adressieren lassen, haben Sensoren eine Adresse in Form von „sn/sensors/[Bezeichner]".

Bezogen auf den in Abschnitt 4.2.1 dargestellten Ablauf kann das Sensorverzeichnis bei der Suche nach Diensten nach „sn/sensors" suchen lassen.

Die komplette Adressierung eines Dienstes setzt sich zusammen aus dem Konnektortyp, dem Dienst-Bezeichner und einer zur Laufzeit gültigen ID (um mehrere Dienste desselben Typs an einem Konnektor zu ermöglichen). Die Adresse sieht dann wie folgt aus:

```
service:[Konnektor]:[Bezeichner]:[Eindeutige ID]
```

Die Adresse anhand eines konkreten Beispiels wäre demnach:

```
service:rmi:sn/sensors/webcam:1
```

Auffindung

Zur Auffindung von Diensten wurde das SLP-Protokoll integriert (siehe Abschnitt A.2). Es lässt sich unabhängig von der Technologie (XML-RPC, RMI oder R-OSGI) in den Basiskonnektoren nutzen. Hierzu wurde die Java-Implementierung *jSLP*[4] eingebunden, welche im Rahmen von Concierge OSGI entwickelt wurde und ebenfalls sehr leichtgewichtig und flexibel einsetzbar ist. Es basiert ebenfalls auf der Adressierung von Diensten über URLs und lässt sich gut mit der zuvor beschriebenen Form der Adressierung kombinieren.

Um bei der Entwicklung Verbindungen zu bestimmten Diensten aufzubauen, existiert eine Möglichkeit der direkten Adressierung des Zielsystems. Durch Angabe der gewünschten IP-Adresse des Zielsystems kann so das Ergebnis der SLP-Suche vorweggenommen werden.

[2] http://www.osgi.org
[3] http://concierge.sourceforge.net/
[4] http://jslp.sourceforge.net/

Es wurde ein Ansatz gesucht, welcher eine gemeinsame Schnittstelle zur Kommunikation auf der Ebene der Dienste bietet und darunter eine auf das zugrunde liegende System angepasste Kommunikation ermöglicht.

Wie bereits in Abschnitt 4.2.3 erläutert wurde, bieten existierende Rahmenwerke zur Kommunikation oftmals nur plattformabhängige Kommunikation oder sind insgesamt relativ schwergewichtig. Da in diesem System eine leichtgewichtige und flexible Möglichkeit der Kommunikation zwischen Diensten benötigt wird, wurde eine eigene Schnittstelle entwickelt. Diese *Konnektor-Schnittstelle* beschreibt die grundlegenden Funktionen zum Aufruf entfernter Methoden und den Transport komplexer Datenstrukturen.

Es wurde eine Reihe von *Basiskonnektoren* umgesetzt, welche diese Schnittstelle implementieren und welche direkt auf Technologien zur Spezifikation eines entfernten Methodenaufrufes aufsetzen (siehe Abschnitt A.2):

- **RMI:** Die *Remote Method Invocation* ist eine Technologie, welche von Java bereitgestellt wird.
- **R-OSGI:** Das Bundle Remote-OSGI[5] ist ein Dienst, welcher von den Entwicklern von Concierge bereitgestellt wird, um eine transparente Kommunikation zwischen verteilten OSGI-Bundles zu ermöglichen.
- **XML-RPC:** Der XML-RPC Konnektor erlaubt die Kommunikation zu Diensten auf anderen Plattformen, welche nicht auf Java basieren.
- **LOCAL:** Der LOCAL Konnektor ermöglicht eine Kommunikation zwischen Diensten, welche in derselben virtuellen Maschine von Java (Java-VM) ausgeführt werden.

Diese *Basiskonnektoren* ermöglichen die Kommunikation zwischen Diensten. Je nach Bedarf können unterschiedliche Konnektoren eingesetzt werden. Die Eigenschaften der Konnektoren können auch kombiniert und erweitert werden. Hierzu wurden eine Reihe weiterer *Metakonnektoren* umgesetzt, welche auf die Basiskonnektoren aufsetzen. Diese Metakonnektoren bieten dieselbe Schnittstelle wie die Basiskonnektoren, wodurch der Zugriff transparent erfolgen kann (gleichermaßen wie auf einen Basiskonnektor).

- **MULTI:** Der MULTI Konnektor erlaubt die Kombination mehrerer Konnektoren. So können beispielsweise das flexible XML-RPC und das effiziente RMI kombiniert und parallel betrieben werden.
- **OFFLINE:** Der OFFLINE Konnektor kann auf den Endgeräten betrieben werden, um in Situationen Daten zu sammeln, in denen das Endgerät keine Verbindung zum Gesamtsystem hat. Hierzu erfasst der OFFLINE Konnektor alle Sensoren, welche auf dem Endgerät über den Sensor angemeldet werden. Sobald die Verbindung zum Gesamtsystem unterbrochen wurde, erfasst und speichert dieser Konnektor die Daten an den lokalen Sensoren. Kann die Verbindung wieder hergestellt werden, so bietet der OFFLINE Konnektor einen weiteren Dienst an, welcher die Zustände der Sensoren in dem Zeitraum mit dem entsprechenden Zeitstempel zur Verfügung stellt.
- **REPLAY:** Der REPLAY Konnektor zeichnet alle Sensorzustände auf und bietet die Möglichkeit diese später wieder in das System einzuspielen. So können die Zustände aller Endgeräte aufgezeichnet und für Tests und Analysen zu einem späteren Zeitpunkt wieder in das System eingespielt werden. Dies kann auch beschleunigt in simulierter Zeit erfolgen (siehe Abschnitt 6.1.3).

Durch den gewählten Ansatz der Konnektoren ist es möglich verschiedene Plattformen an das Rahmenwerk anzubinden. Wie in Abbildung 6.1 dargestellt wird, kann für jede Plattform ein geeigneter Konnektor gewählt und eingesetzt werden. Die umgesetzten Konnektoren setzen direkt auf Basistechnologien zur Kommunikation auf und sind daher sehr leichtgewichtig. Der Ansatz ist darüber hinaus flexibel und kann um weitere Plattformen erweitert werden, indem ein entsprechender Konnektor hinzugefügt wird. Ebenso kann der Funktionsumfang der Konnektoren erweitert werden, indem ein Metakonnektor hinzugefügt wird, welcher auf einem Konnektor aufsetzt und ihn um die benötigte Funktionalität erweitert (wie beispielsweise der OFFLINE Konnektor in der Darstellung 6.1 links).

Vergleich der Konnektoren

Die implementierten Basiskonnektoren weisen alle die gleiche Funktionalität auf. Sie unterscheiden sich jedoch zum einen in den Plattformen, auf denen sie eingesetzt werden können, und zum anderen haben sie unterschiedliche Eigenschaften im Bezug auf den Aufwand und die Verzögerung, welche sie bei der Verarbeitung von Anfragen erzeugen. Um das Laufzeitverhalten zu analysieren wurde ein Dienst umgesetzt, welcher es erlaubt, eine Vielzahl

[5] `http://r-osgi.sourceforge.net/`

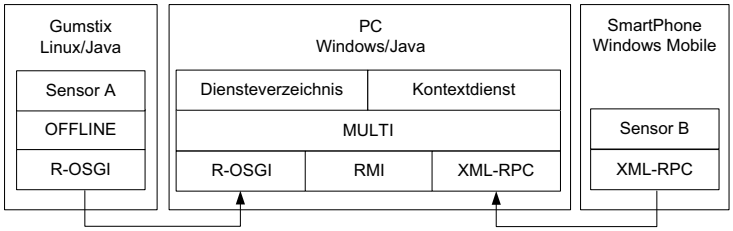


Abbildung 6.1: Anwendungsbeispiel: Nutzung von Konnektoren

von Anfragen an einen entfernten Dienst abzusetzen. Dieser *Anfragengenerator* bietet die Möglichkeit, den anzufragenden Dienst, die aufzurufende Methode, die Dauer des Tests und die durchschnittliche Anzahl von Anfragen pro Sekunde festzulegen. Die Anfragen werden jeweils von einem gesonderten Thread ausgeführt und mit einem (gleich verteilten) Jitter von +/- 500ms versehen, um ein unregelmäßiges Eintreffen von Anfragen nachzubilden.

Jeder Test wurde 30 Sekunden lang durchgeführt, wobei eine Methode am entfernten Dienst aufgerufen wurde, welche eine einfache Abfrage eines Wertes nachbildet, ohne auf der Seite des entfernten Dienstes zusätzlichen Bearbeitungsaufwand zu generieren. Die Anfragen wurden von einem PC (gleicher Bauart wie in der Analyse aus Abschnitt 5.4.5) aus an verschiedene Endgeräte gesendet.

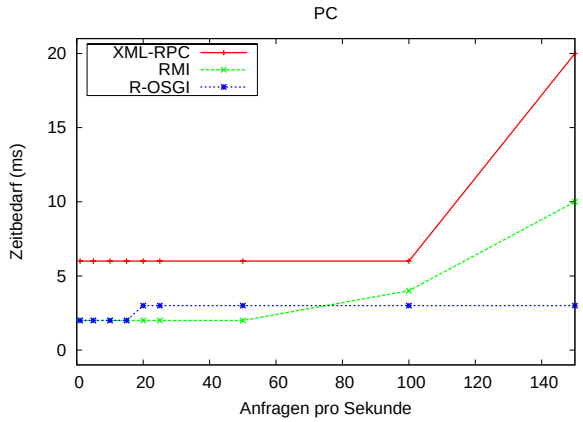
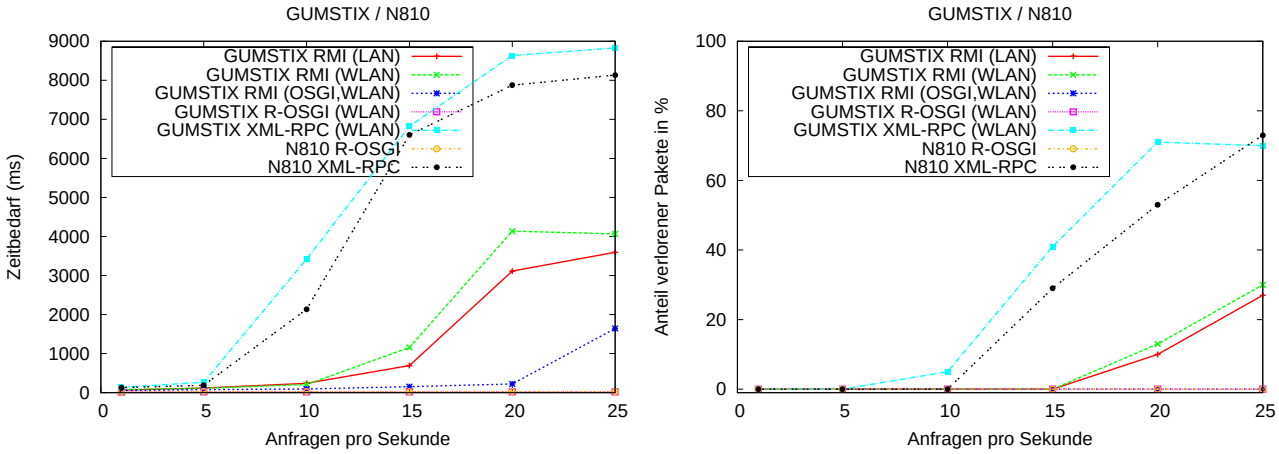


Abbildung 6.2: Vergleich der Konnektoren am PC

In Abbildung 6.2 wurden die Anfragen an einen anderen PC gleicher Bauart gesendet. Bei einzelnen Anfragen benötigen RMI und R-OSGI mit 2ms im Durchschnitt 4ms weniger Zeit als XML-RPC. Bis zu einem Aufkommen von bis zu 100 Anfragen pro Sekunde konnten alle Konnektoren, ohne merkliche Veränderungen an ihrem Laufzeitverhalten, arbeiten. Ab diesem Punkt bis hin zu 150 Anfragen pro Sekunde zeigten XML-RPC und RMI einen geringen Anstieg der Verzögerung bei der Verarbeitung der Anfragen. R-OSGI übertrifft diese Leistung durch gleichbleibende Verzögerung bei bis 150 Anfragen pro Sekunde. Mit gemessen Verzögerungen von durchschnittlich rund drei bis sieben Millisekunden und bei einer Anzahl von bis zu 100 Anfragen pro Sekunde ist dieses Laufzeitverhalten für das angestrebte Anwendungsszenario gut geeignet. Im nächsten Szenario wurden vergleichbare Tests mit ressourcenbeschränkten Systemen wie der Gumstix-Plattform oder dem Nokia N810 (für Hardwaredetails siehe Abschnitt 6.2.1) durchgeführt. In Abbildung 6.3 werden die Ergebnisse dieses Tests dargestellt.

Diese ressourcenbeschränkte Systeme zeigen weitaus höhere Latenzen im Vergleich zu den Messungen mit zwei PCs. Einzelne Anfragen an den Gumstix per WLAN benötigen beispielsweise mit R-OSGI 15ms, mit RMI 66ms und mit XML-RPC ganze 135ms. Steigt die Anzahl von Anfragen pro Sekunde, so steigt auch der Aufwand zur Verarbeitung auf dem Endgerät. Es gibt jedoch deutliche Unterschiede zwischen den Konnektoren. Bei XML-RPC sind die Endgeräte ab einer Rate von 5 bis 15 Anfragen überfordert, was zu starken Verzögerungen und fehlgeschlagenen Anfragen führen kann (siehe Abbildung 6.3). RMI hingegen ermöglicht eine Rate von bis 10 bis 20 Anfragen pro Sekunde.

R-OSGI hebt sich klar gegenüber den anderen Ansätzen hervor. Mit R-OSGI sind auch Raten jenseits von 25 Anfragen pro Sekunde ohne merklichen Anstieg in der Verzögerung möglich.



(a) Geschwindigkeit der Verarbeitung von Anfragen

(b) Fehlgeschlagene Anfragen

Abbildung 6.3: Vergleich der Konnektoren an ressourcenbeschränkten Endgeräten

Ein weiteres interessantes Ergebnis zeigt der Vergleich vom RMI, welches innerhalb von OSGI verwendet wurde und der normalen Verwendung außerhalb von OSGI (siehe Abschnitt 6.1.3). In dem Testaufbau zeigt RMI innerhalb von OSGI ein deutlich besseres Laufzeitverhalten als ohne.

Die Wahl des geeigneten Konnektors hat also gerade bei ressourcenbeschränkten Systemen einen maßgeblichen Einfluss auf die Menge der möglichen Sensoranfragen bei zeitbeschränkten Suchanfragen.

6.1.2 Integration und Wartung

Um Konfigurationskosten an Rechnern und mobilen Endgeräten zu vermeiden, wurde bei der Entwicklung des Systems konsequent Wert auf die Einsetzbarkeit bestehender Geräte und Technologien gelegt. Insbesondere wurde durch eine einfache Installationsroutine dafür Sorge getragen, dass der administrative Aufwand gering bleibt. Das System bietet die Möglichkeit neue Informationsquellen mit geringem Aufwand zu integrieren.

Management der Dienste

Es wurde ein Dienst umgesetzt und integriert, welcher neben den Bundles auch Konfigurationen bereitstellt. Dieses *Repository* ist eine Art Fileserver und bietet den Zugriff auf die entsprechenden Dateien. Die Startfolge der Bundles in OSGI wird standardmäßig über statische Konfigurationsskripte durchgeführt. Um die Ausführung von Diensten anhand der zugrundeliegenden Plattform, sowie jeweils darauf abgestimmter Konfigurationen, zu ermöglichen, wurde ein weiterer Dienst integriert. Dieser Dienst führt ein *Lifecycle-Management* der Dienste des Rahmenwerkes durch. Wie in Abbildung 6.4 dargestellt, wird durch diesen Dienst in der Startphase die unterliegende Plattform ermittelt und die entsprechende Konfiguration nachgeladen. Die Konfiguration bestimmt dann, welche Sensoren und Dienste auf der Plattform ausgeführt werden sollen und lädt die entsprechenden Bundles nach.

Konfiguration

ContextFramework.KOM bietet eine XML-basierte Form der Konfiguration. Die Konfiguration besteht aus mehreren Teilen:

- **Basiskonfiguration:** Angabe der Default-Werte.
- **Systemspezifische Konfiguration:** Auf die jeweilige, erkannte Plattform angepasste Konfiguration.
- **Anwendungsspezifische Konfiguration:** Definition der je nach Anwendungsfall benötigten Einstellungen (beispielsweise die Nutzung eines Systems als Server, Einzelplatz-Testumgebung oder als Endgerät eines Teilnehmers).
- **Nutzerspezifische Konfiguration:** Vom Nutzer selbst vorgenommene Änderungen der Einstellungen.

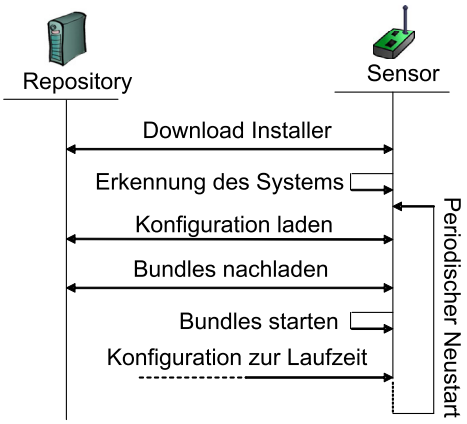


Abbildung 6.4: Startphase des Rahmenwerkes

Diese Konfigurationen werden in dieser Reihenfolge geladen (je nach Einstellung auch vom Repository heruntergeladen) und setzen aufeinander auf. Dies bedeutet, dass eine nachfolgende Konfiguration die vorangehenden Einstellungen gegebenenfalls überschreibt.

Ausgaben

Das Rahmenwerk nutzt leichgewichtige Verfahren, um die Ausgaben des Systems zu verwalten. Je nach Klasse kann ein Ausgabe-Level (aus der Menge *DEBUG*, *INFO*, *WARNING*, *ERROR*) zugewiesen werden. Je nach Konfiguration und Ausgabe-Level werden die Ausgaben auf der Konsole ausgegeben, in eine Datei geschrieben oder verworfen. Zudem wurde die Möglichkeit geschaffen per Telnet auf die Konsole zuzugreifen, um eine Ausführung und Bedienung des Systems im Hintergrund zu vereinfachen.

Administration

Das System beinhaltet einen *Portal-Dienst*, welcher einen leichtgewichtigen Webserver bereitstellt. Die durch den Portal-Dienst aufgebaute Oberfläche ist in Abbildung 6.5 dargestellt. In diesem Rahmen wurde dem Nutzer eine Vielzahl von Funktionalitäten zugänglich gemacht. Zu den Grundfunktionalitäten, welche vom Rahmenwerk angeboten werden zählt unter anderem:

- Einfacher Zugriff auf relevante Einstellungen der Konfiguration. Hierzu gehört die Festlegung der Individuals, sowie das Nachladen oder Stoppen von Diensten.
- Zugriff auf die Ausgaben des System. Anzeige der Ausgaben, sowie Einstellung der Log-Level pro Klasse zur Laufzeit.
- Detaillierte Informationen über die genutzten Konnektoren. Hierzu gehören die Auflistung aller lokal angebotenen Dienste, sowie aller gesuchten und aufgefundenen, entfernten Dienste. Weitere Informationen sind die Aufrufhäufigkeit, der Aufwand, welcher durch Managementnachrichten erzeugt wurde und die Bearbeitungsdauer der Aufrufe. Ebenso werden alle angebundenen Systeme aufgelistet und deren Portal wiederum verknüpft.
- Auflistung aller lokalen Dienste und deren Methoden.
- Manueller Aufruf der Methoden an den Schnittstellen (gegebenenfalls mit Angabe von Parametern).
- Zugriff auf die Konfiguration des Systems.
- Direkte Ansteuerung von OSGI-Befehlen.
- Remote Zugriff auf die OSGI-Konsole per eingebettetem Telnet-Applet.

Der *Portal-Dienst* wurde den Basisfunktionen von *Tomcat* nachempfunden. Jeder Dienst hat die Möglichkeit seine Funktionalitäten an das Portal anzubinden und so dem Nutzer zugänglich zu machen. Hierdurch entstehen eine Reihe weiterer Funktionen, welche durch die jeweiligen Dienste angeboten werden:

- **Repository-Dienst:** Download des *Installers*.
- **Kontextdienst:** Anzeige der Zustände aller Nutzer sowie ein Feedbackagent.
- **Sensorverzeichnis:** Anzeige der registrierten Sensoren, sowie manuelle Durchführung von Suchanfragen.
- **Auswertungsdienst:** Anzeige des Modells und der Trainingsinstanzen, sowie Modifikation der Parameter und des angewendeten Lernverfahrens.

Abbildung 6.5: ContextFramework.KOM Portal

6.1.3 Möglichkeiten der Ausführung

Das System kann in mehreren Modi ausgeführt werden:

- **OSGI:** Grundlegend besteht die Möglichkeit die Dienste innerhalb eines OSGI-Frameworks zu starten. Dies ermöglicht das Nachladen von Programmcode während der Laufzeit und die Nutzung des auf R-OSGI basierenden Konnektors (siehe Abschnitt A.2 und 6.1.1).
- **Debug:** Das Rahmenwerk kann auch ohne OSGI genutzt werden. Dies ist beispielsweise bei der Entwicklung und für Debugging-Aufgaben hilfreich. Es erfordert jedoch, dass der Code vollständig vorliegen muss, da das Nachladen von Bundles zur Laufzeit ohne OSGI nicht möglich ist. Durch das in Abschnitt 6.1.2 beschriebene Dienste-Management vom ContextFramework.KOM ist es möglich, das System von den OSGI-spezifischen Konfigurationsskripten zu entkoppeln und die Startfolge der Dienste manuell zu regeln.
- **Simulation:** Ein speziell für das ContextFramework entworfener Scheduler, welcher durchgängig im System genutzt wurde, ermöglicht es, das System auch in simulierter Zeit und Umgebung zu starten. Mithilfe des Schedulers lassen sich Threads, Wartezeiten und ein synchroner Zugriff auf Objekte sowohl in Realzeit als auch in simulierter Zeit umsetzen. Hierbei kann ebenfalls über einen speziellen Konnektor das Verhalten der Informationsquellen zur Realzeit aufgenommen und zur simulierten Zeit wiedergegeben werden (siehe REPLAY-Konnektor in Abschnitt 6.1.1)

6.2 Sensoren

Das Rahmenwerk vereinfacht die Entwicklung und Integration von Sensoren. So werden alle Basisfunktionen eines Sensors bereitgestellt und müssen nur um die sensorspezifischen Funktionen erweitert werden (siehe Abschnitt 4.3.1 und 4.3). Eine weitere Unterstützung bei der Integration von Sensoren bietet die Generierung der notwendigen Beschreibung (wie später in Abschnitt 6.2.2 erläutert wird).

Basis Schema

Viele der Funktionalitäten eines Sensors lassen sich generisch auf andere abbilden. Daher liegt es nahe eine gemeinsame Basisschnittstelle anzubieten. Diese Basisschnittstelle hat folgende Aufgaben:

- Wiederverwendung gemeinsamer Funktionalitäten.
- Abstraktionsebene für Sensoren - Vereinfachung der Schnittstelle für Sensor-Entwickler.
- Kapselung und Anbindung des Konnektors.
- Generierung und Anbindung der Sensorbeschreibung.

Wie in Abbildung 6.6 dargestellt, erfolgt eine Anbindung eines Sensors in den folgenden Schritten:

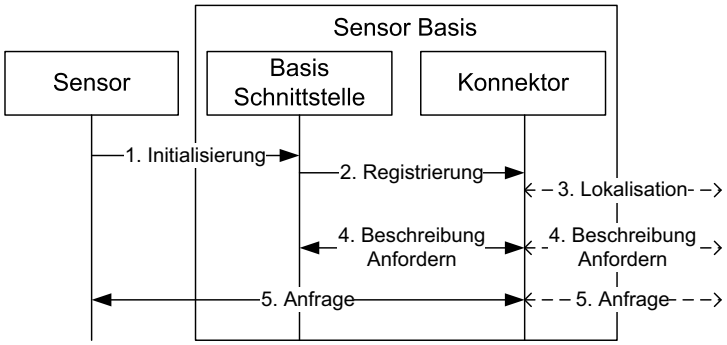


Abbildung 6.6: Ablauf: Nutzung der Sensor-Basis

1. **Initialisierung:** Zur Anmeldung eines Sensors müssen die grundlegenden Beschreibungselemente und die Schnittstelle mit den erweiterten Funktionalitäten (sowie eine Referenz auf das Objekt, welches die Schnittstelle instanziiert) an die Sensor-Basis übergeben werden.
2. **Registrierung:** Anhand der übergebenen Schnittstelle wird die Beschreibung der erweiterten Funktionalitäten des Sensors generiert. Die Sensor-Basis meldet die erweiterten Funktionalitäten, zusammen mit den Grundfunktionalitäten am Konnektor an.
3. **Lokalisation:** Über den Konnektor wird der Sensor als neuer Dienst im Netz angeboten.
4. **Beschreibung anfordern:** Sobald ein System den neuen Dienst im Netz erkennt, kann dieser die Beschreibungselemente des Sensors anfordern.
5. **Anfrage:** Eine Anfrage auf den eigentlichen Sensor wird direkt auf das bei der Anmeldung übergebene Objekt weitergereicht, welches die Schnittstelle implementiert.

Durch die Kapselung aller für die Anbindung eines Sensors benötigten Funktionalitäten in Form einer *Sensor-Basis*, kann eine Anbindung eines Sensors stark vereinfacht und der Aufwand letztlich auf die Übergabe einer Schnittstelle (mit semantischen Anmerkungen) bei der Anmeldung reduziert werden.

6.2.1 Umsetzung von Sensoren

Im Rahmen der Umsetzung von ContextFramework.KOM wurde eine Reihe von Sensoren umgesetzt die innerhalb des Rahmenwerkes nutzbar sind. Die Plattformen, welche hierbei getestet und genutzt wurden, sind:

- **Windows, Linux:** Standard PC System mit dem jeweiligen Betriebssystem.
- **Nokia N810:** Linux basierter PDA, ARM 400 MHz CPU, 128 MB RAM und 2 GB Flash-Speicher.
- **Windows Mobile (WM):** Smartphone mit Unterstützung von Microsofts .NET Compact Framework (CF).
- **Gumstix:** Linux basiertes, eingebettetes (engl. *embedded*) System. Der Gumstix *Verdex* hat eine 600 MHz CPU, 128 MB RAM und 32 MB Flash-Speicher.
- **NSLU2:** Ein *Network Attached Storage*-Gerät (NAS) mit Linux, 266 MHz ARM CPU und 32 MB RAM (vergleichbar mit handelsüblichen Breitbandroutern).
- **SunSpot:** Drahtloser Sensor mit 180 MHz ARM CPU, 512 KB Ram und 4 MB Flash-Speicher. Fähigkeit zur Ausführung von Java-MicroEdition Code, sowie integriertem Helligkeits-, Temperatur- und Beschleunigungssensor. Die Kommunikation erfolgt über IEEE 802.15-4.
- **TMote:** Der drahtlose Sensor *Telos B* hat eine 8 MHz CPU, 10 KB RAM und 1 MB Flash-Speicher. Die Funkschnittstelle unterstützt 802.15-4. Auf dem Sensorknoten sind Helligkeits-, Temperatur- und Feuchtigkeitssensoren integriert.

Die Plattformabhängigkeit einzelner Sensoren beruht meist auf der Nutzung systemspezifischer Schnittstellen. Die Tabelle 6.1 gibt einen Überblick über die entwickelten Sensoren und die jeweils nutzbaren Plattformen.

Bezeichnung	Plattformen	Erfasste Informationen
Anwendung	Windows	Aktuelle Anwendung im Vordergrund eines Nutzers, Tastaturaktivität
Aufgabenplaner	Windows	Ausgewählte Aufgabe aus einem Aufgabenplaner
Batterie	Nokia N810	Ladestand der Batterie
Bewegung	SunSpot	Bewegungs-, Lagesensor
Bluetooth	Linux, Gumstix, NSLU2	Sichtbare Geräte in der Umgebung, Signalstärke
Computer Aktivität	Windows	Mausbewegung
Datenbank	*	Parameterisierbare Abbildungen (z.B. Abbildung von Name auf Arbeitsplatz)
E-Mail	*	Nachrichtenaustausch mit anderen Nutzern
Endgeräte	*	Endgerät eingeschaltet, Adresse (Telefonnummer)
Endgeräte- Temperatur	NokiaN810	Interner Temperaturfühler
GPS	WM, NokiaN810	Position
Helligkeit	NokiaN810, TMote, SunSpot	Lichtstärke im Raum
Kalender	Windows, WM	Aktuelle, nächste Ereignisse
Kamera	Windows, Linux, Gumstix, NSLU2	Bewegung im Raum, Bild
Kontakte	Windows, WM	Abbildung von Name zu Adressen und umgekehrt, Beziehungen
Kontext	*	Zustände am Kontextdienst (zur Nutzung anderer Kontextdimenstionen)
Nachbarschaft	Tmote, SunSpot	Sichtbare Sensorknoten in der Umgebung (ID und Signalstärke)
Ping	*	Erreichbarkeit von Geräten im Netzwerk
Puls	SunSpot	Herzschlag
Skype	Windows	Eingestelltes Verfügbarkeitslevel, Kontakte
Stuhl	TMote	Nutzung eines Bürostuhls (eingebaute Kontaktmatte)
Telefonie	*	Angemeldete Geräte, aktive Gespräche
Temperatur	TMote, SunSpot	Umgebungstemperatur
Terminkalender	Windows	Eingetragene Ereignisse im Terminkalender
WLan	Linux, WM, Gumstix, NSLU2	Sichtbare Funknetzwerke in der Umgebung, Signalstärke

* plattformunabhängige Integration des Sensors.

Tabelle 6.1: Überblick über die im Rahmen von ContextFramwork.KOM entwickelten Sensoren

6.2.2 Sensorbeschreibung

Das System ist offen gegenüber neuen Sensoren und Technologien. So können im Nachhinein auch neue, unbekannte Sensoren am System angemeldet und genutzt werden. Sensoren und deren benötigte bzw. angebotene Informationen werden hierbei geeignet beschrieben, um neben der Relevanz eines Sensors für bestimmte Suchanfragen auch eine korrekte Verarbeitung und Nutzung der Informationen zu gewährleisten. Zur Beschreibung kommen sowohl Technologien aus dem Bereich der Schnittstellenbeschreibung von Diensten als auch Methoden zur semantischen Beschreibung von Diensten und Informationen zum Einsatz.

Die Nutzung eines Standards oder eines offenen Ansatzes verspricht Kompatibilität zu anderen Systemen und die Wiederverwendung von bestehenden und etablierten Technologien und Programmen. Wie in Abschnitt 4.4 beschrieben, existieren verschiedene Ansätze um Dienste und Informationen semantisch zu beschreiben.

Die Hauptaufgabe besteht in der semantischen Beschreibung der Eingabe- und Ausgabedaten eines Sensors, sowie der Beschreibung der Relation eines Sensors zu anderen Objekten (Individuals). Die beschriebenen Technologien SAWSDL, WSMO und OWL-S sind hierzu prinzipiell geeignet. Für diese Arbeit wurde OWL-S ausgewählt, da eine Reihe weiterer Technologien und Werkzeuge vorhanden sind, welche sich mit OWL-S gut anwenden lassen:

- **OWL:** Das auf XML und RDF basierende Format OWL wird in vielen Arbeiten zur Beschreibung von Ontologien genutzt. OWL-S geht aus OWL hervor und lässt sich gut in Kombination damit anwenden.
- **JENA:** Das JENA-Framework[6] basiert auf Java und bietet eine umfangreiche API, um mit OWL und OWL-S zu arbeiten.
- **CMU:** Diese API für OWL-S wurde von der Carnegie Mellon University erstellt[7]. Da das Rahmenwerk grundlegend unabhängig von anderen Systemen betrieben werden soll und diese API nur mit Verweisen auf extern gelagerte Ontologien arbeitet, kam dieses System für diese Arbeit nicht in Betracht.
- **Mindswap:** Dieses System der University of Maryland bietet ebenfalls eine API für OWL-S, welche wiederum auf JENA aufsetzt[8]. Diese API ermöglicht jedoch die Verwendung eines Zwischenspeichers für die Beschreibungen, welcher auch mit lokal gelagerten Beschreibungen gefüllt werden kann. Dies ermöglicht die lokale Bereitstellung und den Zugriff auf OWL und OWL-S Dokumente innerhalb abgeschlossener Systeme. Diese API wird im Rahmen dieser Arbeit genutzt, um mit OWL-S Dateien zu arbeiten.
- **SWRL:** Weitere Technologien, wie SWRL ermöglichen die Definition von Anfragen über semantische Inhalte einer Ontologie. Solche Anfragen können mit SWRL anhand der semantisch beschriebenen Suchergebnisse und der Ontologie ausgewertet werden. Die Nutzung von SWRL zur Auswertung der Suchergebnisse zählt zu den Aufgaben nachfolgender Arbeiten am Rahmenwerk.
- **Protégé:** Der OWL Editor Protégé[9] eignet sich gut, um mit OWL und OWL-S zu arbeiten. Der Editor ist speziell für OWL entworfen und bietet ein entsprechend angepasstes GUI an.
- **SWeDE:** Das *Semantic Web Development Environment* (SWeDE) bietet einen OWL Editor als Eclipse Plugin an. Durch die Integration in Eclipse können bekannte Funktionen wie Syntax-Hervorhebung, Code-Vervollständigung und die syntaktische Fehlererkennung bei der Bearbeitung von OWL Dateien genutzt werden.

Generierung von OWL-S-Beschreibungen

Durch die Entscheidung OWL-S[10] zu nutzen, wird für jeden Sensor eine OWL-S konforme Beschreibung benötigt. Wie in Abschnitt A.4 beschrieben, werden dazu mehrere Dateien benötigt: *OWL-S Service*, *OWL-S Profile*, *OWL-S Process* und *OWL-S Grounding*. In Abschnitt A.5 des Anhangs wird die OWL-S Beschreibung eines Sensors bereits in gekürzter Form dargestellt. Wie sich daran erkennen lässt ist der Umfang dieser XML basierten Beschreibung relativ groß. Eine weitere Funktionalität, welche im Rahmen von ContextFramework.KOM entwickelt wurde, ist die automatische Generierung der OWL-S Beschreibung. Hierzu wird ein Java-Interface mit zusätzlichen Annotationen (in Form von normalen Kommentaren) versehen genutzt. Innerhalb dieser Annotationen können semantische Zusatzinformationen definiert werden. In dem folgenden Beispiel wird das Java-Interface des Skype-Sensors dargestellt:

[6] http://jena.sourceforge.net/
[7] http://www.daml.ri.cmu.edu/owlsapi/
[8] http://www.mindswap.org/2004/owl-s/api/
[9] http://protege.stanford.edu/
[10] OWL-S 1.1: http://www.daml.org/services/owl-s/1.1/overview/

```
package sn.sensors.skype;
/**
 * This class accesses the Skype client and returns its status
 * @category AbstractSensor
 */

/**
 * @return SkypeActivity
 * @expirationTime 10000
 */
public String getSkypeStatus();
```

Die Definition der Java-Methode kann direkt genutzt werden, um beispielsweise Elemente des *OWL-S Groundings* zu generieren. Die mit speziellen Schlüsselworten versehenen Kommentare (*@category*, *@param*, *@return* und *@expirationTime*) werden ausgelesen, um die Kategorie des Sensors, den semantischen Datentyp und weitere Eigenschaften wie deren Gültigkeit zu bestimmen.

In der Abbildung 6.7 ist die Anwendung dieser semantisch annotierten Schnittstelle dargestellt.

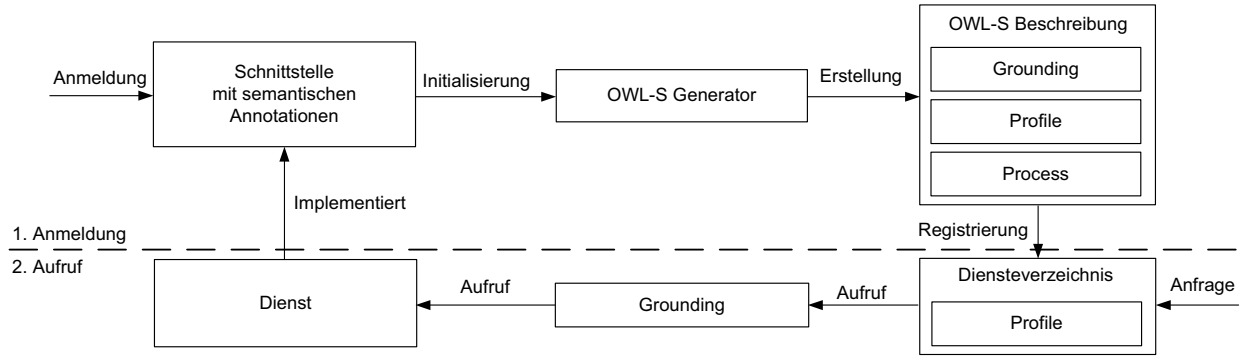


Abbildung 6.7: OWL-S Generierung und Nutzung

Ein Sensor benötigt für die Anbindung an den Konnektor ein Interface. Dieses Interface als Grundlage für die OWL-S-Beschreibung zu nutzen ist weitaus intuitiver und einfacher, als manuell eine OWL-S-Beschreibung zu erstellen oder zu editieren. Durch die Integration in das Rahmenwerk kann die Beschreibung zur Laufzeit generiert werden, wodurch kein zusätzlicher Schritt in der Entwicklung oder Ausführung von Diensten notwendig wird.

Wie in Abschnitt 4.4 beschrieben, steht bei der Beschreibung eines Sensors die semantische Beschreibung der bereitgestellten Informationen im Vordergrund. Die Funktionsweise lässt sich bei normalen Sensoren meist auf die Bereitstellung der jeweiligen Information reduzieren. Daher wird ein Großteil der Möglichkeiten, welche durch die Beschreibung im *OWL-S Process* zur Verfügung gestellt werden, nicht benötigt. Relevant wird dieser Teil beim Zwischenspeichern von Sensordaten, da dort entsprechende Informationen über die Gültigkeit der Sensordaten enthalten sind.

6.2.3 Gateway

Zur transparenten Anbindung drahtloser Sensornetzwerke wurde ein *Gateway* entwickelt. Dieses Gateway bildet den Proof-of-Concept zur Anbindung von Kleinstgeräten an das Rahmenwerk. Wie in Abbildung 6.8 dargestellt, wird jeder aufgefundene, drahtlose Sensor am Gateway durch einen *Proxy-Sensor* am Rahmenwerk angemeldet. Dieser Proxy-Sensor bietet auf der Seite des Rahmenwerkes eine vollständige Implementierung der Sensor-Schnittstelle an. Auf der anderen Seite werden die Anfragen auf die Funktionen der drahtlosen Sensoren abgebildet.

Auf der Seite der drahtlosen Sensoren besteht eine sehr starke Ressourcenbeschränkung. Diese führt zu Herausforderungen im Bereich der Selbstbeschreibung von Sensoren und der Erfüllung von hohen QoS-Anforderungen.

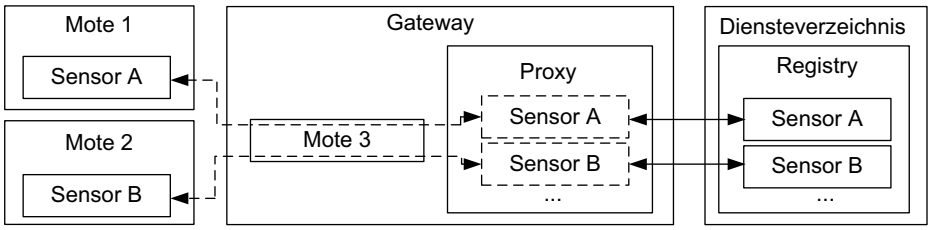


Abbildung 6.8: Anwendungsbeispiel: Nutzung des Gateways

Beschreibung von drahtlosen Sensoren

Eine vollständige Beschreibung eines Sensors auf der Basis des herangezogenen OWL-S Standards erfordert eine relativ große Menge an Speicher und Transfervolumen. Der Einsatz eines solchen Standards auf Systemen wie beispielsweise den TMotes ist nicht sinnvoll. Möglichkeiten diesem zu begegnen bestehen meist in Ansätzen wie:

- Kompression der Beschreibung [RHS09].
- Reduzierung auf wesentliche Aussagen.
- Nutzung geeigneter, ressourcensparender Formate und Auszeichnungssprachen.
- Verweis auf externe, öffentlich zugängliche Speicherorte mit der vollständigen Beschreibung des Sensortyps.

Aktuell wird die zweite Variante angewandt. In weiteren, aktuell durchgeführten Arbeiten am Rahmenwerk werden auch die anderen Ansätze weiter verfolgt.

Erfüllung von QoS Anforderungen

Aktuell werden Anfragen direkt an den jeweiligen drahtlosen Sensor weitergeleitet. Die direkte Anfrage eines Sensors erfordert eine Reihe von Aufrufen, welche dazu möglicherweise über mehrere drahtlose Sensoren weitergeleitet werden müssen. Eine solche Anfrage würde einen erhöhten Zeitaufwand mit sich führen, wodurch mögliche QoS Anforderungen verletzt werden könnten. Zum einen wird diese Herausforderung durch den Mechanismus zur Vorhaltung von Sensorinformationen (engl. *Prefetching*) adressiert. Zum anderen behandeln aktuelle Arbeiten am Rahmenwerk mögliche Ansätze zur aktiven Weitergabe von Sensordaten eines drahtlosen Sensors in Richtung des Gateways.

6.3 Sensorverzeichnis

Das Sensorverzeichnis wurde in Abschnitt 4.5 eingeführt. Zentrale Aufgaben des Sensorverzeichnisses sind die Auffindung und Registrierung vorhandener Sensoren im Netz, die Verwaltung der Ontologie, sowie die Möglichkeit, nach bestimmten Sensoren zu suchen.

Das Auffinden von Sensoren kann wie in Abschnitt 6.1.1 beschrieben durchgeführt werden. Findet zum Zeitpunkt der Suche eine Anfrage auf einen nicht mehr verfügbaren Sensor statt, so wird bis zu der Meldung einer Zeitüberschreitung innerhalb der vom Konnektor eingesetzten Übertragungstechnik gewartet. Die Bearbeitung einer Suchanfrage würde in solchen Fällen möglicherweise bis zur Überschreitung der eigenen Zeitvorgabe verzögert werden. Eine weitere notwendige Funktionalität des Sensorverzeichnisses besteht daher in der Überwachung der Verbindung zu den vorgefundenen Sensoren, um mögliche Sensor-Ausfälle zu erkennen, um so frühzeitig die jeweiligen Sensoren aus der Liste zu entfernen. Hierzu enthält jeder Dienst im Rahmenwerk eine Methode, welche sofern der Dienst funktionsfähig ist den Wert *true* zurückliefert (vergleichbar zu einem *Totmannschalter*). Diese Methode wird genutzt, um die Verbindung zu den Diensten und Sensoren zu überwachen und Ausfälle zu erkennen.

6.3.1 Verwaltung der Ontologie

Wie bereits erwähnt, wird für das Rahmenwerk eine lokal gehaltene Ontologie genutzt. Die Ontologie ermöglicht eine Beziehung zwischen Sensoren und Kontextobjekten herzustellen, sowie für verschiedene Suchen, die jeweils relevanten Sensoren zu bestimmen.

Web Ontology Language (OWL)

Die Web Ontology Language wird in vielen Arbeiten zur Beschreibung von Ontologien genutzt [Lac05] und bietet sehr umfangreiche Funktionalitäten.

> "The OWL Web Ontology Language is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics."[11]

Zudem existiert eine Reihe von Werkzeugen, welche sich mit OWL einsetzen lassen (siehe Abschnitt 6.2.2). Für die Integration der Ontologie in das Rahmenwerk wurde in dieser Arbeit ebenfalls OWL genutzt.

OWL nutzt zwei grundlegende Elemente zur Beschreibung einer Ontologie: Die *Klassen* und *Eigenschaften*. Die Eigenschaften lassen sich wiederum aufteilen in:

- **ObjectProperty:** Beziehungen zwischen Objekten.
- **DatatypeProperty:** Beziehungen von Objekten zu Datentypen.
- **AnnotationProperty:** Beschreibung von Objekten.
- **OntologyProperty:** Beziehungen zu anderen Ontologien.

Das folgende Beispiel zeigt die Anwendung der verschiedenen Eigenschaften:

```
<!-- Definition of new ObjectProperty -->
<owl:ObjectProperty rdf:ID="daughterOf"/>

<!-- Definition of new DatatypeProperty -->
<owl:DatatypeProperty rdf:ID="wasBorn">
<rdfs:domain rdf:resource="#Person"/>
<rdfs:range rdfs:resource="&xsd;date">
</owl:DatatypeProperty>

<Person rdf:ID="Person1">
<!-- Usage of predefined Annotation Property -->
<rdfs:comment>the object representing a parent</rdfs:comment>
</Person>

<Person rdf:ID="Person2">
<daughterOf rdf:resource="#Person1"/> <!-- ObjectProperty -->
<wasBorn>2001</wasBorn> <!-- DatatypeProperty -->
</Person>
```

Als Eigenschaften können sowohl vordefinierte Elemente von OWL und RDF(S) genutzt werden, als auch eigene definiert werden. Eine vordefinierte Eigenschaft ist die *subClassOf*. Sie wird genutzt, um Hierarchien zwischen Klassen aufzubauen. In dem folgenden Beispiel wird einem Mobiltelefon die Oberklasse Telefon zugewiesen:

```
<owl:Class rdf:ID="MobilePhone">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Phone"/>
  </rdfs:subClassOf>
</owl:Class>
```

Dies bedeutet, dass ein *Mobiltelefon* auch zu der Klasse der *Telefone* gehört und daher alle Eigenschaften von der Klasse *Telefon* erbt und diese gegebenenfalls erweitert.

Editoren wie Protégé unterstützen unter anderem die Nutzung und Darstellung der Klassen anhand dieser Eigenschaft. Über eine Hierarchie lässt sich gut eine *Taxonomie* aufbauen. Diese spezielle Form einer Ontologie lässt zwar keine Quer-Beziehung zwischen Klassen zu, ist aber für viele Anwendungsfälle ausreichend. Für den grundlegenden Ansatz der ontologiebasierten Suche dieser Arbeit ist eine solche Taxonomie ebenfalls ausreichend. Für weiterführende Arbeiten kann dieser Ansatz jedoch bei Bedarf erweitert werden.

[11] http://www.w3.org/2004/OWL/

Grundstruktur der verwendeten Ontologie

Die Grundstruktur der verwendeten Ontologie ist in Abbildung 6.9 dargestellt. In dem Rahmenwerk wird die semantische Beschreibung der Daten benötigt. Daher ist der Ursprungsknoten für alle Datentypen *Data*. Darunter gliedern sich die Beschreibung in *Subjekte*, *Eigenschaften* und *Beziehungen* auf. Die *Subjekte* beschreiben die Objekte auf die sich eine Information bezieht, die *Eigenschaft* umfasst die Zustände oder Informationen des Sensors und mittels *Beziehungen* lassen sich Relationen zwischen Objekten ableiten.

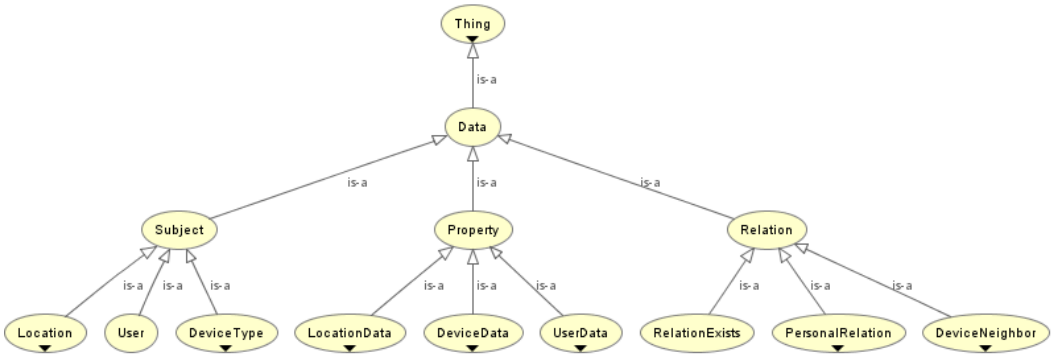


Abbildung 6.9: Grundstruktur der Ontologie

Die Abbildung 6.10 zeigt den Ausschnitt der *Subjekte* im Detail dargestellt. Hierbei lässt sich erkennen, dass in den Anwendungsszenarien bisher Benutzer, Orte und Endgeräte im Vordergrund standen. Diese Subjekte lassen sich jeweils weiter untergliedern. Wie bereits in Abschnitt 6.3.1 erläutert, handelt es sich bei dieser Ontologie nur um eine Grundstruktur, welche jederzeit um weitere Elemente ergänzt werden kann.

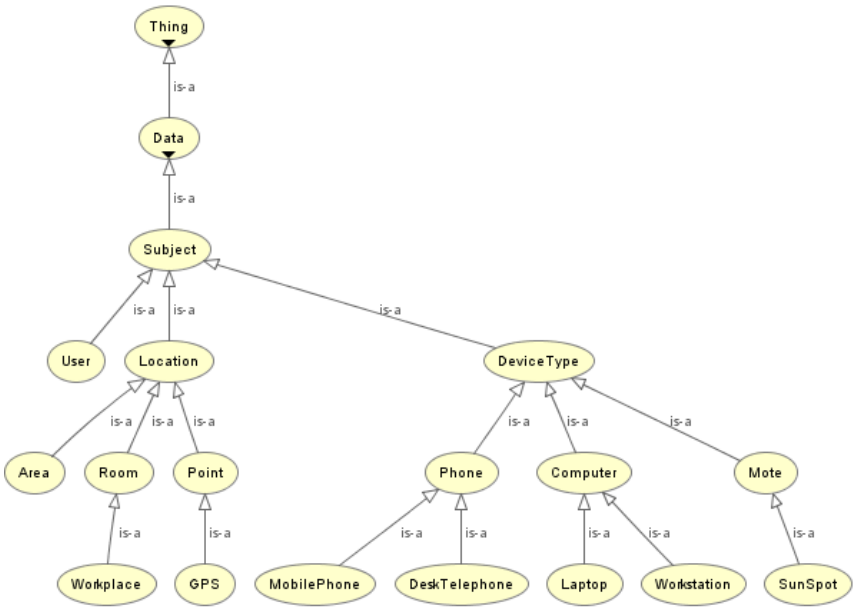


Abbildung 6.10: Ausschnitt aus der Ontologie

Erweiterung der Ontologie

Wie in Abschnitt 4.5.4 erläutert ist es notwendig, die Ontologie zu erweitern, falls neue unbekannte Sensortypen an das System angebunden werden sollen. Hierzu wurde der Ansatz des *Ankers* entwickelt und umgesetzt. Der Entwickler eines Sensors kann einen neuen Sensor in das System integrieren, ohne selbst die Ontologie bearbeiten zu müssen. Er hat die Möglichkeit einen neuen Sensortyp in die Ontologie einzuhängen, indem er eine Reihe von möglichen Anknüpfungspunkten (die Anker) vorgibt. Die Anker können direkt in der semantischen Annotation der Schnittstelle benannt werden:

```
/**
 * @return Property@DeviceData@Activity@PhoneActivity@SkypeActivity
 * @expirationTime 10000
 */
public String getSkypeStatus();
```

In dem dargestellten Beispiel wird ein zum Startzeitpunkt des Rahmenwerkes unbekannter *Skype-Sensor* angebunden. Hierbei wird der Anker vor den eigentlichen Sensortyp gestellt (es können mehrere Anknüpfungspunkte mit „@" angegeben werden). Ist der Skype-Sensor dem System noch nicht bekannt, so wird dieser an den Sensortyp *PhoneActivity* angehängt, indem eine Beziehung über die Eigenschaft *subClassOf* (siehe Abschnitt 6.3.1) hergestellt wird.

6.3.2 Suche

Eine Suche beinhaltet, neben den initialen Informationen (z.B. Name oder Nummer eines Teilnehmers), welche als Startpunkte für die Suche genutzt werden, auch die angestrebten Zielpunkte der Suche. Zielpunkte sind diejenigen Arten an Informationen, welche für eine Entscheidung innerhalb des Anwendungsfalls benötigt werden (z.B. die Aktivität oder den Aufenthaltsort eines Teilnehmers). Solche Zielpunkte können beispielsweise „alle Informationen über die verfügbaren Endgeräte" oder „Telefonnummern der benachbarten Personen im gleichen Raum" darstellen.

Wie in Abschnitt 4.6.2 erläutert wurde, erfolgt die Suche iterativ, d.h. je nach gefundener Information, können neue Informationsquellen relevant werden. So können beispielsweise über den ermittelten Aufenthaltsort auch die Informationsquellen, welche sich an diesem Ort befinden, relevant werden.

Die Werte der Parameter einer Suche über die Informationsquellen können je nach Anwendungsfall variieren. Was jedoch innerhalb eines Anwendungsfalls meist nicht variiert, ist der semantischen Typ der Parameter, oder die Qualitätsanforderungen an die Suche.

Definition von Interessen zur Vorbereitung von Suchanfragen

Wenn zuvor für einen Anwendungsfall bekannt ist, welche semantischen Typen als Start- und Zielpunkte für eine Suche genutzt werden und welche Qualitätsanforderungen von der Anwendung an die Suche gestellt werden, kann die Suche entsprechend vorbereitet werden.

Eine Anwendung kann eine Suchanfrage vorbereiten, indem sie Informationen bezüglich der Start- und Zielpunkte sowie der Qualitätsanforderungen in Form von einem *Interesse* beschreibt:

```
<Interest>
 <searchName>Search_for_PhoneNumbersOfPersonsInSameRoom</searchName>
 <searchesFor>PhoneNumber</searchesFor>
 <iteratesOver>Room,User</iteratesOver>
 <givenProperty>
   <string>User</string>
 </givenProperty>
 <timeout>500</timeout>
 <quality>80.0</quality>
</Interest>
```

Ein Interesse wird dazu genutzt, vorab sicher zu stellen, dass die QoS-Anforderungen einer Suche erfüllt werden können.

Das Beispiel zeigt die Definition des Interesses an benachbarten Personen im gleichen Raum. Als Startpunkt wird der Name eines Teilnehmers angegeben. Gesucht wird nach anderen Teilnehmern (*searchesFor*). Kann der Raum des angegebenen Teilnehmers ermittelt werden (*iteratesOver*), so soll nach weiteren Sensoren gesucht werden, welche wie in diesem Beispiel, die Telefonnummern der Teilnehmer im gleichen Raum bereitstellen. Zudem werden als Qualitätsmerkmale eine maximale Gesamtdauer der Suche und die gewünschten Aktualität der Ergebnisse festgelegt. Die Abbildung 6.11 zeigt, wie mit Interessen umgegangen wird.

Wenn mehrere Anwendungen dieselben Anforderungen haben, kann ein Interesse mehrfach adressiert und genutzt werden. Sobald ein Interesse angemeldet wurde, wird die Menge der für die Suche relevanten Sensoren bestimmt. Dies ermöglicht ebenfalls die Vorbereitung eines Suchbaums, welcher die Menge der Sensoren, die für

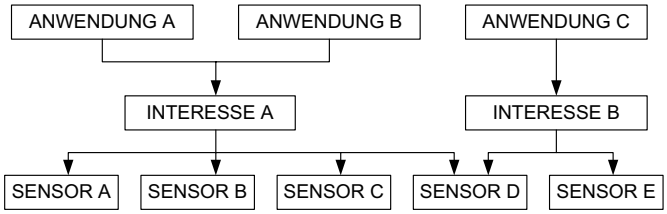


Abbildung 6.11: Anwendung der Definition von Interessen

jeweils eine der Iterationen in Frage kommt bestimmt (siehe auch Abschnitt 4.6.2). Dies reduziert den Aufwand zur Laufzeit der Suche. Ein Suchbaum, welcher anhand des zuvor beschriebenen Beispiels der Suche nach Telefonnummern von Personen im gleichen Raum erstellt wird, könnte wie in Abbildung 6.12 dargestellt aufgebaut sein.

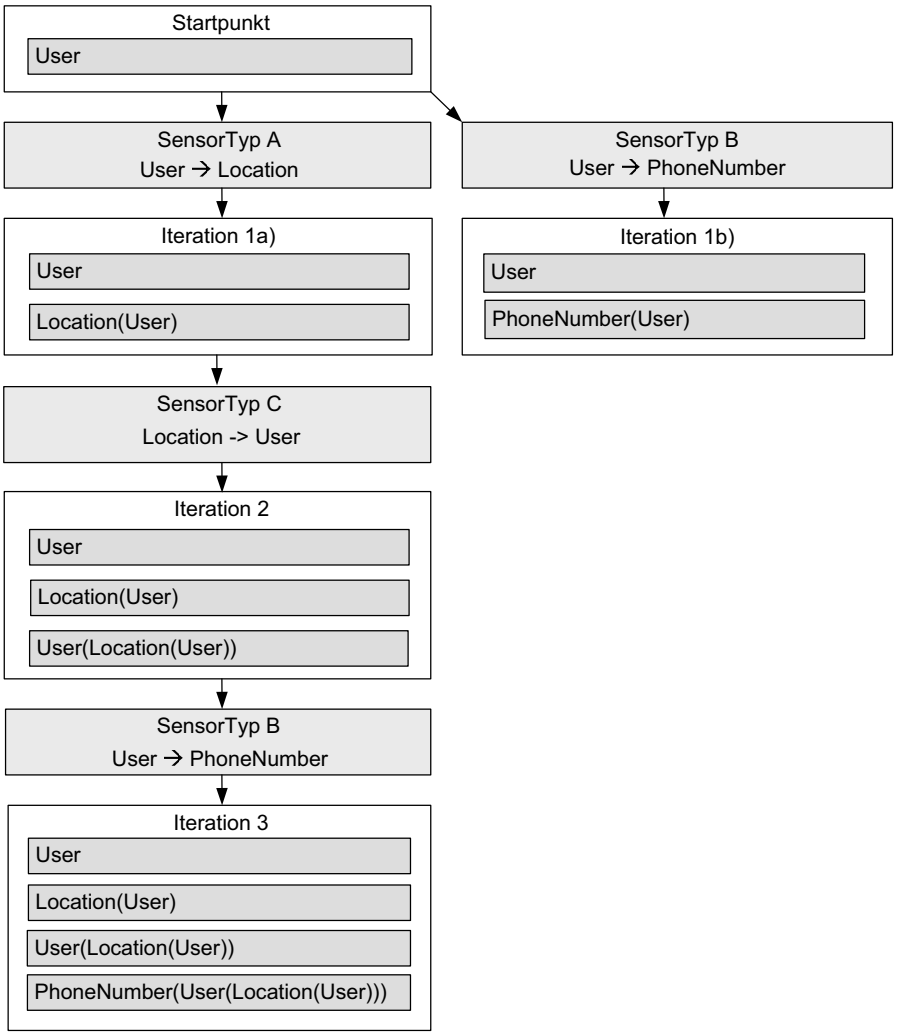


Abbildung 6.12: Beispiel für einen Suchbaum: Suche nach Telefonnummern von Personen im Raum

Die Suche erfolgt schließlich analog zum Suchbaum, wobei in jeder Iteration jeweils diejenigen Sensoren adressiert werden, welche zu dem entsprechenden Sensortyp zählen und zu denen eine Beziehung über die gesammelten Informationen hergestellt werden kann (siehe Abschnitt 4.6.1).

Qualität, Zeitlimit

Es gibt eine Reihe von Qualitätsmerkmalen von Sensordaten. Hierzu zählen die Genauigkeit, Zuverlässigkeit, Relevanz, Abdeckung, Aussagekraft und Aktualität. Im Rahmen dieser Arbeit wird zunächst die Aktualität von Sensordaten betrachtet. Der gewählte Ansatz erlaubt jedoch die Beschreibung und Nutzung weiterer Merkmale (durch entsprechende Erweiterungen der Sensorbeschreibung und der Anwendung geeigneter Metriken über die Qualität von Sensordaten). Die Aktualität wird anhand der Gültigkeit eines Sensorwertes berechnet, welche in der Beschreibung des Sensors über das Schlüsselwort *@expirationTime* gesetzt werden kann (siehe Abschnitt 6.2.2). Der *quality*-Wert, wie er durch das Interesse definiert wird, ist als Prozentwert zu verstehen. Üblicherweise hat ein aktuell erfasster Wert eine Aktualität von 100%. Wohingegen ein Wert, welcher gerade am Ende seiner Gültigkeitsdauer (*expirationTime*) ist, eine Aktualität von 0% aufweist. Werden keine Angaben gemacht, so wird ein linearer Abfall der Aktualität über den Zeitraum der Gültigkeitsdauer eines Sensorwertes angenommen.

Wie in Abbildung 6.13 dargestellt wird, kann durch die Definition einer Menge von Punkten der Verlauf der Aktualität eines Sensorwertes abhängig von der Gültigkeitsdauer beschrieben werden. Im Allgemeinen ist es dabei ausreichend, zwischen den Punkten linear zu interpolieren.

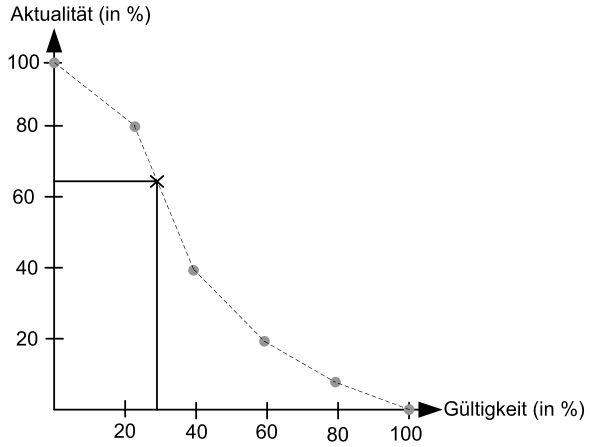


Abbildung 6.13: Berechnung der Aktualität

Anhand der Auswertung der Antwortzeiten von Sensoren kann bestimmt werden, ob es möglich ist die Informationen in der geforderten Qualität und innerhalb der vorgegebenen Maximaldauer zu erfassen. Ist dies nicht möglich, so werden *Prefetching*-Mechanismen innerhalb des integrierten Zwischenspeichers eingestellt, um kritische Informationsquellen vorab anzufragen und die Daten mit ausreichender Aktualität bereitzustellen.

Repräsentation der Suchergebnisse

Die Suchergebnisse werden analog zum Suchbaum abgespeichert. Jeder Knoten beschreibt den Wert über den gesucht wurde, und die jeweiligen Unterknoten entsprechen den Suchergebnissen dieser Iteration. Die Ergebnisse, welche als Grundlage für weitere Iterationen herangezogen wurden, enthalten entsprechend die dabei aufgefundenen Resultate. Über diese Vorgehensweise lässt sich anhand des Suchergebnisses die Relation der Ergebnisse zu dem Startpunkt der Suche bestimmen. Jeder Knoten auf dem Weg zwischen dem Startpunkt und dem Suchergebnis beschreibt eine Indirektion in der Beziehung zwischen den Objekten. Diese Repräsentation eignet sich gut, um die geforderte Darstellung der Suchergebnisse für die Anwendung auf das Lernverfahren zu generieren (siehe Abschnitt 6.4.1).

In jedem Knoten werden zunächst alle Informationen gespeichert, die für nachfolgende Operationen relevant sein könnten. Hierzu zählen:

- **Resultat:** Semantischer Datentyp und Wert.
- **Quelle:** Adresse und Typ des Sensors und Zeiger auf den genutzten Zwischenspeicher.
- **Verzögerung:** Benötigte Zeit zur Bearbeitung der Anfrage.

6.4 Kontextdienst

Der Kontextdienst setzt die in Kapitel 5 beschriebenen Funktionalitäten um.

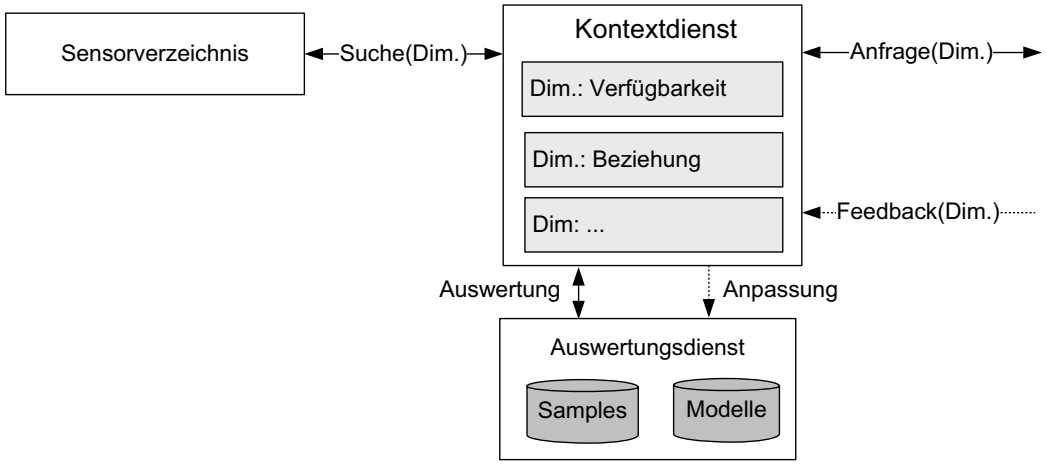


Abbildung 6.14: Umsetzung des Kontextdienstes

Von außen ist der Kontextdienst über zwei Schnittstellen ansprechbar. Eine Schnittstelle implementiert die Funktionen für die Anfrage zur Auswertung von Informationen. Eine weitere Schnittstelle stellt dem Nutzer die Feedback-Funktionalitäten zur Verfügung, mit welchen der Dienst gesteuert werden kann.

Es können mehrere Instanzen des Kontextdienstes gestartet werden, um für jeweils eine Kontextdimension angepasste Verhaltensweisen zu definieren. Jede Dimension stellt ihre eigene Suchanfrage an das Sensorverzeichnis, um jeweils nur die für die Dimension relevanten Sensoren abzufragen. In jeder Dimension ist auch die Art und Weise der Kontextbestimmung unterschiedlich. Der Beziehungskontext weist beispielsweise eine sehr lange Gültigkeit auf und kann eine Beziehung zwischen einer Menge von Teilnehmern im Voraus bestimmen und über einen längeren Zeitraum hinweg vorhalten, wohingegen die Verfügbarkeit erst zum Zeitpunkt des Anrufes ermittelt werden kann und der ermittelte Wert nur über eine relativ kurze Gültigkeit verfügt. Je nach Kontextdimension können gegebenenfalls auch unterschiedliche Modellarten und Lernverfahren benötigt werden.

Jeder Kontextdienst kann bestimmen, welcher Dienst zur Auswertung genutzt werden soll. Für diese Arbeit wurden die Funktionalitäten im Rahmen der adaptiven Auswertung aus Kapitel 5 durch Lernverfahren in Form des *Auswertungsdienstes* umgesetzt.

6.4.1 Auswertungsdienst

Der Auswertungsdienst wurde als funktionaler Bestandteil aus dem Kontextdienst ausgelagert, da er von allen Kontextdimensionen heraus gleichermaßen genutzt wird. Die Schnittstellen, welche der Auswertungsdienst implementiert, sind die Auswertung von Informationen, sowie die Anpassung des Modells.

Während der Kontextdienst die Aufgabe hat, zur Laufzeit den Kontext zu bestimmen und vorzuhalten, hat der Auswertungsdienst die Aufgabe eine persistente Speicherung der Trainingsinstanzen und der Modelle zu ermöglichen. Je nach Dimension und gegebenenfalls auch dem Namen des Teilnehmers werden entsprechende Mengen von Trainingsinstanzen oder Modelle geladen und verarbeitet.

Kapselung der Verfahren durch gemeinsames Datenformat

Die Anbindung verschiedener Verfahren kann durch ein gemeinsames Interface ermöglicht werden. Diese Schnittstelle muss die Daten in möglichst generischer Form darstellen, so dass alle Verfahren gleichermaßen genutzt werden können.

Die frei verfügbare Software *WEKA*[12], bietet eine Reihe von Werkzeugen und Verfahren im Bereich des Datamining. WEKA stellt eine Plattform zur Verfügung, welche es möglich macht, verschiedene funktionale Komponenten

[12] `http://www.cs.waikato.ac.nz/ml/weka/`

wie beispielsweise Lernverfahren zu kombinieren und auszutauschen. WEKA ermöglicht dies durch ein gemeinsames Datenformat, welches auf alle Komponenten sowohl für Eingabedaten als auch für die Ausgabedaten angewandt wird. Da dieser Ansatz gut zur Problemstellung dieser Arbeit passt, wurde eine ähnliche Vorgehensweise gewählt. Hierzu wurde der Ansatz eines gemeinsamen Datenformates übernommen. Zudem wurde der Ansatz um notwendige Komponenten, wie die Transformationen und die Repräsentation der Daten in dieses gemeinsame Datenformat, erweitert. Über diesen Ansatz ist es unter anderem möglich, Verfahren aus der WEKA-Plattform direkt zu integrieren.

Da WEKA zur Auswertung von Daten genutzt wird, wie sie in Datenbanken vorkommen, ist dieses Datenformat darauf ausgelegt mit zweidimensionalen Tabellen zu arbeiten. Hierzu erhält jeder Sensor eine eigene Spalte, während jede Trainingsinstanz in einer Zeile dargestellt wird. Bei dynamischen Mengen von Sensoren in den Suchergebnissen führt dies jedoch dazu, dass diese Tabelle entsprechend angepasst werden muss. Im Falle unbekannter Sensoren müssen neue Spalten integriert, sowie im Falle wegfallender Sensoren, durch leere Felder ergänzt werden. Dies führt jedoch zu einem Mehraufwand, der sich in der Kombination mit WEKA nicht vermeiden lässt. Die im Rahmen dieser Arbeit entwickelten Verfahren FLORA-MC und WAH wurden entsprechend auf das WEKA Datenformat angepasst und sind daher transparent gegen die WEKA eigenen Verfahren austauschbar.

Repräsentation der Daten

Die Lernalgorithmen, welche in dieser Arbeit betrachtet werden, sind für unterschiedliche Anwendungen konzipiert. Das Format der Daten auf denen diese Lernalgorithmen arbeiten, ist daher generisch gehalten. Die betrachteten Verfahren und Modelle, arbeiten auf Datensätzen mit Informationen in Form von Schlüssel-Wert (engl. *Key-Value*) $(x =< k, v >,\ \ k \in K, x \in X)$ Paaren. Bei den Instanzen kann zwischen Klassifikationsinstanzen $X_{klassifikation}$ und Trainingsinstanzen $X_{training}$ unterschieden werden.

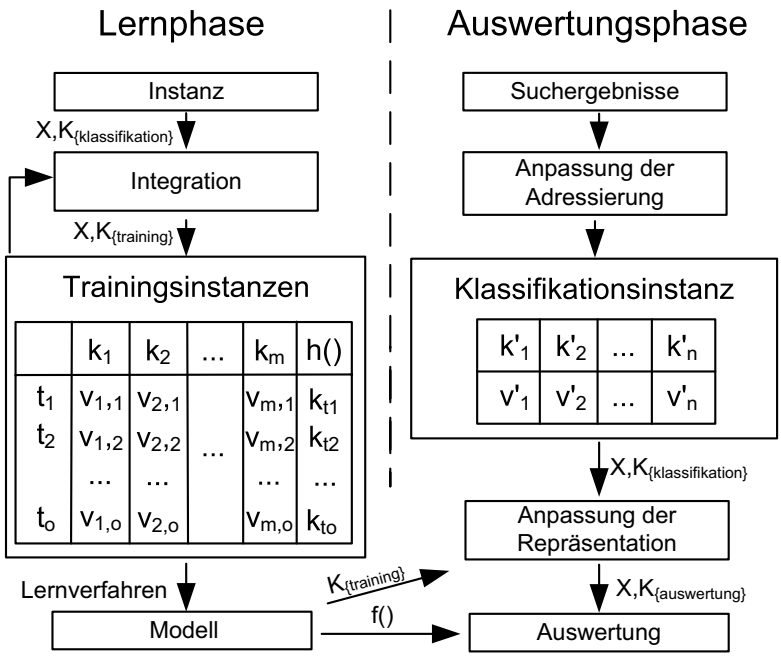


Abbildung 6.15: Aufgaben und Arbeitschritte der Datenrepräsentation

Datenrepräsentation zur Anpassung

Ein Datensatz mit den Trainingsinstanzen $\{X_{training}\}$ zur Generierung eines Modells kann in Form einer zweidimensionalen Tabelle beschrieben werden (wie in Abbildung 6.15 links dargestellt). Jede Zeile entspricht einer Instanz X. Im Falle des überwachten Lernens beinhaltet eine Zeile zudem noch ein spezielles Feld für das Feedback, welches zu der Instanz gegeben wurde $h(X)$. Die Spaltenüberschrift bezeichnet den Schlüssel und die Einträge in jeder Spalte beinhalten die Werte zu den jeweiligen Schlüsseln.

Zur Integration einer Instanz müssen die Mengen der Schlüssel der Trainingsinstanzen und der neuen Instanz gleich sein. Auf der einen Seite müssen hierzu die fehlenden Einträge in der neuen Instanz durch *leere Werte* ergänzt werden. Hat eine hinzuzufügende Instanz neue Schlüssel, welche in den Trainingsinstanzen bisher nicht enthalten waren, so muss die Tabelle auf der anderen Seite um weitere Spalten erweitert werden. Die Einträge in dieser Spalte werden bei allen anderen Zeilen ebenfalls mit *leeren Werten* gefüllt.

Datenrepräsentation zur Auswertung

In der Auswertungsphase müssen die Suchergebnisse angepasst werden, um sie geeignet auf ein Modell anwenden zu können (wie in Abbildung 6.15 rechts dargestellt).

Eine Klassifikationsinstanz, welche auf ein Modell angewandt werden soll $f(X_{Training})$, muss seine Informationen in Form von einer Liste solcher Tupel repräsentieren.

$$X_{Training} = \{x\}, \quad x_m = \{< k_1, v_{1,m} >, < k_2, v_{2,m} >, \ldots, < k_n, v_{n,m} >\} \tag{6.1}$$

Eine Trainingsinstanz $X_{Training}$ beinhaltet somit eine Menge von Schlüsseln $K_{Training} = \{k_1, k_2, \ldots, k_n\}$.

Ein Modell benötigt eine Klassifikationsinstanz, welche möglichst dieselbe Menge an Schlüsseln $K_{Klassifikation}$ aufweist, wie sie in den Trainingsinstanzen $K_{Training}$ enthalten ist. Ein Modell kann nur Informationen zur Auswertung nutzen, wenn deren Schlüssel bereits in den Traininginstanzen verwendet wurden.

$$f(X_{Auswertung}), \quad X_{Auswertung} = \{\ldots\} \forall x_n | k_n \in K_{Training} \tag{6.2}$$

Alle anderen Informationen werden nicht zur Auswertung herangezogen.

Auf der anderen Seite sollte im Idealfall jeder Schlüssel, welcher in den Trainingsinstanzen verwendet wurde, auch in der Klassifikationsinstanz enthalten sein. Die Menge aller Schlüssel, zu denen keine Information aus der Klassifikationsinstanz gefunden wurde beschreibt die Menge der *fehlenden Informationen*.

$$K_{Fehlend} = K_{Training} - K_{Auswertung} \tag{6.3}$$

Fehlende Informationen beeinträchtigen die Qualität der Entscheidung (QoD), wie in Abschnitt 5.3.1 erläutert und in Abschnitt 5.4.7 analysiert wurde.

Adressierung von Informationen

Wie in Abbildung 6.15 rechts oben dargestellt, müssen die Suchergebnisse in eine Klassifikationsinstanz überführt werden. Die Adressierung einer Information spielt an dieser Stelle eine entscheidende Rolle.

Die Schlüssel k der Einträge innerhalb einer Instanz können prinzipiell beliebig gewählt werden, müssen jedoch eindeutig sein. Die Zuordnung der Schlüssel der Klassifikationsinstanz muss jedoch mit den Schlüsseln der Trainingsinstanzen übereinstimmen, welche sich auf dieselbe Aussage oder Information beziehen.

Syntaktische Adressierung

Ein möglicher Ansatz würde darin bestehen, einen Sensor (dessen eindeutige ID) direkt auf einen Schlüssel abzubilden (wie in Abbildung 6.16 rechts oben dargestellt). Diese 1:1-Zuordnung würde jedoch dazu führen, dass jeder Sensor eine eigene Spalte in dem Datensatz der Traininginstanzen erhält. In manchen Anwendungsfällen wäre dies sogar ausreichend. In diesem Ansatz wären Sensoren jedoch nicht vergleichbar und damit die Aussagen des Modells nicht auf andere Situationen oder Kontextobjekte übertragbar. Die Aussagen des Modells wären abhängig von bestimmten Sensoren. Sobald beispielsweise der Nutzer an einen anderen Arbeitsplatz umzieht, werden andere Sensoren aus der Umgebung relevant. Dies würde dazu führen, dass die Suchergebnisse neue Sensoren beinhalten würden und das zuvor trainierte Modell diese Schlüssel nicht verarbeiten kann.

Semantische Adressierung

Die semantische Adressierung eines Sensors führt dazu, dass die Aussagen des Modells allgemein gültig sind. Die Adressierung eines Sensors muss dabei dessen semantische Zuordnung zum Kontextobjekt wiedergeben, also beispielsweise wiedergeben, ob sich der Sensor direkt auf den Nutzer bezieht oder ob er sich auf denselben Raum bezieht, in dem sich der Nutzer befindet oder auf den Kollegen, der sich im selben Raum aufhält. Die Anforde-

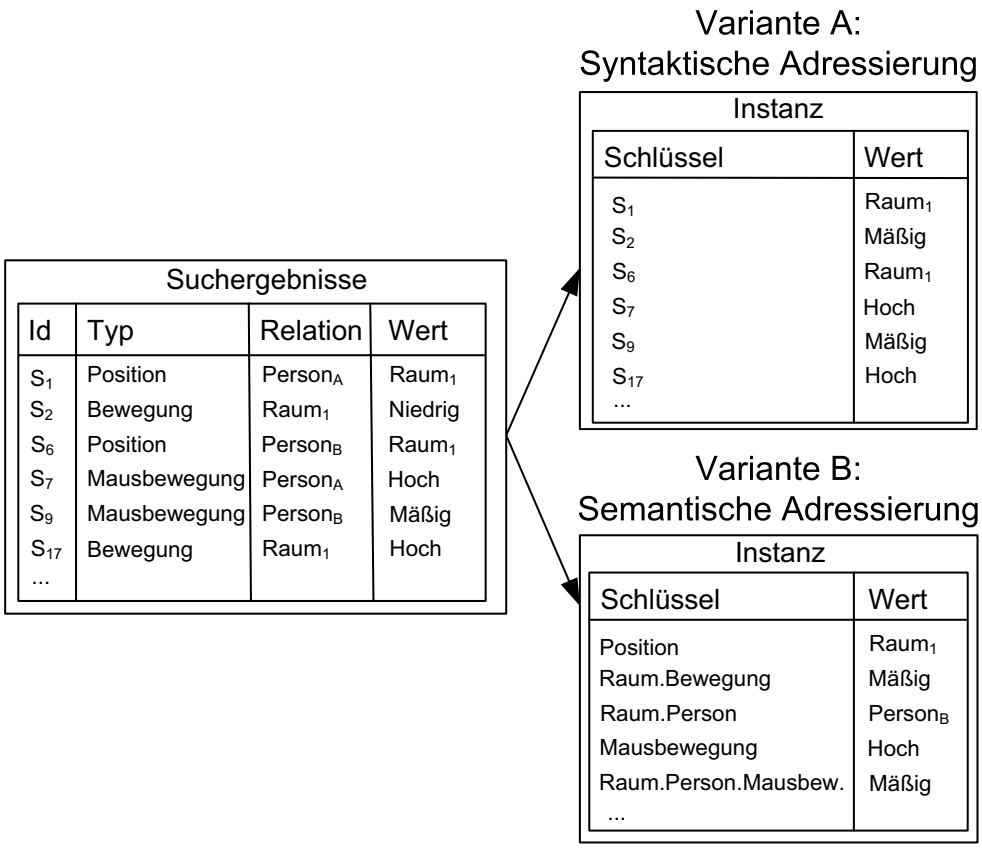


Abbildung 6.16: Adressierung von Sensorinformationen in Instanzen und Modellen

rung seitens der Datenrepräsentation durch Key-Value-Paare besteht jedoch darin, diese Zuordnung durch einen einfachen, textuellen Schlüssel darzustellen.

Der Ansatz, welcher für diese Arbeit entwickelt wurde, setzt auf dem Prinzip von URLs auf und wird in Abbildung 6.16 rechts unten skizziert. Der semantische Typ des Sensors wird auf die *Top-Level-Domäne* abgebildet. Die *Subdomänen* beschreiben die Beziehung zwischen dem Sensor und dem Objekt. Jede Indirektion der Beziehung zwischen dem Sensor und dem Kontextobjekt spiegelt sich in einer Subdomäne wider. Bezieht sich der Sensor direkt auf das Kontextobjekt, so werden keine Subdomänen benötigt.

In den Suchergebnissen aus dem Beispiel aus Abbildung 6.16 ist die Aussage enthalten, dass sich Person A und Person B im gleichen Raum aufhalten. Bei der iterativen, ontologiebasierten Suche aus Abschnitt 4.6.2 kann dieser Zusammenhang genutzt werden, um über diese semantische Beziehung zu iterieren, um weitere Informationen über die Person B zu erhalten. Bei der semantischen Adressierung wird dieser Zusammenhang genutzt und geeignet dargestellt.

Eine Herausforderung entsteht jedoch in Situationen, in der mehrere Informationen dieselbe semantische Zuordnung zum Kontextobjekt aufweisen. Da die Schlüssel per Definition eindeutig sein sollen, muss in solchen Fällen entschieden werden, wie diese Informationen geeignet miteinander verarbeitet werden können. Der gewählte Ansatz sieht vor, diese Verarbeitung anhand der semantischen Beschreibung der Typen durchzuführen. In dem angeführten Beispiel, existieren zwei Bewegungssensoren, welche die gleiche Relation zum Kontextobjekt aufweisen und entsprechend den gleichen Schlüssel haben. Aussagen, wie die der Bewegungssensoren, welche sich kombinieren lassen, können auf eine gemeinsame Aussage in der Instanz abgebildet werden. Andere Aussagen, bei denen dies nicht der Fall ist, müssen gegebenenfalls durch eine Integration der ID in den Schlüssel eindeutig gestaltet werden.

Dieses gesamte Vorgehen führt dazu, dass zumindest ein Großteil der Aussagen der Traininginstanzen und letztlich des Modells auf andere Situationen übertragen oder sogar benutzerübergreifend angewendet werden kann.

6.4.2 Feedback Schnittstellen

Das System erbringt seine Dienste weitgehend autonom. Neben der vereinfachten Prozedur zur Installation der Sensor-Dienste braucht ein Teilnehmer nur noch Feedback zu geben, um das Verhalten des Systems an seine Wünsche anzupassen. Für jede Form von Feedback (siehe Abschnitt 3.2.4) existieren andere Schnittstellen. Für *implizites* Feedback muss die Möglichkeit geschaffen werden, dass der Nutzer direkt aus einer Situation heraus Feedback geben kann:

- **Feedback-Tray:** Solange der Teilnehmer am Rechner sitzt kann er hierfür das *Feedback-Tray* nutzen. Dieses Programm wurde entwickelt, um als Element in der Symbolleiste Auskunft über die aktuelle Situation zu geben (mittels eines Symbols, welches zur Kontextklasse zugehörig ist) und direkt aus der Liste der verfügbaren Kontextklassen die aktuell gültige Klasse auszuwählen.
- **Portal, Kontextdienst:** Das Portal des Kontextdienstes lässt sich im Browser als Webseite öffnen. Über das Portal kann neben einigen anderen Funktionen auch die aktuell gesetzte Kontextklasse eingesehen und modifiziert werden.
- **Kommunikationsdienst:** Eine weitere Form, Feedback zu geben, wurde in Form eines Kommunikationsdienstes geschaffen. Vergleichbar zu einem Anrufbeantworterdienst kann ein Teilnehmer über eine dedizierte Nummer seine aktuelle Kontextklasse angesagt bekommen. Anschließend hat er die Möglichkeit, die Kontextklasse über ein interaktives Menü und der Tastatur seines Telefons zu setzen.
- **Eingebettet in einer Armbanduhr:** Um zu jedem Zeitpunkt die Möglichkeit zu bieten Feedback zu geben, wurden verstärkt Endgeräte adressiert, welche die Nutzer häufig bei sich tragen, sowie möglichst einfach zu handhaben und zudem unauffällig in der Bedienung sind (beispielsweise zur Anwendung während Konferenzen). In diesem Rahmen wurde eine Bluetooth-fähige Armbanduhr dahingehend erweitert, den aktuellen Zustand anzuzeigen und die gewünschte Kontextklasse zu wählen.

Für *explizites* Feedback muss der Nutzer die Möglichkeit haben, Einblick auf die zurückliegenden Situationen zu bekommen, um diese zu einem späteren Zeitpunkt interpretieren und bewerten zu können. Zu diesem Zweck wurden Programme entwickelt, welche die zur Auswertung herangezogenen Informationen in einer Datenbank ablegen und dem Nutzer geeignet darstellen. Eines dieser Programme wird später in Abschnitt 6.5.3 erläutert. Ein weiteres Programm mit der Bezeichnung *Feedback Interface* wurde in Zusammenarbeit mit der Arbeit von Zipfel und Kropff entwickelt [Zip09]. Dieses Programm ermöglicht zudem die Betrachtung des zeitlichen Verlaufs der Kontextklassen und der zugrunde liegenden Informationen (siehe Abbildung 6.17). Dieses Vorgehen ermöglicht das Sammeln von Daten über einen langen Zeitraum hinweg. Die in solchen Langzeittests gesammelten Daten können später für verschiedene Kontextdimensionen herangezogen und bewertet werden.

6.5 Nutzung im Rahmen des Kommunikations-Szenarios

Das Rahmenwerk ermöglicht die Bestimmung und Nutzung des Kontextes von Nutzern. Andere Anwendungen können nun das Rahmenwerk anfragen, um den Kontext zu erfragen und anschließend im Rahmen der Anwendung zu nutzen.

Zur Kontextnutzung innerhalb eines Kommunikationssystems muss der Kontext als Parameter mit in den Verarbeitungsprozess für Kommunikationsanfragen einfließen. Schnittstellen für nutzergenerierte Kommunikationsdienste (siehe Abschnitt 2.4.5) oder Skriptsprachen zur Diensteerstellung bieten die Möglichkeit, den Ablauf bei dem Verarbeitungsprozess zu beeinflussen.

Die Schnittstelle zwischen dem Telefonie-Server und der Kontextbestimmung muss derart definiert sein, dass Informationen über den Kontext der Anfrage zur Verfügung stehen (siehe Abschnitt 3.1.2). Zur Bestimmung des Kontexts einer Anfrage sind hierzu erforderlich:

- Der Adressat einer Anfrage ist zur Bestimmung der Verfügbarkeit, der verfügbaren Endgeräte und der Lokation des angefragten Teilnehmers notwendig.
- Zur Bestimmung der Relation zwischen Teilnehmern wird eine Identifizierung des anfragenden und des angefragten Teilnehmers benötigt.
- Zur Bestimmung einer geeigneten Kommunikationsform sind die Angaben über den eingehenden Kommunikationskanal relevant (textuelle oder sprachliche Form der Anfrage, Signalisierungsprotokoll, usw.). Ebenso sind weitere Informationen über die Fähigkeiten des Kommunikationssystems relevant (z.B. Gateway-Funktionalitäten, Anrufbeantworter, Ansagen).

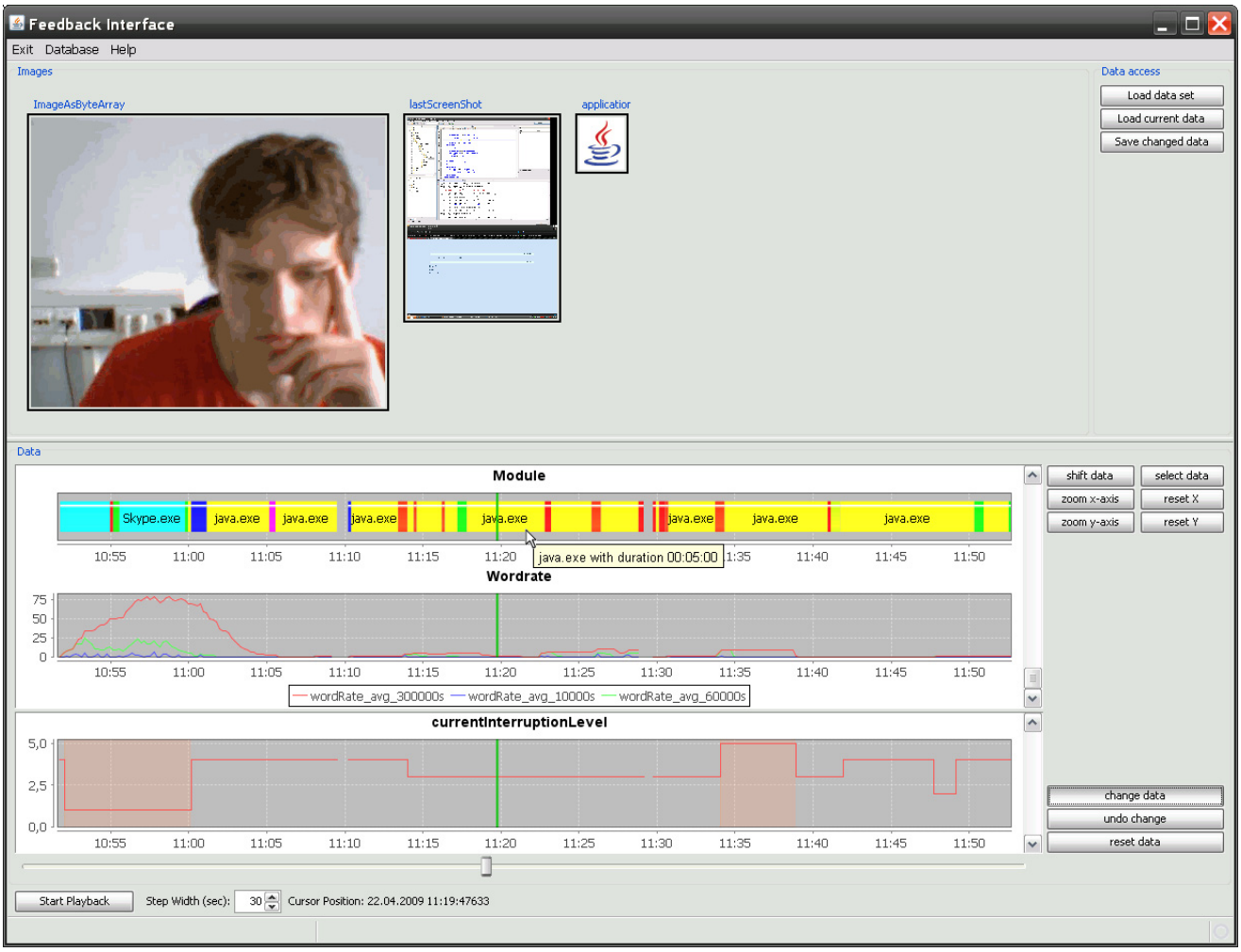


Abbildung 6.17: Benutzeroberfläche für explizites Feedback

- Weitere Merkmale der Anfrage können ebenfalls relevant sein und gegebenenfalls als Sensoren mit in den Entscheidungsprozess einfließen. Hierzu gehören beispielsweise Informationen über stattgefundene Weiterleitungen, Einstellungen am Endgerät (Rufton-Lautstärke) oder ob der Anruf aus dem internen Rufnummernblock der Telefonanlage stammt.

Auf der anderen Seite muss der Kommunikationsdienst die Möglichkeit haben, die Anfrage zu verarbeiten und gegebenenfalls weiterführende Schritte einzuleiten. Der Funktionsumfang der Schnittstelle beschreibt somit die Möglichkeiten des Kommunikationsdienstes:

- Die *Anrufweiterleitung oder -umleitung* gehört zu den grundlegenden Fähigkeiten in der Anrufverarbeitung. Hierzu gehört auch das Abweisen von Anrufen.
- Der Umfang der Möglichkeiten bei der *Vermittlung* zwischen verschiedenen Kommunikationsformen hängt stark von den Fähigkeiten des Kommunikationssystems ab und bietet die Möglichkeit, Anfragen auf unterschiedlichen Kommunikationsmedien miteinander zu vereinen.
- Die *Notifikation* über eingehende oder eingegangene Anfragen machen die Anbindung des Kommunikationssystems an eine asynchrone Form der Kommunikation notwendig (z.B. E-Mail oder SMS).
- Weitere Funktionalitäten, wie das Abspielen von automatisch generierten *Ansagen oder Interaktionen* mit dem Nutzer, können gerade in Situationen sinnvoll sein, in denen keine eindeutige Entscheidung getroffen werden kann.
- Die *Initiierung* von Anrufen kann gerade bei der Notifikation über stattgefundene Ereignisse (z.B. den Wechsel des gewünschten Gesprächspartners in einen geeigneten Kontext zum Aufbau einer Verbindung) genutzt werden.

Spezifikationen für Schnittstellen zu Kommunikationsdiensten

Je nach Kommunikationssystem existieren Schnittstellen, sogenannte *Application Programming Interfaces* (API), um nutzergenerierte Kommunikationsdienste zu definieren und zu integrieren. Im Bereich der Telefonie sind dies unter anderem:

Parlay /OSA
Als ein Beispiel für eine API, welche es ermöglicht komplexere Entscheidungen zu definieren und dabei auch externe Informationsquellen heranziehen kann, sei an dieser Stelle Parlay/OSA[13] genannt. Über Parlay/OSA kann sowohl aus dem Kommunikationssystem heraus auf Dienste aus dem Internet/Intranet zurückgegriffen werden, als auch umgekehrt. Parlay/OSA bietet eine Reihe von Funktionsmodulen, die als *Service Capability Feature* (SCF) bezeichnet werden. Mit jedem SCF wird ein netzspezifischer Dienst zur Verfügung gestellt. Dazu gehören z.B. die Bestimmung der Lokation von Teilnehmern in Mobilfunknetzen (beispielsweise anhand der aktuellen Funk-Zelle in der sich der Nutzer befindet), Auf- und Abbau von Verbindungen, die Abfrage, ob das Mobiltelefon eingeschaltet ist, Abspielen von Ansagen, Versenden von SMS oder die Abfrage der Fähigkeiten des Endgerätes.

JAIN
Die *Java API for Integrated Networks*[14] bietet eine Reihe von *JAIN APIs* zur Anbindung von Kommunikationsdiensten an die verschiedenen Kommunikationssysteme (ISDN, SIP, H.323, MGCP, usw.) an. Auf der Seite der Kommunikationsdienste besteht somit eine einheitliche Schnittstelle zur Verarbeitung von Anfragen in unterschiedlichen Kommunikationssystemen.

SIP-CGI
Das *Common Gateway Interface* (CGI) wird häufig genutzt, um (durch Aufruf einer Webseite ausgelöst) ausführbare Programme auf einem Webserver zu starten und die Übergabe von Parametern zu ermöglichen. SIP-CGI [LSR01] stellt eine spezielle Form von CGI dar und wurde für die Aufgaben zur Verarbeitung von Anfragen in dem *Session Initiation Protokoll* (SIP) entworfen.

AGI
Das *Asterisk Gateway Interface* (AGI) ist eine Schnittstelle, welche von *Asterisk* (softwarebasiertes Kommunikationssystem, siehe Abschnitt 6.5.2) angeboten wird. Analog zu SIP-CGI kann mittels AGI die Verarbeitung einer Kommunikationsanfrage innerhalb eines externen Programms durchgeführt werden. Das Programm bekommt dazu alle Parameter der Kommunikationsanfragen übermittelt und kann zur Verarbeitung auf einen Großteil der Funktionen der Telefoniesoftware zurückgreifen. Zu AGI existieren zwei interessante Erweiterungen: *Fast-AGI* und *Asterisk-Java*. Statt eine ausführbare Datei aufzurufen, nutzt Fast-AGI Socketkommunikation, um mit einem bereits im Hintergrund laufenden Programm zu kommunizieren. Asterisk-Java bildet die Funktionalitäten der Fast-AGI Schnittstelle auf Java-Objekten ab.

SIP-Servlets
Ein *Servlet* ist vergleichbar mit einem Dienst. Servlets werden jedoch im Rahmen der Verarbeitung von Anfragen von Webseiten genutzt und erhalten daher Zugriff auf die Parameter des Aufrufs einer Webseite und die Umgebungsvariablen des Webservers. SIP-Servlets bauen auf Java-Servlets[15] auf und erweitern deren Ansatz. Ein SIP-Servlet erhält dazu Zugriff auf die Parameter der Kommunikationsanfrage. SIP-Servlets sind jedoch von sich aus in der Lage, Anrufe zu initiieren und können zudem in einer geschützten Umgebung (*Sandbox*) ausgeführt werden.

Skriptsprachen zur Diensteerstellung

Statt externe Dienste zu integrieren, kann der Verarbeitungsprozess einer Kommunikationsanfrage auch durch Skriptsprachen gesteuert werden. Skriptsprachen haben im Vergleich zu externen Kommunikationsdiensten den Vorteil, dass sie eine sichere Umgebung bieten und leichter zu erstellen sind. Daher kann dem Teilnehmer selbst

[13] `http:www.parlay.org`
[14] `http://java.sun.com/products/jain/`
[15] JSR 116: SIP-Servlet API

	Parlay, OSA	JAIN	SIP-CGI	AGI	SIP Servlets	CPL, SCML	VoiceXML, CCXML
Methode	API	API	API	API	API	Skript	Skript
Funktionsumfang	umfassend	umfassend	mittelmäßig	umfassend	mittelmäßig	eingeschränkt	eingeschränkt
Plattform	C++, Java, Perl	Java	*	*	Java	XML	XML
Protokolle	*	*	SIP	*	SIP	*	*
Anforderungen	hoch	hoch	hoch	hoch	hoch	niedrig	niedrig
GUI für Entwicklung	nein	nein	nein	nein	nein	ja	ja
Gefährdung	hoch	hoch	hoch	hoch	hoch	niedrig	niedrig

Tabelle 6.2: Vergleich verschiedener Diensterstellungsmethoden für Kommunikationsdienste

die Möglichkeit gegeben werden, Skripte zu definieren. Skriptsprachen sind jedoch meist passiv. Sie werden also erst durch einen eingehenden Anruf ausgelöst und können daher selbst keine Anrufe initiieren.

CPL/SCML

Die beiden Skriptsprachen *Call Processing Language* (CPL) [LS04] und die *Service Control Markup Language* (SCML) [BJ02] sind sich relativ ähnlich. Beide Skriptsprachen basieren auf XML und verfügen nur über einen begrenzten Funktionsumfang, welcher allgemein gehalten wurde, um unabhängig vom Signalisierungsprotokoll zu sein. Mit Hilfe dieser Skriptsprachen können Teilnehmer die Bearbeitung eingehender und ausgehender Anfragen definieren. SCML kann im Gegensatz zu den anderen Ansätzen auch genutzt werden, um Anrufe zu initiieren.

VoiceXML/CCXML

VoiceXML ist eine Skriptsprache zur Definition von sprachgesteuerten Interaktionen zwischen Teilnehmern und dem System. Mit Hilfe von VoiceXML lassen sich Dialoge und Menüstrukturen definieren. Neben den implizit vorhandenen Funktionalitäten wie dem Abspielen von Ansagen, existieren auch grundlegende Funktionen zur Steuerung von Kommunikationssystemen, wie der Verbindungsauf- und Abbau. Die *Call Control eXtensible Markup Language* (CCXML) wurde entwickelt, um VoiceXML zu ergänzen. Beide sind jedoch eigenständige Skriptsprachen. CCXML ergänzt den Funktionsumfang von VoiceXML um Methoden, wie beispielsweise die Weiterleitung einer Anfrage an mehrere Personen, die Initiierung von Anrufen oder Konferenzen. Einen ähnlichen Ansatz liefert die Erweiterung der XML-basierten Skriptsprache VoxML wie sie in der Arbeit von Guedhami, Klein und Kellerer vorgestellt wird [GKK00].

Vergleich der Ansätze

Die Ansätze zur Integration haben eine Reihe unterschiedlicher Eigenschaften. In der Tabelle 6.2 werden die zuvor angeführten Ansätze miteinander verglichen. Neben dem Funktionsumfang der Schnittstelle und den anwendbaren Plattformen und Protokollen sind auch weitere Aspekte von Interesse. Skriptbasierte Ansätze stellen geringere *Anforderungen* an den Entwickler, als die Erzeugung von ausführbaren Programmen (wie beispielsweise bei CGI basierten Ansätzen). Eine grafische Unterstützung mittels einer *GUI für die Entwicklung* eines Kommunikationsdienstes reduziert den Aufwand zusätzlich. Auf der einen Seite haben skriptbasierte Ansätze einen geringeren Funktionsumfang als API-basierte Ansätze, stellen jedoch für die Ausführung auf einem Kommunikationssystem eine geringere *Gefährdung* dar, da die Integration von schadhaften Codes vermieden wird.

Für die Umsetzung ist es sinnvoll einen Ansatz zu wählen, welcher einen möglichst großen Funktionsumfang, sowie möglichst wenig Beschränkungen hinsichtlich der anwendbaren Plattformen und Protokolle aufweist. Im Rahmen dieser Arbeit wurde unter anderem Asterisk als Kommunikationssystem herangezogen. Daher wurde als Schnittstelle AGI ausgewählt und integriert (siehe Abschnitt 6.5.2).

6.5.1 Integration in ein Kommunikationssystem

Unter Berücksichtigung der zuvor genannten Aspekte über geeignete Schnittstellen zur Kontextnutzung im Rahmen der Verarbeitung von Kommunikationsanfragen, gilt es nun ein geeignetes Kommunikationssystem auszuwählen.

Die Grundlage für diese Arbeit wurde im Rahmen meiner Diplomarbeit gelegt [Sch04]. Dort bestand die Aufgabe darin, die Anrufverarbeitung anhand von einzelnen, extern erfassten Kontextinformationen zu beeinflussen. Dies sollte im Rahmen der Skriptsprache CPL möglich gemacht werden, welche bereits Bestandteil des Kommunikationssystems VOCAL[16] war. Durch Integration neuer Funktionen in CPL war es möglich externe Informationen einzubinden. Zu Beginn dieser Arbeit habe ich versucht Editoren für CPL möglichst einfach (mittels grafischer Editoren oder der Darstellung der Regeln im Klartext) handhabbar zu machen. Da VOCAL seit 2003 nicht mehr weiter entwickelt wurde, bestand ein weiterer Schritt in der Integration von CPL in den *Sip Express Router* (SER)[17]. Der Funktionsumfang von SER ist jedoch stark auf Routing-Funktionen innerhalb des Signalisierungsprotokolls SIP eingeschränkt. Es zeigte sich jedoch, dass dieses manuelle Vorgehen und die Nutzung von CPL (auch mittels grafischer Editoren) für normale Anwender zu komplex oder zu aufwendig waren und daher dieser Ansatz nicht erfolgversprechend schien.

6.5.2 Integration in Asterisk

Asterisk[18] wird häufig als softwarebasierte Telefonanlage bezeichnet. Asterisk ist selbst nicht auf ein Protokoll festgelegt. Der Kern von Asterisk verarbeitet Kommunikationsanfragen auf generische Art. Per Plugin können alle gängigen Protokolle und Systeme angebunden und zwischen ihnen vermittelt werden. Weitere Plugins bieten ein umfassendes Angebot an Funktionen wie Anrufbeantworter, Call-Center Dienste, Spracherkennung, Konferenzgespräche und vieles mehr.

Asterisk verfügt über eine eigene Skriptsprache zur Definition der Anrufverarbeitung, den sogenannten *Dialplan*. Der Dialplan wird im Gegensatz zu Ansätzen wie CPL nicht vom Teilnehmer selbst, sondern vom Administrator der Telefonanlage festgelegt. Die Möglichkeiten des Dialplans umfassen alle Funktionen des Systems. Das AGI-Interface ermöglicht die Integration eigener Dienste in die Verarbeitung von Anfragen.

Die Nutzung des Kontextes als Parameter innerhalb des Dialplans kann wie folgt durchgeführt werden:

```
exten => _X.,1,Set(callee=${EXTEN:3})
exten => _X.,2,Agi(agi://10.0.0.1/presence.agi)
exten => _X.,3,GotoIf($["${presence}" = "available"]?5)
exten => _X.,4,Hangup
exten => _X.,5,GOTO(dialout-sip,${callee},1)
```

Der Aufruf von *presence.agi* startet eine Abfrage der Verfügbarkeit des angerufenen Teilnehmers. Das Ergebnis der Anfrage wird dann in Form des Parameters *presence* zurückgegeben, welcher dann im folgenden Teil des Dialplans verwendet werden kann. In dem dargestellten Fall wird der Anruf abgewiesen, sofern der gewünschte Gesprächsteilnehmer nicht erreicht werden möchte, was ein eventuell störendes Klingeln des Apparates vermeidet.

Diese Form der Nutzung ist jedoch nur eine einfache Variante. Die Kontrolle über die Verarbeitung der Anfrage kann vollständig auf das Programm, welches durch AGI aufgerufen wird übertragen werden. Der Aufruf dazu ist vergleichbar mit den ersten beiden Zeilen der zuvor dargestellten Variante. Das Programm, welches aufgerufen wurde, kann über die AGI Schnittstelle auf einen Großteil aller Funktionen von Asterisk zurückgreifen.

Es wurde ein weiterer Dienst entwickelt, welcher einen Fast-AGI Server bereitstellt, über den Anfragen von Asterisk an das Rahmenwerk gestellt werden können. Durch diesen Ansatz wird der Funktionsumfang von Asterisk um kontextberücksichtigende Elemente erweitert. Im Rahmen der Anbindung dieses *Asterisk-Dienstes* wurden neben der zuvor dargestellten Nutzung der Verfügbarkeit als Parameter im Dialplan eine Reihe weiterer Funktionen umgesetzt:

- **Abfrage des Kontextes:** Diese Methode des Asterisk-Dienstes erlaubt die Abfrage des Kontextes (beispielsweise der Verfügbarkeit) eines Teilnehmers. Die vom Kontextdienst ermittelte Klasse dieses Kontextes wird dem Anrufer in Form einer automatisch generierten Ansage mitgeteilt.

[16] Informationen zu VOCAL unter `http://www.voip-info.org/wiki/view/VOCAL`
[17] `http://www.iptel.org/ser/`
[18] `http://www.asterisk.org/`

- **Feedback:** Jeder Teilnehmer kann (vergleichbar zu einem Anrufbeantworter) eine Nummer wählen unter der er sein Verfügbarkeitslevel mitgeteilt bekommt und er anschließend die Möglichkeit hat (durch ein interaktives Menü) den Status zu verändern, beziehungsweise Feedback zu geben.
- **Kontextabhängige Auswahl von Aktionen:** Um in der Anrufverarbeitung mehrere verschiedene Kontextdimensionen zu berücksichtigen, bedarf es einer weiteren Entscheidung. Es muss entschieden werden welche Aktion im Falle einer Kombination von zutreffenden Kontextklassen wie beispielsweise { *Standort: Zu Hause; Beziehung: Kollege; Verfügbarkeit: Erreichbar* } ausgewählt wird. Hierzu bietet der Asterisk-Dienst über das Portal die Möglichkeit, Regeln über diese Zusammenhänge und die auszuführende Aktion zu definieren. Diese Einstellungsmöglichkeit in Abhängigkeit zur Relation zum Anrufer (*Unbekannt/Bekannt*) und zum eigenen Verfügbarkeits-Zustand (*Erreichbar/Beschäftig/Abwesend*) ist in Abbildung 6.18 dargestellt. In dem Beispiel werden zudem die Details für den Fall {*Unbekannt*, *Verfügbar*} dargestellt. In diesem Fall wird eine Durchleitung des Anrufes zum Telefon durchgeführt.
  Da viele der Zusammenhänge auf unterschiedliche Nutzer übertragbar sind, kann ein Satz von Basis-Regeln genutzt werden, die von jedem Nutzer bei Bedarf individuell angepasst werden können.

6.5.3 Integration in HiPath Telefonanlage

Im Zuge einer Kooperation mit Siemens wurde das Rahmenwerk ebenfalls in die Siemens Telefonanlage *HiPath* integriert. Hierbei bestand die Herausforderung der Integration des Systems in eine große bestehende Architektur. In dem HiPath System existiert ein Dienst, welcher dem Nutzer statische Regeln zur Anrufverarbeitung zu definieren erlaubt (siehe Abschnitte 1.2 und 5.1.1). Dieser Ansatz wurde dahingehend erweitert, die Profile, welche jeweils eine Menge dieser Regeln umfassen, dynamisch anhand des aktuellen Kontextes auszuwählen. Ein neuer Dienst, welcher diese Erweiterung implementiert, wird durch einen Anruf aktiviert, welcher in der Telefonanlage eintrifft. In einem nächsten Schritt greift dieser Dienst auf das Rahmenwerk ContextFramework.KOM zurück, um den aktuellen Kontext zu bestimmen und anhand des Klassifikationsergebnisses eine Auswahl des Profils durchzuführen. Das ausgewählte Profil wird schließlich zur weiteren Anrufverarbeitung herangezogen.

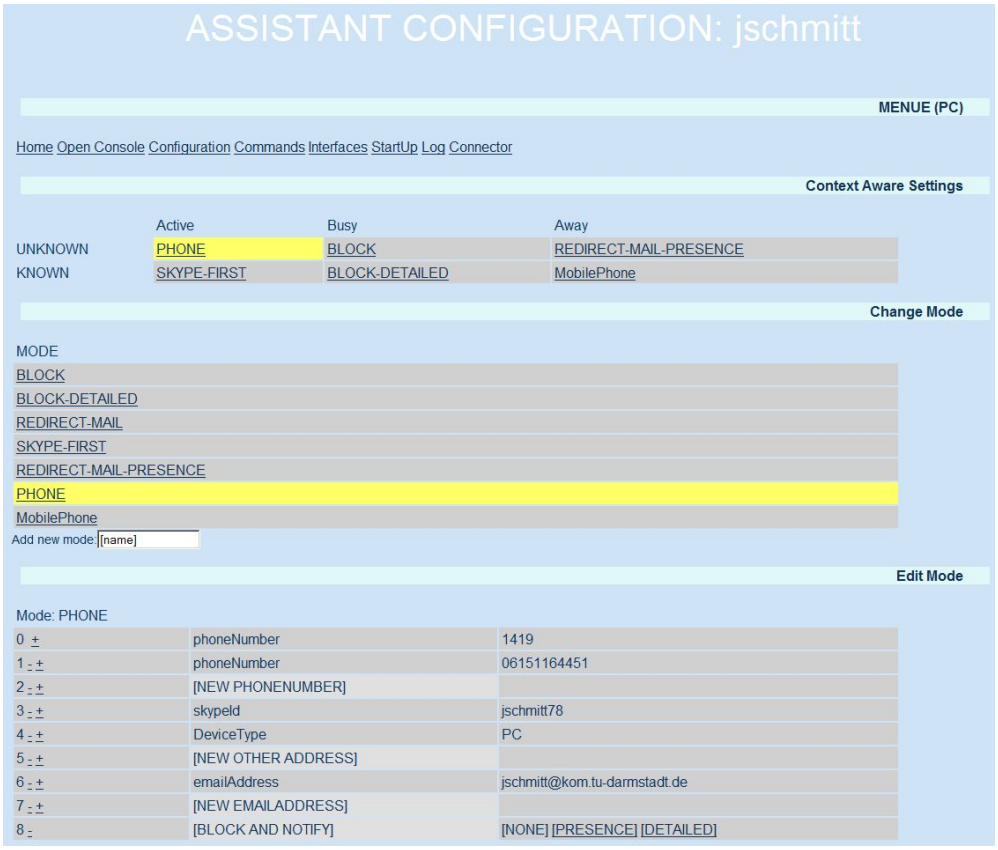
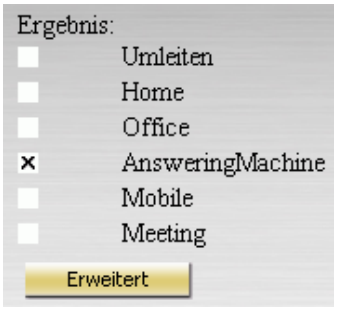


Abbildung 6.18: Kontextabhängige Auswahl von Aktionen über das Portal vom ContextFramework.KOM

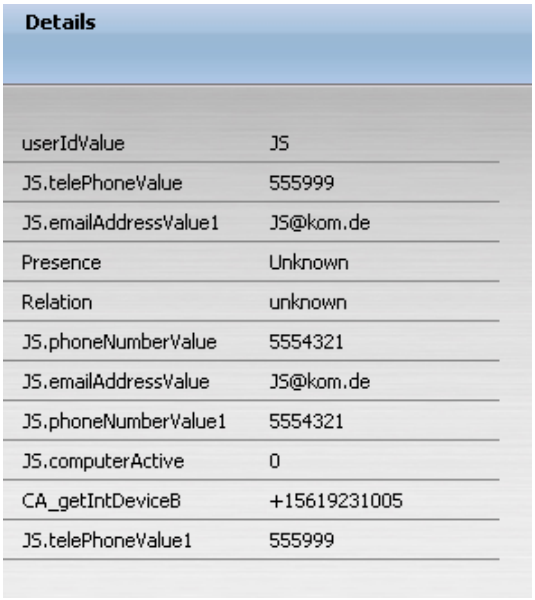
Abbildung 6.19: Anzeige von Verfügbarkeit und Beziehung im Portal der Siemens HiPath Anlage

Der Dienst wurde als Element in dem Web-Portal der Telefonanlage integriert, um Einblick auf das Verhalten des Rahmenwerkes und das Geben von Feedback zu ermöglichen. Wie in Abbildung 6.19 dargestellt wird, ist eine Anzeige des eigenen Kontextes und der Kontexte anderer Nutzer möglich (in diesem Falle die Beziehungs-Kontext und die Verfügbarkeit).

Die Abbildung 6.20 a) zeigt eine Auswahl des aktuellen Profils innerhalb des Portals.



(a) Feedback: Auswahl des Profils



(b) Explizites Feedback: Sensor Details

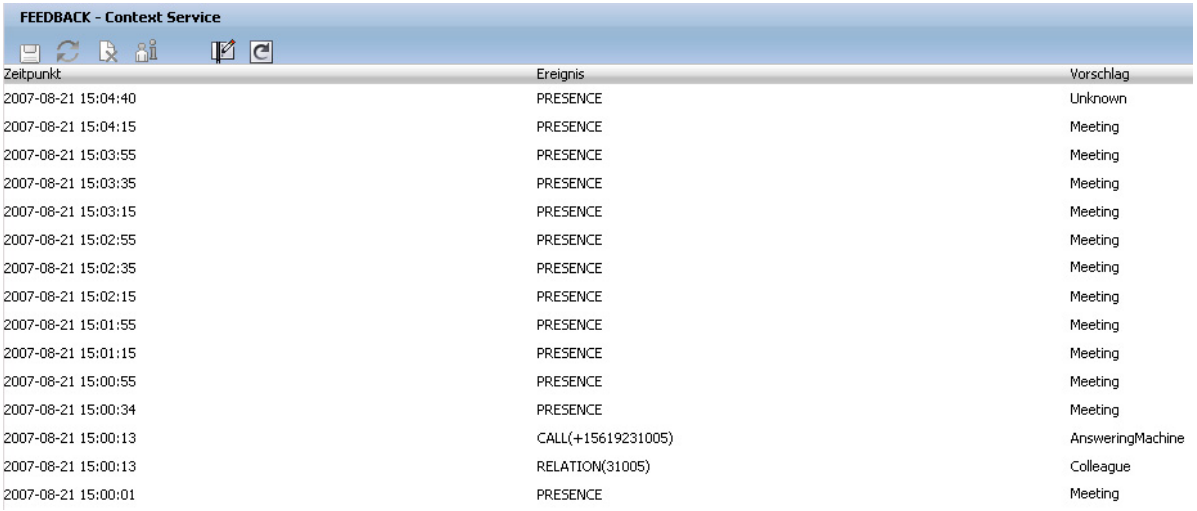
Abbildung 6.20: Zusätzliche Elemente im Portal der Siemens HiPath Anlage

Das dabei ausgewählte Profil wird als implizites Feedback (siehe Abschnitt 5.2.2) herangezogen. Eine weitere Herausforderung ist die Bestimmung von Feedback, sobald sich das Feedback nicht mehr auf die aktuelle Situation bezieht. Dieses explizite Feedback kann vom Nutzer nur gegeben werden, wenn er ausreichend Informationen erhält, um die Situation zu erfassen und selbst zu klassifizieren. Hierzu erhält der Nutzer eine Historie über die getroffenen Entscheidungen, wie sie in Abbildung 6.21 dargestellt wird.

Der Benutzer kann dann bei Bedarf für die Vorschläge aus den Einträgen entsprechendes Feedback geben, wie es in Abbildung 6.20 a) dargestellt wurde. Benötigt der Nutzer weitere Informationen über die Situation, so kann er sich auch die Details über die zur Klassifikation herangezogenen Sensoren anzeigen lassen, wie sie in Abbildung 6.20 b) gezeigt werden.

6.6 Fazit

Das Gesamtsystem wurde als Proof-of-Concept implementiert und findet als Rahmenwerk *ContextFramwork.KOM* Einsatz in verschiedenen Arbeiten am Fachgebiet KOM. Dieses Kapitel zeigte ausgewählte Teile der Realisierung des Gesamtsystems in Form von Komponenten von ContextFramework.KOM. Bei der Umsetzung wurde weiterhin auf eine Reduzierung des Aufwands und der Kosten geachtet, welche durch eine vereinfachte Integration und Nutzung von Komponenten erzielt wird. Durch die Nutzung von OSGI können komplexe Dienste in einzelne,



Zeitpunkt	Ereignis	Vorschlag
2007-08-21 15:04:40	PRESENCE	Unknown
2007-08-21 15:04:15	PRESENCE	Meeting
2007-08-21 15:03:55	PRESENCE	Meeting
2007-08-21 15:03:35	PRESENCE	Meeting
2007-08-21 15:03:15	PRESENCE	Meeting
2007-08-21 15:02:55	PRESENCE	Meeting
2007-08-21 15:02:35	PRESENCE	Meeting
2007-08-21 15:02:15	PRESENCE	Meeting
2007-08-21 15:01:55	PRESENCE	Meeting
2007-08-21 15:01:15	PRESENCE	Meeting
2007-08-21 15:00:55	PRESENCE	Meeting
2007-08-21 15:00:34	PRESENCE	Meeting
2007-08-21 15:00:13	CALL(+15619231005)	AnsweringMachine
2007-08-21 15:00:13	RELATION(31005)	Colleague
2007-08-21 15:00:01	PRESENCE	Meeting

Abbildung 6.21: Explizites Feedback über das Portal der Siemens HiPath Anlage: Historie

wieder verwendbare Komponenten aufgeteilt und bei Bedarf zur Laufzeit dynamisch nachgeladen werden. Die Komponenten lassen sich zudem über ein leichtgewichtiges Web-Portal einsehen und konfigurieren.

Bei der semantischen Beschreibung von Relationen und Sensoren wurde auf die Standards OWL und OWL-S zurückgegriffen. Zur vereinfachten Integration von Sensoren wurde von der Basisfunktionalität eines Sensors abstrahiert und Mechanismen zur automatischen Generierung standardkonformer OWL-S Beschreibungen der Sensoren zur Laufzeit wurden geschaffen. Zur Anbindung weiterer Sensoren wurde der Ansatz eines Gateways entwickelt. Dieses Gateway ermöglicht auch die Anbindung ressourcenbeschränkter Sensoren. Diese Sensoren können dabei durch Proxies angebunden werden, welche auf der Seite des Rahmenwerks die vollständige Funktionalität der Sensor-Schnittstelle bereitstellen. Zur Kommunikation zwischen den Diensten wurde eine Reihe von Konnektoren umgesetzt, welche den Einsatz des Systems auf verschiedenen Plattformen ermöglichen.

Des Weiteren wurde das Sensorverzeichnis vorgestellt, welches die Funktionen der Informationsgewinnung umsetzt. Hierzu verwaltet das Sensorverzeichnis die Liste der aktuell verfügbaren Sensoren und deren Beschreibungen. Die Beschreibungen der Sensoren dienen ebenfalls dazu, die auf OWL basierende Ontologie, welche ebenfalls durch das Sensorverzeichnis verwaltet wird, auf dem aktuellen Stand zu halten. Ein Sensor kann seine Beschreibung mit zusätzlichen Referenzen versehen, welche gegebenenfalls dazu genutzt werden können, unbekannte Begriffe in die Ontologie zu integrieren.

Der Ansatz der ontologiebasierten iterativen Suche wurde ebenfalls im Rahmen des Sensorverzeichnisses umgesetzt. Eine zusätzliche Erweiterung des Ansatzes ermöglicht die Vorbereitung einer Suchanfrage. Eine Anwendung kann im Vorfeld einer Anfrage ihr Interesse an bestimmten Informationen formulieren, sowie Qualitätsanforderungen festlegen. Dieses Vorwissen ermöglicht die Vorbereitung möglicher Suchbäume und das aktive Anfragen von kritischen Sensoren.

Im Rahmen der Auswertung der Informationen wurde der Ansatz der semantischen Adressierung von Informationen entwickelt und umgesetzt. Dieser Ansatz ermöglicht die Anwendung von Aussagen des Modells auf andere Sensoren, welche semantisch vergleichbare Zusammenhangen aufweisen, sowie die Übertragbarkeit von Modellen auf andere Teilnehmer. Im Rahmen der Umsetzung vom ContextFramework.KOM wurde eine Integration des Systems in verschiedene Anwendungen zur Verarbeitung von Kommunikationsanfragen durchgeführt. Hierzu wurde das System in das Kommunikationssystem Asterisk und die Telefonanlage HiPath von Siemens integriert und dort zur kontextabhängigen Verarbeitung von Anfragen anwendbar gemacht.

7 Zusammenfassung und Ausblick

Diese Arbeit hatte zum Ziel, eine adaptive Kontextbestimmung zur Steuerung von Kommunikationsdiensten zu erarbeiten. Um dieses Ziel genauer zu erfassen wurden eine Reihe von Szenarien vorgestellt, welche durch das vorgestellte System adressiert werden. Eine Problemanalyse über diese Szenarien ermöglichte die Ableitung von Rahmenbedingungen, unter welchen das Gesamtkonzept für das System entworfen wurde. Es wurde eine Qualitätsbetrachtung durchgeführt, um die Abhängigkeiten der Leistung des Gesamtsytems von einzelnen Effekten und möglichen Fehlerquellen zu verdeutlichen. Für den Entwurf des Systems wurden verwandte Arbeiten herangezogen, welche ähnliche Themengebiete behandeln. Diese Systeme entsprachen jedoch nicht den geforderten Rahmenbedingungen oder verfügten nicht über die erforderlichen Funktionalitäten.

Das in dieser Arbeit systematisch entwickelte System ermöglicht die Anbindung externer Informationsquellen in den Entscheidungsprozess unter Berücksichtigung dynamischer und heterogener Mengen von Sensoren, wobei besonders auf die nachträgliche Erweiterbarkeit des Systems um neue Sensoren geachtet wurde. Es wurden Technologien aus dem Bereich der semantischen Beschreibung von Netzdiensten und Informationen herangezogen, um die Relevanz von Informationsquellen zu bestimmen. Durch die offene Architektur ist es möglich, mit geringem Aufwand neue Informationsquellen in das System zu integrieren. Darauf aufbauend wurden verschiedene Strategien zur Suche betrachtet und entwickelt, welche aus einer Vielzahl potentiell verfügbarer Informationsquellen effizient diejenigen ermitteln, welche für die Kontextbestimmung relevant sind. Durch eine optimierte Datenvorhaltung ist es nun möglich die Anforderungen zeitbeschränkter Anwendungen auch in Anbetracht ressourcenbeschränkter Sensoren zu erfüllen.

Es wurde die Problematik und die Komplexität statischer Ansätze zur Auswertung großer und dynamischer Mengen von Informationsquellen erläutert. Um Benutzern die Nutzung dieser Informationen zu ermöglichen, muss aus der Sicht des Nutzers von dieser Komplexität abstrahiert werden. Mit dem entwickelten Verfahren ist es möglich, das Verhalten des Systems, an die Vorstellungen und Wünsche der Teilnehmer und den Veränderungen der Informationsgrundlage, anzupassen. Es wurde hierzu eine Reihe von Ansätzen für eine Anpassung der Kontextbestimmung an das Konzept des Nutzers vorgestellt, kategorisiert und analysiert. Im Rahmen dieser Arbeit wurde eine Reihe von unterschiedlichen Lernverfahren durch Simulationen von verschiedenen Bedingungen (bezogen auf die aufgezeigten Rahmenbedingungen) analysiert. Um eine möglichst hohe Akzeptanz des Nutzers im Bezug auf die angestrebten, automatisierten Entscheidungsprozesse zu erreichen, wurde auf eine möglichst schnelle und umfassende Adaption des Nutzerkonzeptes durch das Modell geachtet. Um ebenfalls geeignet auf Änderungen der Nutzerkonzepte reagieren zu können, wurden hierbei Ansätze aus den Bereichen des maschinellen Lernens betrachtet, erweitert und umgesetzt, so dass dies nun möglich ist.

Um das Ziel der Arbeit zur adaptiven Verarbeitung von Kommunikationsanfragen zu erreichen, wurde das Gesamtssystem in einem Kommunikationssystem integriert. Das System wurde mit den in dieser Arbeit beschriebenen Funktionalitäten umgesetzt und erfüllt die geforderten Rahmenbedingungen und Anforderungen. Es bildet ein Rahmenwerk, mit einer generischen Middleware als zentrale Komponente, welche stetig um weitere Funktionen erweitert wird. Durch die Integration vorhandener Informationsquellen und die Anpassung an das Verhalten und die Vorstellungen der Teilnehmer konnten Kommunikationsdienste entwickelt werden, welche die Zielführung von Kommunikationsanfragen optimieren. Als Proof-of-Concept sind zudem eine Reihe von Sensoren und Diensten implementiert worden und stehen ebenfalls für weitere Arbeiten zur Verfügung.

7.1 Sicherheit und Datenschutz

In der personenbezogenen Kontextverarbeitung geht es darum, persönliche Daten zu verarbeiten. Das Rahmenwerk sammelt Informationen über Nutzer und nutzt diese zur Erstellung von (Verhaltens-)Modellen. Dieser Aspekt führt häufig zu der Frage, wie sich dies mit dem Thema Datensicherheit und Datenschutz vereinbaren lässt. Eine andere Arbeit aus dem Bereich Kontextverarbeitung liefert dazu eine Behandlung dieser Thematik, wie sie auch für diese Arbeit herangezogen werden kann [ZHF+08].

Schutz von Sensordaten und Modellen

Die Sensoren, welche sich auf den Nutzer beziehen, der Datensatz mit den gesammelten Trainingsinstanzen und das Modell sind prinzipiell schützenswerte Informationen. Geht man von einem zentralisierten Ansatz aus, so befinden sich die gesammelten Informationen und das Modell jedoch auf einem Server, von dem aus sie für andere Mitarbeiter (beispielsweise aus der Administration) einsehbar wären. Dieser Aspekt von schützenswerten Informationen auf einem Server ist nicht neu. Auf vergleichbare Art und Weise kann etwa auf die Inhalte und Kopfdaten von E-Mails oder Protokollen aus der Telefonanlage zurückgegriffen werden. Eine Einsicht oder eine personenbezogene Auswertung dieser Daten ist jedoch nicht zulässig. Eine analoge Regelung sollte daher bei Modellen, Sensordaten und Trainingsinstanzen Anwendung finden.

Abwägung von Datenschutz und Nutzen

Der zunehmende Anteil von Personen, die persönliche Informationen frei verfügbar im Internet preisgeben (etwa in Systemen zum Aufbau von sozialen Netzwerken oder in Form ihres Verfügbarkeitsstatus in IMS Programmen), deutet darauf hin, dass Teilnehmer bereit sind, Einschränkungen in ihrer Privatsphäre in Kauf zu nehmen, sofern sie ein Nutzen für sich erkennen. Die Daten, die mit den Sensoren im Rahmen dieses Projekts erfasst werden können, können jedoch einen deutlich höheren Grad an Details und privaten Informationen aufweisen, weswegen besondere Mechanismen zur Sicherstellung des Datenschutzes ergriffen werden müssen.

Abgeschlossenes System

Eine Möglichkeit, mit der Problematik umzugehen, besteht darin, ein geschlossenes System aufzubauen. Hierzu können autonom agierende, zentrale Server in einzelnen Unternehmen zum Einsatz kommen, die vom Internet entkoppelt oder durch ausreichende Sicherheitsmaßen geschützt sind und somit nur von einem geschlossenen internen Benutzerkreis verwendet werden können. Alternativ kann das System aber auch für die Ausführung auf einzelnen Endgeräten angepasst werden. So wären die gesammelten Informationen und das Modell des Nutzers prinzipiell nur für den Nutzer selbst zugänglich, und er selbst kann bestimmen, in wie weit Anfragen beantwortet oder Informationen nach außen weitergegeben werden. Im Rahmen der exklusiven Ausführung auf einem Endgerät besteht allerdings der Nachteil, dass Sensorinformationen von anderen Teilnehmern nicht oder nur eingeschränkt zur Kontextbestimmung verwendet werden können. Bei einer geringen Anzahl von Sensoren auf dem Endgerät kann eine gute Approximation an das Konzept des Benutzers möglicherweise nicht sicher gestellt werden. Zur Ausführung auf einem mobilen Endgerät müsste zudem auf eine ressourcensparende Variante der Suche und des Lernverfahrens geachtet werden.

Weitere Sensoren des Nutzers, welche sich auf Geräten in der Umgebung befinden, könnten durch Sicherheitsmaßnahmen wie Authentifikation und Verschlüsselung gesichert werden, so dass nur das Systems des Nutzers an diese Informationen gelangen kann. Falls anderen Teilnehmern ein Zugriff erlaubt werden soll, können die Sensoren explizit freigegeben werden, so wie dies beispielsweise in Skype gehandhabt wird.

Nutzung von Informationen über andere Nutzer

Ein abgeschlossenes System hat den Nachteil, dass mancher Nutzen, der aus den Informationsquellen gezogen werden kann, verloren geht. Beispielsweise kann es von Nutzen sein zu erfahren, welche anderen Teilnehmer sich gerade im selben Raum aufhalten, um Situationen wie Besprechungen zu detektieren oder den Kontext der anderen Teilnehmer bei der Bestimmung des eigenen Kontextes zu berücksichtigen.

Detailgrad der Informationen

Geht man von einem abgeschlossenen System aus, stellt sich die Frage, bis zu welchem Detailgrad ein Benutzer bereit ist, Informationen über sich zu veröffentlichen. Der Nutzer kann prinzipiell selbst entscheiden, welche Informationen veröffentlicht werden.

Das Ergebnis der Auswertung stellt im Allgemeinen jedoch nur eine stark abstrahierte Form der gesammelten Informationen dar und sollte daher in der Regel weniger Bedenken im Rahmen des Datenschutzes aufweisen. Der Detailgrad konkreter Sensordaten kann bedarfsweise herabgesetzt werden, um beispielsweise den genauen Aufenthaltsort zu verschleiern. Die Bestimmung, ob und zu welchem Detailgrad Informationen bereitgestellt werden, kann auch von der Beziehung zu dem anfragenden Teilnehmer abhängig gemacht werden.

Fokus dieser Arbeit

Über die zuvor genannten Ansätze ist eine Berücksichtigung des Datenschutzes prinzipiell möglich. In dieser Arbeit liegt der Fokus auf der Betrachtung der generellen Machbarkeit des Ansatzes, um den prinzipiellen Nutzen

eines solchen Systems zu analysieren. Erst wenn gezeigt werden kann, dass ein solches System realisierbar ist und ein Mehrwert aus der Nutzung eines solchen Systems gezogen werden kann, ist der Ansatz in einem Stadium, in dem weitere Schritte wie Datenschutz relevant werden. Daher ist das Thema Datenschutz ein wichtiger Aspekt, welcher in den weiterführenden Arbeiten an dem Rahmenwerk berücksichtigt werden muss.

7.2 Ausblick

Es besteht eine Reihe von Anknüpfungspunkten, um diese Arbeit fortzusetzen. Diese Anknüpfungspunkte betreffen verschiedene Aspekte und wurden zum Teil in den betreffenden Abschnitten bereits erwähnt. Sie umfassen:

- **Dezentrale Architektur/Suche:** Die Gesamtarchitektur ist bisher auf zentrale Elemente, wie das Sensorverzeichnis oder den Kontextdienst ausgelegt. Es wurde bisher auch Wert auf eine Reduzierung des Aufwandes gelegt. Ab einer großen Anzahl von Sensoren ist jedoch eine Verteilung der Last erforderlich. Das System könnte dahingehend erweitert werden, dass für jeden (semantisch abgrenzbaren) Bereich (z.B. ein Gebäude) ein eigenes Sensorverzeichnis genutzt wird. Diese Sensorverzeichnisse lassen sich selbst mittels semantischer Beschreibungen auszeichnen und verknüpfen, um eine Suche über mehrere Sensorverzeichnisse hinweg durchzuführen. Die Suche kann dann auf diejenigen Sensorverzeichnisse begrenzt werden, die für die Suche relevant sind.
- **Adaptive Suche:** Die Suche ist aktuell unabhängig von dem Nutzerkonzept oder dem Modell, welches zur Auswertung herangezogen wird. Der nächste Schritt in diesem Bereich besteht in dem Heranziehen des Modells zur Bestimmung relevanter Sensoren. Zur Auswertung des Modells kann dann die Suche auf diese relevanten Sensoren beschränkt werden. Dadurch kann die Suche weiter eingegrenzt und effizienter gestaltet werden.
- **Qualitätsmetriken:** Die Qualität des Gesamtsystems orientiert sich am Empfinden des Nutzers. Ziel ist es, eine möglichst hohe Übereinstimmung mit dem Konzept des Nutzers, bei der Wahl einer geeigneten Aktion, zu erzielen. Andererseits muss der Aufwand bei der Suche minimal gehalten werden, um Quality of Service (QoS) Anforderungen der Laufzeit und der Skalierbarkeit einzuhalten. Bei einer *adaptiven Suche* kannte eine Optimierung hinsichtlich der Abwägung zwischen Quality of Decision (QoD) und QoS durchgeführt werden. Weitere Qualitätsmerkmale, wie beispielsweise die Kosten einer Anfrage in Form von Batterieverbrauch oder benötigter Bandbreite, können innerhalb einer Metrik genutzt werden, um die Anzahl der benötigten Suchanfragen weiter zu reduzieren. Hierdurch kann eine effiziente Suche unter Einhaltung von QoD-Anforderungen erzielt werden.
- **Suche nach versteckten Attributen:** Wie in Abbildung 4.6.3 dargestellt wurde, ist in der Suche das Auffinden von versteckten Attributen eine komplexe Aufgabe. Diese Aufgabe kann gegebenenfalls durch eine Vorverarbeitung aller möglichen Suchpfade angegangen werden.
- **Vorverarbeitung:** Aktuell können die gesammelten Informationen innerhalb eines Suchergebnisses bereits über statische Methoden vorverarbeitet und miteinander kombiniert werden. Bei einer steigenden Menge von semantischen Datentypen, ist auch hier eine statische Definition aller möglichen Kombinationen der Vorverarbeitung zu komplex und aufwändig. In diesem Bereich kann ebenfalls die semantische Beschreibung der Datentypen und die Ontologie genutzt werden. So lassen sich Vorverarbeitungsregeln möglicherweise effizienter mittels semantischer Ausdrücke beschreiben, wie sie beispielsweise durch SWRL bereitgestellt werden. Hierzu ist im aktuellen System bereits die Darstellung der Suchergebnisse innerhalb der Ontologie möglich.
- **Lernverfahren:** Es wurde gezeigt, dass die Informationsgrundlage innerhalb des angestrebten Anwendungsszenarios die Lernverfahren vor eine Reihe von Herausforderungen stellt. Gerade im Bereich der Online-Lernverfahren können hierzu noch Optimierungen durchgeführt werden, um auch auf stark fehlerbehafteten Trainingsdaten und unter Konzeptänderungen gute Ergebnisse mit geringem Rechenaufwand zu erhalten.
- **Langzeittests:** Diese Arbeit nutzt eine Reihe von Annahmen und künstlich erzeugten Eigenschaften bei der Verarbeitung von Sensordaten und der Auswertung von Modellen. Das System wurde jedoch für den Einsatz als reales Testbed konzipiert und ist bereit für die Erfassung und Auswertung von Informationen aus realen Szenarien. Langzeittests können dazu genutzt werden, um die realen Eigenschaften der Informationsquellen zu bestimmen und das Zusammenspiel mit Konzepten realer Nutzer zu analysieren.
- **Kommunikation zwischen virtuellen Assistenten:** Diese Arbeit nutzt das Rahmenwerk, um den angerufenen Teilnehmer bei der Verarbeitung von Kommunikationsanfragen zu unterstützen. Nutzt der Anrufer ebenfalls ein ähnliches System, welches den Kontext des Anrufers erfasst, so kann dieser ebenfalls zur Ent-

scheidungsfindung herangezogen werden. Hierzu ist es denkbar, dass die Systeme auf beiden Seiten bereits im Vorfeld einer Kommunikation den geeigneten Zeitpunkt und die geeignete Form für die Durchführung bestimmen, indem sie den Kontext austauschen und eine gemeinsame Entscheidungsfindung durchführen.

7.3 Fazit

Eine lernende Inferenzlogik schließt aus den verfügbaren Informationen auf den aktuellen Kontext, der im System registrierten Benutzer und kann dieses Wissen gezielt zur Unterstützung von Kommunikationsdiensten bereitstellen.

Konkret wird dies realisiert, indem aus einem autonom lernenden Modell, das sowohl Sensorwerte als auch persönliche Präferenzen der Nutzer berücksichtigt, verschiedene höherwertige Informationen generiert werden, die sowohl zur Verbesserungen in der Erreichbarkeit innerhalb von Kommunikationssystemen, als auch in der Kommunikation allgemein herangezogen werden können.

Im exemplarisch betrachteten Fall der kontextberücksichtigenden Telefonie können diese Informationen etwa dahingehend eingesetzt werden, dass etwa bei längeren Arbeitssituationen des Nutzers, die keine Unterbrechung erlauben, Telefone automatisch stumm geschaltet werden und Anrufenden eine entsprechende Mitteilung ausgegeben wird.

Durch die Erfassung aller Kommunikationsmöglichkeiten eines Teilnehmers und die Berücksichtigung der im jeweiligen Moment verfügbaren Endgeräte, wird zudem eine erhöhte Zielführung der Kommunikation durch die Auswahl des am besten geeigneten Kommunikationsmediums gewährleistet.

Im Gegensatz zu existierenden Lösungen, bei denen die Konfiguration von Einstellungen am Telefon, wie z.B. Weiterleitung oder Einspielung einer entsprechenden Ansage, manuell oder durch komplexe Regelwerke vorgenommen werden muss, handelt ContextFramework.KOM autonom und adaptiv, und berücksichtigt hierbei verfügbare Sensorwerte zur Erstellung eines Benutzerprofils und der deduktiven Ableitung von Regeln. Insgesamt zeichnet sich das System für jeden Benutzer dadurch aus, dass es das Potential besitzt, eine Vielzahl von Tätigkeiten zu automatisieren, die heute nur ein persönlicher Assistent erfüllen kann.

Literaturverzeichnis

[AAK91] Aha, D. ; Albert, M. ; Kibler, D.: Instance-Based Learning Algorithms. In: *Machine Learning* 6 (1991), Nr. 1

[Ack03] Ackermann, R.: *Gateways and Components for Supplemtary IP Telephony Services in Heterogeneous Environments*, Technische Universität Darmstadt, Dissertation, 2003

[Ait06] Aitenbichler, E.: *System Support for Ubiquitous Computing*, Technische Universität Darmstadt, Dissertation, 2006

[Ait07] Aitenbichler, E.: *MundoCore*. 2007. – `http://www.tk.informatik.tu-darmstadt.de/research/ubicomp/mundocore`

[All96] Allan, J.: Incremental Relevance Feedback for Information Filtering. In: *SIGIR '96: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM Press, 1996

[AMB+94] Althoff, K. ; Manago, M. ; Bergmann, R. ; Maurer, F. ; S.Wess ; Auriol, E. ; Conruyt, N. ; Traphöner, R. ; Dittrich, M. Bräuerand S.: Induction and Case-Based Reasoning for Classification Tasks. In: *Information Systems and Data Analysis, Prospects Foundations -Applications, Proceedings of the 17th Annual Conference of the GfKl*, Springer Verlag, 1994

[AWSBL99] Adjie-Winoto, W. ; Schwartz, E. ; Balakrishnan, H. ; Lilley, J.: The Design and Implementation of an Intentional Naming System. In: *17th ACM Symposium on Operating Systems Principles*, 1999

[BBHS] Bauer, M. ; Becker, C. ; Hähner, J. ; Schiele, G.: *ContextCube - Providing Context Information Ubiquitously*. – `http://www.vs.inf.ethz.ch/publ/se/contextcube-final-final.pdf`

[BBK02a] Balazinska, M. ; Balakrishnan, H. ; Karger, D.: INS/Twine: A Scalable Peer-to-Peer Architecture for Intentional Resource Discovery. In: *International Conference on Pervasive Computing*, 2002

[BBP+01] Becker, O. ; Brose, M. ; Plume, B. ; S. ; Trebs, G. ; Witzel, R.: *Projekt X-ING*. 2001. – `http://www.informatik.hu-berlin.de/~xing/CPLEditor`

[BDPT07] Bisdikian, C. ; Damarla, R. ; Pham, T. ; Thomas, V.: *Quality of Information in Sensor Networks*. 2007. – `http://www.usukita.org/papers/2924/QoI-ITAConf-2007.pdf`

[Ber07] Berbner, R.: *Dienstgüteunterstützung für Service-orientierte Workflows*, Technische Universität Darmstadt, Dissertation, 2007

[BF99] Brodley, C. ; Friedl, M.: Identifying Mislabeled Training Data. In: *Journal of Artificial Intelligence Research* 11 (1999)

[BFK05] Bruijn, J. de ; Fensel, D. ; Kifer, M.: *Relationship of WSMO to other relevant technologies*. 2005. – `http://www.w3.org/Submission/WSMO-related/`

[BGRS99] Beyer, K. ; Goldstein, J. ; Ramakrishnan, R. ; Shaft, U.: When is Nearest Neighbor Meaningful? In: *Lecture Notes in Computer Science* 1540 (1999)

[BGS99] Beigl, M. ; Gellersen, H. ; Schmidt, A.: There is more to Context than Location: Environment-Sensing Technologies for Adaptive Mobile User Interfaces. In: *Computers and Graphics* 23 (1999), Nr. 6

[BHRS09] Bergsträsser, S. ; Hildebrandt, T. ; Rensing, C. ; Steinmetz, R.: Virtual context based services for multiplayer online games to facilitate community participation. In: *Multimedia Tools and Applications* (2009). – ISSN 1380–7501 (Print) 1573–7721 (Online)

[BI98] Brooks, R. ; Iyengar, S.: *Multi-Sensor Fusion: Fundamentals and Applications*. Prentice Hall, 1998

[BJ02] Bakker, J. ; Jain, R.: Next Generation Service Creation Using XML Scripting Languages. In: *Proceedings of ICC, New York(USA)*, 2002

[BLHL] Berners-Lee, T. ; Hendler, J. ; Lassila, O.: The Semantic Web. In: *Scientific American, Mai 2001*

[BLMM04] Becker, R. ; Liebsch, F. ; Michel, Y. ; Müller, D.: *JXTA: Einführung und Überblick*. 2004. – http://www.ag-nbi.de/lehre/04/S_P2PNET/Ausarbeitungen/JXTA.pdf

[Blu04] Bluetooth Consortium: *Bluetooth Core Specification v2.0 - Volume 3 Part B - Service Discovery Protocol (SDP)*. 2004

[BM08] Bach, S. ; Maloof, M.: Paired Learners for Concept Drift. In: *Proceedings of the Eighth IEEE ICDM International Conference on Data Mining* (2008)

[BR00a] Bettstetter, C. ; Renner, C.: *A Comparison of Service Discovery Protocols and Implementation of the Service Location Protocol*. 2000. – http://www.bettstetter.com/publications/bettstetter-2000-eunice-slp.pdf

[CGM03] Castillo, G. ; Gama, J. ; Medas, P.: Adaptation to Drifting Concepts. In: *EPIA* Bd. 2902, Springer, 2003

[CJK+01] Case, J. ; Jain, S. ; Kaufmann, S. ; Sharma, A. ; Stephan, F.: Predictive Learning Models for Concept Drift. In: *Theoretical Computer Science* 268 (2001), Nr. 2

[CM00] Castro, P. ; Muntz, R.: *Managing Context Data for Smart Spaces*. 2000. – http://www.comsoc.org/pci/private/2000/oct/castro.html

[CN87] Clark, P. ; Niblett, T.: Induction in Noisy Domains. In: *Proceedings of the 2nd European Working Session on Learning*, 1987

[CQV01] Cortese, E. ; Quarta, F. ; Vitaglione, G.: *Scalability and Performance of JADE Message Transport System*. 2001. – http://jade.tilab.com/papers/Final-ScalPerfMessJADE.pdf

[DAS99] Dey, A. ; Abowd, G. ; Salber, D.: A Context-based Infrastructure for Smart Environments. In: *Proceedings of the 1st International Workshop on Managing Interactions in Smart Environments (MANSE)*, 1999

[Das02b] Dasarathy, B.: Nearest-Neighbour Approaches. In: Klösgen, Willi (Hrsg.) ; Zytkow, Jan M. (Hrsg.): *Handbook of Data Mining and Knowledge Discovery*. Oxford University Press, 2002

[Dea00] Deason, N.: *SIP and SOAP*. 2000. – http://www.softarmor.com/wgdb/docs/draft-deason-sip-soap-00.txt

[Dey00] Dey, A.: *Providing Architectural Support for Building Context-Aware Applications*, Georgia Institute of Technology, Dissertation, 2000. http://www-static.cc.gatech.edu/fce/ctk/pubs/dey-thesis.pdf

[Die03] Dieckmann, J.: *DAML+OIL und OWL: XML-Sprachen für Ontologien*. 2003

[DKS06a] Danninger, M. ; Kluge, T. ; Stiefelhagen, R.: *MyConnector - Analysis of Context Cues to Predict Human Availability for Communication*. 2006

[DMA+02] Dey, A. ; Mankoff, J. ; Abowd, G. ; ; Carter, S.: *Distributed Mediation of Ambiguous Context in Aware Environments*. 2002. – http://www.intel-research.net/Publications/Berkeley/120520021019_6.pdf

[DRT04] Dague, S. ; Rzeszutek, K. ; Taylor, K.: *Patent: Method for Using SNMP as an RPC Mechanism for Exporting the Data Structures of a Remote Library*. 2004

[EG01] Edwards, K. ; Grinter, R.: *At Home with Ubiquitous Computing: Seven Challenges*. 2001. – http://www.cc.gatech.edu/~keith/pubs/ubicomp2001-challenges.pdf

[EHN94] Erol, K. ; Hendler, J. ; Nau, D.: Semantics for Hierarchical Task-Network Planning. 1994. – Forschungsbericht

[Eis08] Eisenman, S.: *People-Centric Mobile Sensing Networks*, University of Kent at Canterbury, Dissertation, 2008

[EW91] Elio, R. ; Watanabe, L.: An Incremental Deductive Strategy for Controlling Constructive Induction in Learning from Examples. In: *Machine Learning* 7 (1991), Nr. 1

[Faa04] Faatz, A.: *Ein Verfahren zur Anreicherung fachgebietsspezifischer Ontologien durch Begriffsvorschläge*, Technische Universität Darmstadt, Dissertation, 2004

[FMR03] Ferscha, A. ; Mayrhofer, R. ; Radi, H.: Recognizing and predicting context by learning from user behavior. In: *Proceedings of the 1st International Conference on Advances in Mobile Multimedia* Bd. 171, 2003

[Fro97] Froitzheim, K.: *Multimedia-Kommunikation: Dienste, Protokolle und Technik für Telekommunikation und Computernetze*. Dpunkt Verlag, 1997

[FW00] Frank, E. ; Witten, I.: *Data Mining*. Morgan Kaufmann, 2000

[GAS04] Goertz, M. ; Ackermann, R. ; Steinmetz, R.: *The Digital Call Assistant: Determine Optimal Time Slots for Calls*. 2004

[GKK00] Guedhami, B. ; Klein, C. ; Kellerer, W.: Web Enabled Telecommunication Service Control Using VoxML. In: *SmartNet2000, Sixth IFIP International Conference on Intelligence in Networks*, 2000

[Gör05] Görtz, M.: *Effiziente Echtzeit-Kommunikationsdienste durch Einbeziehung von Kontexten*, Technische Universität Darmstadt, Dissertation, 2005. – `http://elib.tu-darmstadt.de/diss/000592/`

[GW97] Ganter, B. ; Wille, R.: *Applied Lattice Theory: Formal Concept Analysis*. `http://www.bib.mathematik.tu-darmstadt.de/Math-Net/Preprints/Listen/pp97.html`. Version: 1997

[HA03] Horvitz, E. ; Apacible, J.: *Learning and Reasoning about Interruption*. 2003. – `http://www.cse.unr.edu/~sushil/class/ps/papers/LearnInterruptiw.pdf`

[Hec04] Heckmann, O.: *A System-oriented Approach to Efficiency and Quality of Service for Internet Service Providers*, Technische Universität Darmstadt, Dissertation, 2004

[HH98] Harries, M. ; Horn, K.: Learning Stable Concepts in a Changing World. In: *Lecture Notes in Computer Science* 1359 (1998)

[Hil09] Hilbrich, R.: *Das Leistungspotential von DPWS für Service-orientiertes Ubiquitäres Computing*, Humboldt-Universität zu Berlin, Diplomarbeit, 2009

[HK01] Han, J. ; Kamber, M.: *Data Mining. Concepts and Techniques.* Morgan Kaufmann, 2001

[HKKJ02] Horvitz, E. ; Koch, P. ; Kadie, C. ; Jacobs, A.: *Coordinate: Probabilistic Forecasting of Presence and Availability*. 2002. – `http://research.microsoft.com/en-us/um/people/horvitz/coordinate.htm`

[HKPH03] Horvitz, E. ; Kadie, C ; Paek, T. ; Hovel, D.: *Models of Attention in Computing and Communication: From Principles to Applications*. 2003. – `http://research.microsoft.com/en-us/um/people/horvitz/cacm-attention.htm`

[Hol04] Hollick, M.: *Dependable Routing for Cellular and Ad hoc Networks*, Technische Universität Darmstadt, Dissertation, 2004

[HRSV00] Hammer, D. ; Romashchenko, A. ; Shen, A. ; Vereshchagin, N.: Inequalities for Shannon Entropy and Kolmogorov Complexity. In: *Journal of Computer and System Sciences* 60 (2000), Nr. 2

[Ian07] Iancu, Bogdan-Andrei: *CPLEd - A CPL Editor*. 2007. – `http://old.iptel.org/products/cpled/`

[IWL88] Iba, W. ; Woogulis, J. ; Langley, P.: Trading Off Simplicity and Coverage in Incremental Concept Learning. In: *Proceedings of the Fifth International Conference on Machine Learning*, Morgan Kaufmann, 1988

[JH01] Jerzy, W. ; Hu, M.: A Comparison of Several Approaches to Missing Values in Data Mining. In: *Lecture Notes in Computer Science* 2005 (2001)

[Kam95] Kambhampati, S.: A Comparative Analysis of Partial Order Planning and Task Reduction Planning. In: *SIGART Bull.* 6 (1995), Nr. 1

[Kar00] Karsten, M.: *QoS Signalling and Charging in a Multi-service Internet using RSVP*, Technische Universität Darmstadt, Dissertation, 2000

[KBKA05] Karlsson, B. ; Bäckström, O. ; Kulesza, W. ; Axelsson, L.: *Intelligent Sensor Networks - an Agent-Oriented Approach*. 2005. – http://www.sics.se/realwsn05/papers/karlsson05intelligent.pdf

[KBM+02] Kindberg, T. ; Barton, J. ; Morgan, J. ; Becker, G. ; Bedner, I. ; Caswell, D. ; Debaty, P. ; Gopal, G. ; Frid, M. ; Krishnan, V. ; Morris, H. ; Pering, C. ; Schettino, J. ; Serr, B. ; Spasojevic, M.: People, Places, Things: Web Presence for the Real World. In: *WMCSA2000, MONET Vol. 7*, 2002. – http://champignon.net/cooltown.php

[Ker09] Kern, N.: *Multi-Sensor Context-Awareness for Wearable Computing*, Technische Universität Darmstadt, Dissertation, 2009

[Kja09] Kjaer, K.: *A Survey of Context-aware Middleware*. 2009. – http://www.hydramiddleware.eu/hydra_papers/A_Survey_of_Context-aware_Middleware.pdf

[KK92] Krizakova, I. ; Kubat, M.: FAVORIT: Concept Formation with Ageing of Knowledge. In: *Pattern Recognition Letters* 13 (1992)

[KK08] Klusch, M. ; Kapahnke, P.: *Semantic Web Service Selection with SAWSDL-MX*. 2008. – http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-416/paper1.pdf

[Kli98] Klingenberg, R.: *Maschinelle Lernverfahren zum adaptiven Informationsfiltern bei sich verändernden Konzepten*, Universität Dortmund, Diplomarbeit, 1998

[KM03] Kolter, J. ; Maloof, M.: Dynamic Weighted Majority: A new Ensemble Method for Tracking Concept Drift. In: *Proceedings of Third IEEE International Conference on Data Mining (ICDM 2003)*, 2003

[KMP07] Karnick, M. ; Muhlbaier, M. ; Polikar, R.: *Incremental Learning in Non-stationary Environments with Concept Drift using a Multiple Classifier Based Approach*. 2007

[Koh96] Kohavi, R.: Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid. In: *Proceedings of the Second International Conference on Knowledge Discoveryand Data Mining*, 1996

[Koy00a] Koychev, I.: *Adaptation to Drifting User's Interests*. 2000

[Koy00b] Koychev, I.: Gradual Forgetting for Adaptation to Concept Drift. In: *Proceedings of ECAI Workshop Current Issues in Spatio-Temporal Reasoning*, 2000

[Koy01] Koychev, I.: *Learning about User in the Presence of Hidden Context*. 2001. – http://www.cs.rutgers.edu/ml4um/mirrors/ml4um-2001/papers/IK.pdf

[KR98] Klinkenberg, R. ; Renz, I.: Adaptive information filtering: Learning in the Presence of Conceptdrifts. In: *Learning for Text Categorization*, AAAI Press, 1998

[Kre05] Kresse, F.: Schwerpunkt: Filtermethoden für Spam. In: *Computerwelt* (2005). http://www.computerwelt.at/detailArticle.asp?a=95558&n=3

[KSBM99] Keerthi, S. ; Shevade, S. ; Bhattacharyya, C. ; Murthy, K.: Improvements to Platt's SMO Algorithm for SVM Classifier Design / Dept. of Mechanical and Production Engineering - National University of Singapore. 1999 (CD-99-14). – Forschungsbericht

[KW96] Kubat, M. ; Widmer, G.: Learning in the Presence of Concept Drift and Hidden Contexts. In: *Machine Learning* 23 (1996), Nr. 1

[Lac05] Lacy, L.: *OWL: Representing Information Using the Web Ontology Language*. Trafford Publishing, 2005

[Lae99] Laerhoven, K. V.: *Online Adaptive Context Awareness*, Universität Brüssel, Diplomarbeit, 1999. – http://www.teco.edu/tea/thesis99.ps

[LAV+05] Lima, W. de ; Alves, R. ; Vianna, R. ; Almeida, M. ; Tarouco, L. ; Granville, L.: *Evaluating the Performance of SNMP and Web Services Notifications*. 2005. – http://elib.tu-darmstadt.de/ieee/stamp/stamp.jsp?arnumber=1687583&isnumber=35594

[LF03] Licciardi, C. ; Falcarin, P.: *Technologies and Guidelines for Service Creation in NGN*. 2003

[Lit88] Littlestone, N.: Learning Quickly when Irrelevant Attributes Abound: A new Linear-Threshold Algorithm. In: *Machine Learning* 2 (1988), Nr. 4

[LJ05] Li, G. ; Jacobsen, H.: *Composite Subscriptions in Content-based Publish/Subscribe Systems*. 2005. – http://www.msrg.utoronto.ca/publications/padres-middleware2005.pdf

[LKM06] Liu, S. ; Küngas, P. ; Matskin, M.: *Agent-Based Web Service Composition with JADE and JXTA*. 2006. – http://www.idi.ntnu.no/~peep/papers/SWWS2006_LiKM.pdf

[LPL+05] Lara, R. ; Polleres, A. ; Lausen, H. ; Roman, D. ; Bruijn, J. de ; Fensel, D.: *A Conceptual Comparison between WSMO and OWL-S*. 2005. – http://www.wsmo.org/2004/d4/d4.1/v0.1/20050106/d4.1v0.1_20050106.pdf

[LS04] Lennox, J. ; Schulzrinne, H.: *Call Processing Language (CPL): A Language for User Control of Internet Telephony Services*. 2004. – Request for Comments 3880

[LSR01] Lennox, J. ; Schulzrinne, H. ; Rosenberg, J.: *Common Gateway Interface for SIP*. 2001. – RFC 3050

[LW00] Lehn, J. ; Wegmann, H.: *Einführung in die Statistik*. Bd. 3. B. G. Teubner Verlag, 2000

[Mal04] Maloof, M.: *Incremental Learning with Partial Instance Memory*. 2004. – http://www.cs.georgetown.edu/~maloof/pubs/ismis02.pdf

[MAP00] Metsis, V. ; Androutsopoulos, I. ; Paliouras, G.: *Spam Filtering with Naive Bayes - Which Naive Bayes?* 2000

[Mar03] Markowetz, F.: *Klassifikation mit Support Vector Machines*. 2003. – http://lectures.molgen.mpg.de/statistik/docs/Kapitel_16.pdf

[May04] Mayrhofer, R.: *An Architecture for Context Prediction*, Universität Linz, Dissertation, 2004

[MCE02] Mascolo, C. ; Capra, L. ; Emmerich, W.: Mobile Computing Middleware / Dept. of Computer Science University College London. 2002. – Forschungsbericht

[Meh08] Mehlhase, S.: *Kontextvorhersage in einem adaptiven, kontext-sensitiven System*, Technische Universität Darmstadt, Diplomarbeit, 2008. – http://www.ke.informatik.tu-darmstadt.de/lehre/arbeiten/master/2008/Mehlhase_Stephan.pdf

[Mic83] Michalski, R.: *Machine Learning*. Morgan Kaufmann, 1983

[MLF+08] Miluzzo, E. ; Lane, N. ; Fodor, K. ; Peterson, R. ; Lu, H. ; Musolesi, M. ; Eisenman, S. ; Zheng, X. ; Campbell, A.: *Sensing Meets Mobile Social Networks: The Design, Implementation and Evaluation of the CenceMe Application*. 2008

[MM00b] Maloof, M. ; Michalski, R.: Selecting Examples for Partial Memory Learning. In: *Machine Learning* 41 (2000), Nr. 1

[MPW07] Martin, D. ; Paolucci, M. ; Wagner, M.: *Toward Semantic Annotations of Web Services: OWL-S from the SAWSDL Perspektive*. 2007. – http://www.ai.sri.com/OWL-S-2007/final-versions/OWL-S-2007-Martin-Final.pdf

[MS06] Moodley, D. ; Simonis, I.: A New Architecture for the Sensor Web: The SWAP Framework. In: *ISWC: 5th International semantic web conference*, 2006

[Nel98] Nelson, G.: *System Support for Ubiquitous Computing*, University of Cambridge, Dissertation, 1998. – http://www.sigmobile.org/phd/1998/theses/nelson.pdf

[New05] Newmarch, J.: *UPnP Services and Jini Clients*. 2005. – http://jan.newmarch.name/publications/jini_upnp.isng05.pdf

[New06] Newmarch, J.: *AGENTOWL: Semantic Knowledge Model and Agent Architecture*. 2006. – http://agentowl.sourceforge.net/publications/AgentOWL_cai2006.pdf

[OFG97] Osuna, E. ; Freund, R. ; Girosi, F.: *Improved Training Algorithm for Support Vector Machines*. 1997

[Ome00] Omelayenko, B.: *Machine Learning for Ontology Learning*. 2000

[Pas01] Pascoe, J.: *Context-Aware Software*, University of Kent at Canterbury, Dissertation, 2001. – http://www.cs.kent.ac.uk/pubs/2001/1390/index.html

[PDMQ04] Pras, A. ; Drevers, T. ; Meent, R. van d. ; Quartel, D.: *Comparing the Performance of SNMP and Web Services-Based Management*. 2004. – http://www.simpleweb.org/nm/research/results/publications/pras/2004-eTNSM.pdf

[PF02] Ponnekanti, S. ; Fox, A.: *SWORD: A Developer Toolkit for Web Service Composition*. 2002. – http://www2002.org/CDROM/alternate/786/

[Pla99] *Kapitel* Fast Training of Support Vector Machines using Sequential Minimal Optimization. In: Platt, J.: *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999

[PNL02] Pease, A. ; Niles, I. ; Li, J.: The Suggested Upper Merged Ontology: A Large Ontology for the SemanticWeb and its Applications. In: *Working Notes of the AAAI Workshop on Ontologies and the SemanticWeb*, 2002

[RA07] Rellermeyer, J. ; Alonso, G.: *Concierge: A Service Platform for Resource-Constrained Devices*. 2007. – http://www.iks.inf.ethz.ch/publications/files/rellermeyer_eurosys07.pdf

[RAR07b] Rellermeyer, J. ; Alonso, G. ; Roscoe, T.: *R-OSGi: Distributed Applications through Software Modularization*. 2007. – http://www.iks.inf.ethz.ch/publications/files/rellermeyer-middleware07.pdf

[Rep09] Repp, N.: *Überwachung und Steuerung dienstbasierter Architekturen - Verteilungsstrategien und deren Umsetzung*, Technische Universität Darmstadt, Dissertation, 2009

[RHS09] Reinhardt, A. ; Hollick, M. ; Steinmetz, R.: Stream-oriented Lossless Packet Compression in Wireless Sensor Networks. In: *Proceedings of the Sixth Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2009

[Rie94] Riedmiller, M.: Advanced Supervised Learning in Multi-layer Perceptrons - From Backpropagation to Adaptive Learning Algorithms. In: *Int. Journal of Computer Standards and Interfaces* 16 (1994)

[RKT05] Russomano, D. ; Kothari, C. ; Thomas, O.: Building a Sensor Ontology: A Practical Approach Leveraging ISOand OGC Models. In: *The International Conference on Artificial Intelligence* (2005)

[Roe02] Roedig, U.: *Firewall Architectures for Multimedia Applications*, Technische Universität Darmstadt, Dissertation, 2002

[Sar04] Sarle, W.: *What are Batch, Incremental, On-line, Off-line*. http://www.faqs.org/faqs/ai-faq/neural-nets/part2/section-2.html. Version: 2004

[Sch00b] Schmitt, J.: *Heterogeneous Network QoS Systems*, Technische Universität Darmstadt, Dissertation, 2000

[Sch02c] Schmidt, A.: *Ubiquitous Computing, Computing in Context*, Lancaster University, Dissertation, 2002. – http://www.comp.lancs.ac.uk/~albrecht/phd/Albrecht_Schmidt_PhD-Thesis_Ubiquitous-Computing_ebook1.pdf

[Sch04b] Schöner, H.: *Working with Real-World Datasets*, Technische Universität Berlin, Dissertation, 2004

[Sch05] Schulzrinne, H.: *Internet Telefonie - Mehr als nur ein Telefon mit Paketvermittlung*. Columbia University, New York, 2005

[SDA99] Salber, D. ; Dey, A. ; Abowd, G.: The Context Toolkit: Aiding the Development of Context-enabled Applications. In: *Proceedings of the SIGCHI conference on Human factors in computingsystems*, 1999. – ISBN 0–201–48559–1

[SE05] Steinmetz, R. ; (Edits.), K. W.: *Peer-to-Peer Systems and Applications*. Springer, 2005

[SG86] Schlimmer, J. ; Granger, R.: Beyond Incremental Processing: Tracking Concept Drift. In: *Fifth National Conference on Artificial Intelligence - AAAI*, 1986

[Sha01] Shannon, C.: A Mathematical Theory of Communication. In: *Mobile Computing and Communications Review* 5 (2001), Nr. 1

[SN04] Steinmetz, R. ; Nahrstedt, K.: *Multimedia Systems*. Springer, 2004

[Ste00] Steinmetz, R.: *Multimedia-Technologie: Grundlagen, Komponenten und Systeme*. Springer Verlag, 2000. – 3. Auflage

[Tsy04] Tsymbal, A.: The Problem of Concept Drift: Definitions and Related Work / Technical Report TCD-CS-2004-15, Computer Science Department, Trinity College, Dublin. 2004. – Forschungsbericht. – https://www.cs.tcd.ie/publications/tech-reports/reports.04/TCD-CS-2004-15.pdf

[VGPK99] Veizades, J. ; Guttman, E. ; Perkins, C. ; Kaplan, S.: *Service Location Protocol, Version 2*. 1999. – RFC 2608

[VRV05] Vallée, M. ; Ramparany, F. ; Vercouter, L.: Dynamic Service Composition in Ambient Intelligence Environments: A Multi-Agent Approach. In: *Proceeding of the First European Young Researcher Workshop on Service-OrientedComputing*, 2005

[WF01] Witten, I. ; Frank, E.: *Data Mining - Praktische Werkzeuge und Techniken für das maschinelle Lernen*. Carl Hanser Verlag, 2001

[Wid97] Widmer, G.: Tracking Context Changes through Meta-Learning. In: *Machine Learning* 27 (1997), Nr. 3

[WK96] Widmer, G. ; Kubat, M.: Learning in the Presence of Concept Drift and Hidden Contexts. In: *Machine Learning* 23 (1996), Nr. 1

[WZL06] Whitehouse, K. ; Zhao2, F. ; Liu, J.: *Semantic Streams: a Framework for Composable Semantic Interpretation of Sensor Data*. 2006. – http://www.cs.virginia.edu/papers/whitehouse06ewsn.pdf

[Yüc05] Yücel, M.: *Inspektion von Sensornetzen per PDA*. 2005. – http://people.ee.ethz.ch/~yuecelm/reports/sa_pdasensornetz.pdf

[ZHF+08] Zinnen, A. ; Hambach, S. ; Faatz, A. ; Lindstaedt, S. ; Beham, G. ; Godehard, E. ; Goertz, M. ; Lokaiczyk, R.: *Datenschutzfragen bei der Etablierung einer Arbeitsprozess-integrierten e-Learning-Lösung*. http://www.tmptech.net/publications/pdf/delfi-privacy.pdf. Version: 2008

[ZHF09] Zhang, W. ; Hansen, K. ; Fernandes, J.: *Towards OpenWorld Software Architectures with Semantic Architectural Styles, Components and Connectors*. 2009. – http://www.hydramiddleware.eu/hydra_papers/Towards_Open_World_Software_Architecture.pdf

[Zim06] Zimmer, T.: *QoC: Quality of Context - Improving the Performance of Context-Aware Applications*. 2006. – http://www.pervasive2006.org/ap/pervasive2006_adjunct_4E.pdf

[Zip09] Zipfel, T.: *Sensorbasierende Verfügbarkeitserkennung für eine Kontext-Sensitive Kommunikationsplattform*, Technische Universität Darmstadt, Diplomarbeit, 2009

Publikationen des Autors

[SAGS06] Schmitt, J. ; Ackermann, R. ; Goertz, M. ; Steinmetz, R.: VoIP-Sicherheit Status Quo und neue Aspekte. In: *D-A-CH 2006*, 2006

[Sch04] Schmitt, J.: *Dimplomarbeit. Extension of CPL for context-aware communication services.*, Technische Universität Darmstadt, Diplomarbeit, 2004

[SGP+05] Schmitt, J. ; Görtz, M. ; Pandit, K. ; Ackermann, R. ; Steinmetz, R. ; Lederer, T.: *Automatisierte Kontextbestimmung basierend auf selbstlernenden Algorithmen durch Beobachtung und Feedback des Nutzers zur Steuerung und Beeinflussung von Kommunikationsdiensten*. Erfindungsmeldung, 2005

[SHHS07] Schmitt, J. ; Hecker, T. ; Hollick, M. ; Steinmetz, R.: Dynamische Authentifizierung für Providerübergreifende VoIP-Kommunikation. In: *PIK - Praxis der Informationsverarbeitung und Kommunikation* 04 (2007), Dezember, Nr. 07. – ISSN 0930–5157

[SHK+06] Schmitt, J. ; Heckmann, O. ; König, A. ; Hollick, M. ; Steinmetz, R.: *ENUM-Erweiterung zur Sicherung der Kommunikation zwischen VoIP-Infrastrukturen*. Essener Workshop zur Netzwerksicherheit (EWNS) 2006, Oktober 2006

[SHRS08] Schmitt, J. ; Hollick, M. ; Roos, C. ; Steinmetz, R.: Adapting the User Context in Realtime: Tailoring Online Machine Learning Algorithms to Ambient Computing. In: *Mobile Networks and Applications* Volume 13, Number 6 / Dezember 2008 (2008), Oct. – ISSN 1383–469X (Print) 1572–8153 (Online)

[SHS07] Schmitt, J. ; Hollick, M. ; Steinmetz, R.: Der Assistent im Hintergrund: Adaptives Kommunikationsmanagement durch Lernen vom Nutzer. In: *PIK II/2007 - Current Trends in Network and Service Management* (2007), April

[SKHS07] Schmitt, J. ; Kropff, M. ; Hollick, M. ; Steinmetz, R.: The Virtual Assistant: Framework and Algorithms for User-Adaptive Communication Management / TU-Darmstadt. Version: Dezember 2007. ftp://ftp.kom.tu-darmstadt.de/pub/TR/KOM-TR-2007-08.pdf. 2007 (KOM-TR-2007-08). – Forschungsbericht

[SKR+08] Schmitt, J. ; Kropff, M. ; Reinhardt, A. ; Hollick, M. ; Schäfer, C. ; Remetter, F. ; Steinmetz, R.: An Extensible Framework for Context-aware Communication Management Using Heterogeneous Sensor Networks / TU Darmstadt. Version: November 2008. ftp://ftp.kom.tu-darmstadt.de/pub/TR/KOM-TR-2008-08.pdf. 2008 (TR-KOM-2008-08). – Forschungsbericht

[SKRH09] Schmitt, J. ; Kropff, M. ; Reinhardt, A. ; Hollick, M.: *ContextFramework.KOM - Eine offene Middleware zur Integration heterogener Sensoren in eine Kontext-sensitive Kommunikationsplattform*. Software Demonstration, Nominiert für KuVS Software Preis, Feb 2009. – ftp://www.kom.tu-darmstadt.de/pub/papers/SKR09-kuvs-sw-preis.pdf

[GASS04] Goertz, M. ; Ackermann, R. ; Schmitt, J. ; Steinmetz, R.: Context-aware communication services: A Framework for Building Enhanced IP Telephony Services. In: *International Conference on Computer Communications and Networks ICCCN 2004* (2004)

[KHSS06] König, A. ; Hollick, M. ; Schmitt, J. ; Steinmetz, R.: *Sicherheit und Verfügbarkeit in mobilen Ad hoc Netzen - Ein geographischer, schichtenübergreifender Ansatz*. 2. Essener Workshop Ñeue Herausforderungen in der Netzsicherheit;, Oct 2006

[RCH+10] Reinhardt, A. ; Christin, D. ; Hollick, M. ; Schmitt, J. ; Mogre, P. ; Steinmetz, R.: *Trimming the Tree: Tailoring Adaptive Huffman Coding to Wireless Sensor Networks*. To appear in: Proceedings of the 7th European Conference on Wireless Sensor Networks (EWSN 2010), Feb 2010

[RHG+08] Reinhardt, A. ; Hennecke, J. ; Gottwald, S. ; Kropff, M. ; Schmitt, J. ; Hollick, M. ; Steinmetz, R.: Tubicles: Heterogeneous Wireless Sensor Nodes - Testbed Objectives and Assembly Instructions / Technische Universität Darmstadt. 2008 (TR-KOM-2008-09). – Forschungsbericht. – ftp://ftp.kom.tu-darmstadt.de/pub/TR/KOM-TR-2008-09.pdf

[RSZ+10] Reinhardt, A. ; Schmitt, J. ; Zaid, F. ; Mogre, P. ; Kropff, M. ; Steinmetz, R.: *Towards Seamless Binding of Context-aware Services to Ubiquitous Information Sources*. To appear in: Proceedings of the 4th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS-2010), Feb 2010

Betreute Studien-, Bachelor- und Diplomarbeiten

[Bac08] Bachvarov, V.: *Information Preprocessing and Context Estimation Based on Sensor Meta Description*, Technische Universität Darmstadt, Diplomarbeit, 2008

[Bil07] Bill, T.: *Analyse und Erweiterung von Schnittstellen zur nutzerspezifischen Anrufverarbeitung*, Technische Universität Darmstadt, Diplomarbeit, 2007

[Est06] Esteve, C.: *Fixed-mobile convergence in TISPAN/3GPP IMS. Conception and evaluation of systems for seamless vertical handover*, Technische Universität Darmstadt, Diplomarbeit, 2006

[Fre08] Freudenreich, T.: *Dynamic Integration of Mobile Sensing Devices with Transparent Online Logging Capabilities into a Context-aware Environment*, Technische Universität Darmstadt, Bachelorarbeit, 2008

[Hec07] Hecker, T.: *ENUM-Erweiterung zur Sicherung der Kommunikation zwischen VoIP Infrastrukturen*, Technische Universität Darmstadt, Diplomarbeit, 2007

[Lei07] Leipold, F.: *Sensor, Service and Information Discovery in Heterogeneous Sensor Networks*, Technische Universität Darmstadt, Diplomarbeit, 2007

[Pri05] Priebe, M.: *Analysis and evaluation of machine learning algorithms for context-aware call processing in Voice-over-IP networks*, Technische Universität Darmstadt, Diplomarbeit, 2005

[Rem07] Remetter, F.: *Implementation and Evaluation of Middlewarebased Sensors on Embedded Systems*, Technische Universität Darmstadt, Diplomarbeit, 2007

[Roo06] Roos, C.: *Erweiterung und Anpassung des FLORA Verfahrens zur Integration in einen selbstlernenden Kommunikationsdienst*, Technische Universität Darmstadt, Studienarbeit, 2006

[Ros08] Rosenstock, L.: *Social Network Analysis for Context-Aware Communication Systems*, Technische Universität Darmstadt, Bachelorarbeit, 2008

[Sch07] Schäfer, C.: *Communication and Information Processing in Semantic Sensor Networks*, Technische Universität Darmstadt, Diplomarbeit, 2007

[Vit09] Vitanyi, A.: *An Interest Based Search and Adaptive Cache Engine in Semantic Sensor Networks*, Technische Universität Darmstadt, Diplomarbeit, 2009

A Anhang

Im Anhang finden sich detaillierte Beschreibungen über verwandte Forschungsarbeiten, betrachtete Basistechnologien zur Kommunikation, Rahmenwerke zur Kommunikation, sowie über Technologien zur semantischen Beschreibung von Netzdiensten. Des Weiteren finden sich hier Umsetzungsdetails, sowie weiterführende Ergebnisse der Analysen der Lernverfahren.

A.1 Detaillierte Beschreibung der verwandten Forschungsarbeiten

Die verwandten Arbeiten, welche in der Tabelle 2.1 verglichen wurden, werden in diesem Abschnitt genauer erläutert.

A.1.1 Context Toolkit

Das *Context Toolkit* [DAS99] ist ein bekanntes, Rahmenwerk zur Kontextverarbeitung. Aus dem Context Toolkit Projekt entstammen eine Reihe bekannter Arbeiten und häufig verwendeter Definitionen zum Thema Kontext.

Das Context Toolkit bietet eine Infrastruktur zur Erfassung und Aggregation von Sensordaten in heterogenen Umgebungen und zur Verteilung von Kontextinformationen. Es setzt auf Elemente wie beispielsweise Broker zur Kontextverarbeitung mit syntaktischer Query, sowie statisch definierbare Ereignisse. Zur korrekten Verarbeitung müssen Sensoren beziehungsweise Sensortypen dem System vorab bekannt sein. Dadurch ist es weit weniger auf dynamische, unbekannte Umgebungen anwendbar und schwieriger auf komplexe Anforderungen seitens der Nutzer anpassbar, als es in dieser Arbeit gefordert wird. Die Arbeiten zu dem Projekt liegen zudem schon einige Jahre zurück und ziehen daher keinen Vorteil aus leistungsfähiger gewordenen Endgeräten, sowie heute verfügbaren Technologien wie beispielsweise Webservices, semantischen Sensorbeschreibungen oder anpassungsfähigen Entscheidungsmodellen.

A.1.2 Context-Aware and Location Systems

Das Projekt *Context-Aware and Location Systems* (CALAIS) [Nelson1998] weist Überschneidungen mit dem Teilaspekt der Informationsaggregation dieser Arbeit auf. So soll das angestrebte Anwendungsszenario ebenfalls heterogene und dynamisch wechselnde Informationsquellen aufweisen, welche anhand ihrer Beschreibung herangezogen und verarbeitet werden sollen. Es werden zudem eine Reihe von Informationsquellen genannt, welche auch für dieses Anwendungsszenario Verwendung finden sollen (beispielsweise eine Webcam als Bewegungssensor oder die Überwachung der Aktivität am Arbeitsplatz des Teilnehmers). Die Verwendung eines Brokers zur Auffindung von Informationsquellen, ist ebenfalls eine Option für die Architektur in dieser Arbeit. Unterschiede lassen sich ebenfalls wieder im Fokus und in der konkreten Umsetzung finden. Das CALAIS Projekt behandelt vorwiegend das Problem der Lokationsbestimmung, nutzt statische Methoden zur Auswertung der Daten und nutzt ausschließlich syntaktische Beschreibungselemente zur Beschreibung der Informationsquellen.

A.1.3 Context Aware Software

Die Arbeit *Context Aware Software* [Pas01] befasst sich mit einem Rahmenwerk (mit der Bezeichnung *Context Information Service – CIS*), welches im Prinzip ebenfalls alle Ebenen der Kontextverarbeitung adressiert. Ein Anwendungsszenario von zentraler Rolle, ist die Umsetzung eines Systems zur Anzeige von Daten beziehungsweise Notizen, welche in einer bestimmten Situation beziehungsweise in dem jeweiligen Kontext relevant sein könnten. Hierzu baut dieses System auf einem CIS auf, einer Infrastruktur zur Erfassung und Verarbeitung von Kontextinformationen. Die Arbeit gibt einen sehr guten Überblick über die zu diesem Zeitpunkt existierenden Systeme zur Kontextnutzung und deren Fähigkeiten. Zudem liefert diese Arbeit eine Reihe neuer, interessanter Anwendungsszenarien, sowie Ideen zur Verteilung von Diensten zur Kontextbestimmung in globalen Szenarien. Ebenso spielt die

Akzeptanz der Nutzer für ein solches autonom agierendes System und die Reduzierung der Komplexität der Nutzerschnittstelle eine zentrale Rolle in dieser Arbeit. Die Verarbeitung erfolgt über eine Reihe statisch konfigurierbarer Mechanismen zur Aggregation, Fusion und Auswertung der Kontextinformationen. Daher wird eine automatisierte, adaptive Bestimmung und Auswertung relevanter Sensoren im Rahmen von CIS nicht behandelt.

A.1.4 Technology for Enabling Awareness

Das *Technology for Enabling Awareness*(TEA) [Lae99, Sch02c] Projekt hat das Ziel, den Kontext einer Person zu klassifizieren. Diese Bestimmung basiert auf der Auswertung mehrerer Informationsquellen auf einem Gerät, welches diese Person mit sich führt. Die Auswertung selbst wird durch eine Reihe von Schritten durchgeführt. Nach einer Vorverarbeitung der Daten werden diese durch ein KSOM-Verfahren (Kohonen Self-Organizing-Map) verarbeitet. Ein KSOM-Modell ist ein Verfahren zur Bildung von Clustern, welches auf einem einschichtigen Neuronalen Netz basiert. Dieses KSOM-Modell wird durch unüberwachtes Lernen (siehe auch 5.3.2 trainiert und dient zur Unterscheidung der gewonnenen Menge von Sensordaten in unterschiedliche Cluster. Ein Cluster entspricht zwar einem Kontext, jedoch wird in dem TEA-Projekt in einem weiteren Schritt über ein HMM (Hidden Markov Modell) überprüft, ob der Übergang von dem letzten bestimmten Kontext, in den neuen Kontext prinzipiell möglich ist. Dieses HM Modell ist durch Feedback an den Nutzer anpassbar, welcher letztlich auch durch das Feedback den Bezeichner (Label) für den jeweiligen Kontext festlegt.

Mit diesen Eigenschaften ist das TEA-Projekt im Bereich der anpassungsfähigen Entscheidungsmodelle zur Kontextbestimmung verwandt zu dieser Arbeit. Unüberwachte Lernverfahren wie KSOM sind jedoch relativ langsam in der Adaption eines Konzeptes. Das Modell teilt den Lernraum in Gruppen (engl. *Cluster*) auf. Bei diesem Ansatz sind oft viele Trainingssequenzen notwendig und ob letztendlich die gewünschten Gruppen beziehungsweise die gewünschte Anzahl von Gruppen entsteht, ist nicht gesichert oder muss manuell vordefiniert werden. Die Einschränkung auf lokal verfügbare Informationsquellen bringt zudem den Vorteil, keine fehlenden Sensorwerte berücksichtigen zu müssen.

A.1.5 CenceMe

Das Projekt *CenceMe* ist Teil des MetroSense Projektes. Im Rahmen des MetroSense Projektes wurde eine Reihe von Systemen zur Nutzung von Sensoren auf mobilen Endgeräten entwickelt. Unter anderem liefert die Arbeit von Eisenman [Eis08] einen Ansatz zur gemeinsamen Nutzung von Sensoren innerhalb einer Gruppe von mehreren Endgeräten in der Umgebung. Im Rahmen von CenceMe wurde eine Anwendung für das Iphone entwickelt, welche die dort verfügbaren Sensoren nutzbar macht. Vergleichbar zu dem Ansatz von TEA aus Abschnitt A.1.4 können diese Sensoren lokal auf dem Endgerät ausgewertet und zur Zustandserkennung genutzt werden. Dieser Zustand kann für dritte nutzbar gemacht werden, beispielsweise zur Anzeige innerhalb von Portalen für Soziale-Netzwerke.

A.1.6 MyConnector

Das Projekt *MyConnector* [DKS06a] hat ebenfalls zum Ziel, Kommunikationsmanagement zu optimieren und verfolgt eine relativ ähnliche Idee wie diese Arbeit. Über Sensoren wird das Benutzerverhalten beobachtet und per Feedback und einen Lernalgorithmus wird ein Entscheidungsmodell trainiert. Sogenannte *Context Cues* (vergleichbar zu Kontextinformationen) werden ausgewertet, um beispielsweise Aussagen über die Verfügbarkeit eines Nutzers zu gewinnen. Der Ansatz von MyConnector sieht vor, die Kontextinformationen und Kontexte zentralisiert, im *Core Connector* zu verarbeiten. Dort fließen Informationen der Sensoren zusammen, wie z.B. aktuelle Einträge im Kalender eines Nutzers oder die Aktivität an seinem Rechner. Diese Sensoren oder Informationsquellen sind jedoch statisch beziehungsweise vordefiniert. Die Architektur des Netzes zur Informationsgewinnung über die Sensoren ist zentralisiert aufgebaut. Im Unterschied zu dieser Arbeit ist ein einfaches Hinzufügen unbekannter Informationsquellen in den Verarbeitungsprozess nicht vorgesehen. Dadurch können zwar einige Probleme vermieden werden, jedoch limitiert diese Vorgehensweise den Ansatz auf zuvor bekannte Sensoren. Unter anderem muss die Relevanz eines Sensors nicht zur Laufzeit bestimmt werden und der gesamte Suchprozess kann gezielt auf einzelne vorab bekannte Sensoren durchgeführt und angepasst werden. Bei MyConnector wurde ausschließlich ein Bayessches Netz zum Aufbau des Entscheidungsmodells verwendet, wobei einige Eigenschaften wie beispielsweise Konzeptänderungen (siehe Abschnitt 5.3.1) nicht beachtet wurden.

A.1.7 An Architecture for Context Prediction

Die Dissertation von R. Mayrhofer zum Thema *An Architecture for Context Prediction* [May04] verfolgt vergleichbare Ziele wie diese Arbeit. Der Fokus wurde jedoch auf die Auswertung der Kontextdaten gelegt und um die Vorhersage von Kontexten erweitert, welche für die Anwendungsszenarien in dieser Arbeit nicht benötigt werden. Die Anforderungen, wie sie Mayrhofer für die Auswertung angibt, sind relativ ähnlich zu den Anforderungen, welche sich für die Anwendungsszenarien dieser Arbeit ableiten lassen. Mayrhofer trifft in seiner Arbeit jedoch die Entscheidung, dass sogenannte überwachte Lernverfahren (*supervised Learning*, siehe Abschnitt 5.3.2) der Anforderung „der Dringlichkeit" widersprechen würde. Somit verwendet Mayrhofer unüberwachte Lernverfahren zur Kontextauswertung, was zu denselben Problemen führen kann, wie sie bereits in Abschnitt A.1.4 erläutert wurden. In einer späteren Phase seines Systems muss der Nutzer jedoch seinen aktuellen Kontext benennen. Dieser Schritt ist vergleichbar mit dem Feedback dieser Arbeit. Zur späteren Kontextvorhersage verwendet er jedoch ebenfalls Verfahren des überwachten Lernens.

A.1.8 Umgebungsmodelle für Mobile Kontextbezogene Systeme

Der Sonderforschungsbereich NEXUS der Universität Stuttgart mit dem Untertitel *Umgebungsmodelle für Mobile Kontextbezogene Systeme*, spaltet sich in eine Vielzahl von Teilprojekten auf. Mit diesen Teilprojekten spannt NEXUS seine Forschungsaktivitäten über alle Ebenen, das heißt von der Aggregation der Kontextinformation bis hin zu konkreten Anwendungen, welche den Kontext nutzen, auf. Aus den Teilprojekten ist eine große Anzahl von Publikationen zu entstanden. Viele davon behandeln Themen im Bereich Lokationsbestimmung, der Aggregation, Modellierung und Verteilung von Kontextinformationen, sowie der Nutzung von Kontextinformationen im Bereich der *Augmented Reality*. Als Beispiele für Sensoren wurde der Prototyp einer *ContextCube* [BBHS] entwickelt, welcher physikalische Sensoren zur Messung von Temperatur, Luftfeuchtigkeit, Helligkeit, etc., zu einer multi-funktionalen Sensoreinheit zusammenführt, und diese als Dienst im Netz bereitstellt. Die ContextCube ist vergleichbar mit den Plattformen, die in dieser Arbeit als Grundlage zur Informationserfassung dienen sollen (siehe hierzu unsere Arbeit über den *Tubicle* [RHG+08]).

Die Teilprojekte wurden bisher jedoch nicht zu einem Gesamtszenario zusammengeführt, welches diese vielen Ergebnisse nutzt und greifbar macht. Zudem wird zur Kontextbestimmung eine Auswertung von Ontologien und Kontext-Modellen (*Reasoning*, siehe Abschnitt 5.1.1) durchgeführt. Zur statischen Modellierung dieser Modelle wurden zwar Werkzeuge entwickelt, eine Anpassung eines Modells durch den Nutzer selbst, ist jedoch relativ aufwendig und widerspricht der Anforderung einer einfachen Nutzerschnittstelle.

A.1.9 Ambient Intelligence for the networked home environment

Unter der Bezeichnung AMIGO [VRV05] wurde das EU-Projekt *Ambient Intelligence for the networked home environment* bis Anfang des Jahres 2008 durchgeführt. Dabei war eine intelligente Steuerung von Heim- und Multimediageräten im Fokus der Forschungsarbeiten. Die Anforderungen und die gewählte Architektur sind in mehreren Punkten vergleichbar zu dieser Arbeit. Durch das Erfassen und Auswerten von Wissen über die aktuelle Situation soll eine intelligente Steuerung ermöglicht werden. Hierzu sollen Sensoren und Aktoren dynamisch in das Rahmenwerk eingebunden werden können. Das Design des AMIGO Systems basiert auch auf SoA in Kombination mit agentenbasierten Konzepten. Um den Aufwand seitens des Nutzers gering zu halten, hat AMIGO ebenfalls zum Ziel Endgeräte dynamisch an das System anzubinden. Hierzu setzt AMIGO auf OWL und AMIGO-S (eine Erweiterung von OWL-S) zur Formulierung der Ontologie und der Beschreibungen. Der Fokus des Projektes auf Home-Automation führt jedoch zu einer unterschiedlichen Gewichtung der Arbeiten. So spielen beispielsweise im AMIGO Projekt die Erstellung von umfassenden Nutzer- und Geräteprofilen, sowie die Interaktion zwischen Endgeräten als Aktoren eine zentrale Rolle.

Das Teilprojekt AMIGO D4.7 *Intelligent User Services* dient der Auswertung der gesammelten Kontextinformationen und ist damit von besonderer Relevanz für die Arbeiten in Kapitel 5. In den meisten Bereichen wird ein statisches Modell als Grundlage zur Entscheidungsfindung (Reasoning) genutzt. Dies begründet sich unter anderem auch darauf, dass viele Entscheidungen gerade im Bereich der Interoperabilität zwischen Endgeräten (Kompatibilität ihrer Fähigkeiten) ausgewertet werden müssen. Im Teilprojekt *Intelligent User Services* hingegen, wird eine Art hybrider Ansatz gewählt. Neben einem statischen Modell als Basis zur Auswertung des Nutzerprofils, wird auch ein dynamisches Modell genutzt. Dieses Modell dient unter anderem zur Bestimmung von Nutzerpräferenzen (bei-

spielsweise welche Filme er bevorzugt anschauen würde) und lässt sich durch Feedback des Nutzers an den Nutzer anpassen. Als dynamisch anpassbares Modell wird hierbei eine Kombination aus CBR (Case Based Reasoner) und SVM (Support Vector Machine) gewählt. Dadurch ergibt sich eine Art Meta-Classifier welcher die Ergebnisse beider Verfahren gewichtet und miteinander verrechnet, um eine Entscheidung zu treffen. Die Gewichte werden abhängig von der Qualität der Resultate berechnet, welche sich bei der Anwendung des Modells auf die zurückliegenden Ereignisse ergeben. Es liegen zwar auch Ergebnisse und Aussagen über die Nutzung von dynamischen Modellen zur Entscheidungsfindung vor, jedoch nimmt dieser Bereich eher eine Nebenrolle ein. So wurde beispielsweise nur dieser eine Ansatz dokumentiert und es fehlen daher umfassende Vergleiche mit anderen Verfahren, anderen Ansätzen oder Optimierungsmöglichkeiten.

A.1.10 Managing Context Data for Smart Spaces

Das Projekt *Managing Context Data for Smart Spaces* (MUSE) [CM00] verfolgt das Ziel, Kontextinformationen durch Fusion (zur Erläuterung des Begriffs siehe 2.4.7) mittels semantischer Informationen über die Sensordaten, sowie über Bayessche Netze zu verarbeiten. Dieses Projekt verfolgt damit eine Reihe ähnlicher Ziele im Bereich der Informationsaggregation. Der Fokus der Arbeit liegt jedoch weniger auf der Entscheidungsfindung als in der Reduzierung von Anfragen. Das Bayessche Netz zur Fusion der Sensordaten, welches dabei Verwendung findet, ist jedoch statisch, beziehungsweise es muss manuell erstellt werden. Damit fehlt dem System die Fähigkeit, sich an neue, unbekannte Typen von Sensoren und Sensordaten anzupassen, sowie den Entscheidungsprozess mit geringem Aufwand an den Nutzer anzupassen.

A.1.11 Web Presence for the Real World

Im Projekt Web Presence for the Real World (CoolTown) [KBM+02] senden Elemente aller Art (unter anderem auch andere Personen) regelmäßig ihre Adresse in Form einer *Uniform Resource Locator* URL per Funk als von *Beacons*. Ein Gerät, das von einer Person mit sich geführt wird, empfängt diese URLs von Informationsquellen aus seiner unmittelbaren Umgebung. In einem weiteren Schritt kann dieses Gerät anhand einer HTTP-Anfrage über die erhaltene URL semantische Zusatzinformationen beziehen. Diese semantischen Informationen werden ausgewertet, um zu bestimmen, ob das Element, beziehungsweise der dazu angebotene Dienst im Netz oder die Informationen über das Element für den Nutzer relevant sind. Diese Auswertung von personenbezogenen Informationen aus der Umgebung mittels semantischer Zusatzinformationen, verfolgt damit ähnliche Grundgedanken, wie diese Arbeit. In dem Projekt wird allerdings nur eine Relevanzbestimmung zur Anzeige von potentiell interessanten Informationen und Diensten durchgeführt. Daher unterscheiden sich die Rahmenbedingungen stark. So wird beispielsweise keine adaptive Entscheidungsfindung unter Einhaltung zeitlicher Anforderungen durchgeführt.

A.1.12 Notification Plattform

Microsoft Research hat einige Arbeiten im Umfeld des Projektes *Notification Platform* publiziert, welche für diese Arbeit relevant sind. Ziel dieser Arbeiten ist es den Nutzer eines Computers möglichst wenig in seinem Arbeitsablauf zu stören. Durch die Bestimmung des Zustandes eines Nutzers soll ermittelt werden, ob beispielsweise die Anzeige eingehender E-Mails vom Nutzer als störend empfunden wird [HKPH03]. Dazu werden vor allem Informationen und Zustände des Computers selbst ausgewertet. Die Bestimmung und Erfassung relevanter, externer Informationen, sowie die Aufgaben und Anforderungen die dadurch entstehen, stehen jedoch nicht im Fokus der Arbeiten von Microsoft Research. Wie auch in dieser Arbeit wurde die Anpassung von Modellen an den Nutzer als notwendige Anforderung an das System identifiziert [HA03]. Ähnlich wie bei der Arbeit von Mayrhofer wird eine Vorhersage von Zuständen angestrebt [HKKJ02].

A.1.13 Smart Environments

Das MUNDO Projekt [Ait07] wurde in der Gruppe *Smart Environments* des Fachgebiets Telekooperation an der TU-Darmstadt entwickelt und hat den Fokus eine Architektur aufzubauen, welche *ubiquitous computing* ermöglicht. Der Begriff des *ubiquitous computing* folgt der gleichen Annahme, wie sie auch für diese Arbeit genutzt wird, nämlich in Zukunft eine Vielzahl verteilter und vernetzter Geräte zur Verfügung stehen zu haben, die unter anderem

Kontextinformationen bereitstellen. Die in dem Projekt angestrebte Infrastruktur ermöglicht es, die Funktionalitäten der Geräte durch ein Konzept, welches dem von Webservices ähnelt, nutzbar zu machen. MundoCore ist eine Middleware zur Kommunikation zwischen den Diensten und stellt somit eine Kernkomponente des Mundo Projektes dar. MundoCore sieht unter anderem vor die Funktionalitäten eines Gerätes über eine entfernt aufrufbare Schnittstelle (RPC) für andere Systeme im Netz nutzbar zu machen. Analog zu anderen Konzepte für Webservices ist es hierbei möglich diese Schnittstelle zu beschreiben, zu veröffentlichen und zu suchen. Als Unterschied zu klassischen Webservices liegt der Fokus dieses Projektes jedoch darauf, auch Systeme mit geringer Leistungsfähigkeit, sowie über unterschiedliche Netztechnologien hinweg miteinander zu verbinden. Das Teilprojekt MundoCore, wurde zeitweise im Rahmen dieser Arbeit herangezogen, um Informationsquellen zu adressieren. Die Leistungsfähigkeit dieses MundoCore Systems ist jedoch auf die Kommunikation zu den Geräten fokussiert und liefert noch keine Mechanismen zur semantischen Beschreibung von Diensten.

A.1.14 Suggested Upper Merged Ontology, OntoSensor

Die Suggested Upper Merged Ontology (SUMO) [PNL02] wird angewendet, um eine übergeordnete Ontologie zu definieren, welche von anderen Ontologien referenziert werden kann. Diese SUMO wird erstellt, indem existierende Ontologien kombiniert werden. Die Arbeiten im Rahmen der SUMO verfolgen das Ziel, die Kommunikation über unterschiedliche Ontologien hinweg zu verbessern. Statt jeweils zwischen zwei unterschiedlichen Ontologien zu übersetzen, bildet die SUMO das verbindende Element zwischen den verschiedenen Ontologien. Die SUMO befasst sich somit mit Herausforderungen, die vergleichbar sind mit denen, wie sie im Bereich der Ontologie-Anpassung (siehe Abschnitt 4.5.4) dieser Arbeit behandelt werden. Die Nutzung einer SUMO könnte eine Basis für den Aufbau einer verteilten Ontologie für Sensoren sein und somit eine Suche über mehrere Service-Directories hinweg, sowie ohne globale Ontologie ermöglichen.

Die Arbeit Ontosensor von Russomanno et al. [RKT05] füllt quasi die Lücke zwischen einer allgemeinen, übergeordneten Ontologie (beispielsweise SUMO) und den verschiedenen, anwendungsspezifischen Sensornetzwerk Ontologien.

> *„OntoSensor can be viewed as middle-level ontology that extends the concepts from a high-level ontology (SUMO) and whose concepts can be used by more specialized ontologies that model specific domain sensors." [RKT05]*

Die in OntoSensor verwendete Ontologie bietet eine Klassifikation von Sensoren und ein Modell zur Beschreibung des Kontextes eines Sensors. Die Arbeit im Rahmen von OntoSensor (und damit auch die in diesem Rahmen erstellte Ontologie) befasst sich jedoch mit einem unterschiedlichen Anwendungsgebiet: Der Überwachung von Zuständen in Maschinen und Betrieben, wie beispielsweise chemische, mechanische oder elektromagnetische Messungen.

Russomanno et al. veröffentlichte eine prototypische Umsetzung ihrer Arbeit als Proof-of-Concept [RKT05]. Dieser Prototyp verwendete die OntoSensor Ontologie und Mica Motes von der Firma Crossbow[1]. Die vom Sensor generierten Daten werden um Referenzen auf die Ontologie erweitert. Hierzu ist jedoch Vorwissen über die Ontologie notwendig. In dieser Arbeit wird dieser Aspekt über eine vollständige Selbstbeschreibung des Sensors und seiner Daten, sowie gegebenenfalls durch eine Erweiterung der Ontologie behandelt. Die Suche wird über einen *Agenten* abgewickelt, welche ebenfalls die Ontologie nutzt um relevante Sensoren zu identifizieren. So ist es in diesem System ebenfalls möglich, Sensoren zu finden, welche beispielsweise in Relation zu einem Ort stehen oder Informationen über den durchschnittlichen Luftdruck in einem bestimmten Gebiet zu sammeln.

A.1.15 Publish/Subscribe Applied to Distributed Resource Scheduling

PADRES steht für *Publish/Subscribe Applied to Distributed Resource Scheduling* [LJ05] und bietet eine Middleware zur Verarbeitung von Zuständen und Ereignissen in Anwendungsszenarien wie Unternehmensprozessen, mit der Option auch Sensoren und Sensornetzwerke anzubinden[2]. In PADRES werden lokale Broker eingesetzte, welche vergleichbar mit dem Sensorverzeichnis aus dieser Arbeit sind. Die auf Java basierende Umsetzung verwendet RMI zur Kommunikation und bietet durch die Vernetzung der Broker eine Peer-to-Peer ähnliche Overlay-Struktur über die Elemente aus unterschiedlichen Netzen miteinander verbunden werden können.

[1] `http://www.xbow.com`

[2] `http://research.msrg.utoronto.ca/Padres/`

PADRES verwendet ein Inhalts-basiertes Publish-Subscribe Verfahren (*content-based Publish/Subscribe* (CPS)) zur Aggregation und Verteilung der Daten. Der Ansatz von PADRES, Sensoren über eine Suche zu finden, verfolgt in der Basis einen Pull-basierten Ansatz. Je nach Anzahl von Suchanfragen, Zustandsänderungen der Sensoren und Dynamik der Sensoren kann der eine oder der andere Ansatz vorteilhafter sein. Der Ansatz der Suche in dieser Arbeit kann jedoch bei Bedarf um weitere Mechanismen ergänzt werden, wie beispielsweise das aktive Vorhalten von Daten (wie in Abschnitt 4.7 erläutert), oder das Ansteuern von Publish/ Subscribe Mechanismen.

Ein Teil des PADRES Projekts behandelt die semantische Fusion von Daten. Über ein Verfahren werden Sensordaten zu höherwertigen Aussagen kombiniert, zum Beispiel kann aus den Werten (Regen:nein; Temperatur:kalt; Helligkeit:dunkel; Zeit:11.30) die Aussage (Wetter:bewölkt) abgeleitet werden. Diese Ableitung wird anhand der Sensorbeschreibung, sowie Regeln aus der Ontologie durchgeführt. Die Menge der Regeln kann hierbei durch die Anwendungen erweitert werden. Die Fusion von Daten kann sinnvoll eingesetzt werden um Vorwissen über die Aussage und Zusammenhänge von Daten zu repräsentieren. Diese Regeln sind jedoch statisch und müssen manuell erstellt werden (siehe auch Abschnitt 5.2.5).

A.1.16 Sensor Web Agent Platform

Die Arbeit Sensor Web Agent Platform (SWAP) von Moodley und Simonis beschreibt ein Rahmenwerk, um für unterschiedliche Anwendungsbereiche Sensoren über das Internet hinweg anzubinden und zu nutzen [MS06]. SWAP sieht dazu ebenfalls eine dienstorientierte Architektur vor, wobei der Begriff *Dienste* (wie er in dieser Arbeit genutzt wird) vergleichbar mit einem *Agenten* in SWAP ist (siehe auch Abschnitt 4.2.3 - JADE). SWAP sieht zusätzlich Einteilung der Agenten in die folgenden Ebenen vor:

- Der *Sensor Layer* kapselt die Informationsquellen (beispielsweise physikalischen Sensoren oder andere Dienste). Diese Ebene ist vergleichbar mit den Architekturelementen *Sensor* und *Sensor Basis* und aus Abschnitt 4.3 dieser Arbeit.
- In der dem Sensor Layer übergeordneten Ebene dient der *Knowledge Layer* zur Verarbeitung und Verteilung der Sensordaten. Diese Ebene entspricht den Funktionalitäten des Service-Directories, der Kontextdienstes und des Auswertungsdienstes.
- Auf der obersten Ebene, dem *Application Layer* sind Schnittstellen für Interaktionen mit Nutzern, Diensten und Anwendungen vorgesehen. Die Schnittstellen des Kontextdienstes und des Sensorverzeichnisses zur Nutzung von Kontext und Sensordaten aus dieser Arbeit sind dem Application Layer vergleichbar.

Moodley und Simonis heben dabei hervor, dass die Kommunikation zwischen den Agenten eine wesentliche Komponente für das gesamte System ist. Hierfür sind gut strukturierte, aussagekräftige Ontologien notwendig. Das Rahmenwerk „. . . must support thousands of ontologies across multiple end-user applications and in many rent domains" [MS06]. Ähnlich wie in den Arbeit *OntoSensor* und *Suggested Upper Merged Ontology* werden Ontologien auf unterschiedlichen Ebenen betrachtet und behandelt. SWAP nutzt hierzu ebenfalls Ontologien auf der Basis von OWL.

A.1.17 Heterogeneous physical devices in a distributed architecture

Das EU-Projekt Hydra beschäftigt sich mit der Erstellung einer Middleware zur Erfassung, Verarbeitung und Bereitstellung von Sensordaten [ZHF09]. Das Projekt verfolgt dabei ebenfalls das Ziel heterogene Systeme als Informationsquellen dynamisch anzubinden und setzt auch auf Technologien wie OSGI. Zur Beschreibung der Zusammenhänge und zur Auswertung von Anfragen wird eine Ontologie herangezogen. Zur semantischen Beschreibung von Schnittstellen verwenden sie ähnliche Konstrukte wie OWL-S. Erste Veröffentlichungen aus dem Projekt stammen aus 2008. Es wurde bisher jedoch nur eine einfache Testumgebung zur Kommunikation zwischen Diensten (der *Hydra Network Manager*) aus dem Projekt verfügbar gemacht.

A.2 Basistechnologien zur Kommunikation

In Abschnitt 4.2.2 wurden die Basistechnologien zur Kommunikation eingeführt. Dieser Abschnitt bietet detaillierte Beschreibungen der einzelnen Technologien.

Objektbasiert

Für Daten, welche innerhalb von Plattformen gleichen Typs ausgetauscht werden sollen, ist es die *Serialisierung* der Objekte, welche die Daten enthalten eine sehr effiziente Lösung. Hierzu werden die Daten in einen binären Datenstrom umgewandelt, dessen Struktur implizit durch Referenzierung auf die jeweiligen plattformspezifischen Objekte gegeben ist. Sofern das System, welches die Daten deserialisieren soll über die Objekte verfügt, kann dieses den Datenstrom einlesen und direkt in die entsprechenden Objekte umwandeln.

Binärcodiert

Eine weitere Form der binären Repräsentation von Daten ist die Trennung von Struktur und Inhalt der Daten. Über die Struktur beschreibt eine Art Maske oder Regelsatz mit deren Hilfe die Transformation der Daten in eine binäre Darstellung definiert wird (sowie deren Rücktransformation). Diese Trennung von Struktur und Inhalt ist ein effizienter und flexibler Ansatz, da einerseits Informationen über die Struktur nur einmal übertragen werden müssen und andererseits die Beschreibung Struktur eine plattformübergreifende Nutzung der binär codierten Daten ermöglicht. Beispiele für solche Ansätze zur Beschreibung der Struktur finden sich in Technologien wie ASN.1[3] oder IDL (siehe Abschnitt 4.2.3 CORBA) wieder. Entsprechende Ansätze zur Kodierung des Inhaltes finden sich in den Technologien *Basic Encoding Rules* (BER) für ASN.1 oder der *Common Data Representation* (CDR) für IDL wieder. Eine Nutzung solcher Ansätze erfordert jedoch Mechanismen und Aufwand zur Erstellung und Nutzung der Strukturbeschreibung, was außerdem zusätzliche Komplexität und erhöhtem Aufwand für die Entwickler mit sich bringt.

XML

Die *Extensible Markup Language* (XML) ist ein sehr häufig genutztes, textuelles und flexibles Format zum Austausch von Informationen. XML definiert dabei nur die Syntax. Die Bezeichner, Struktur oder Inhalte der Elemente in XML sind grundlegend frei wählbar. Um von verschieden Systemen aus, auf ein gemeinsames XML Dokument zurückzugreifen, sind jedoch gewisse Vorgaben notwendig. Für den Datenaustausch zwischen unterschiedlichen Systemen mittels XML wird daher häufig ein XML-Schema (XSD) zur Festlegung und Einschränkung der Bezeichner, der Struktur oder der Inhalte auf ein gemeinsames Format angewendet. Ein solches Schema kommt beispielsweise in XML-RPC, oder SOAP (siehe Abschnitt A.2) zur Spezifikation einer Dienst Anfrage per XML zum Einsatz.

SOAP-Nachricht

Der Name SOAP stammte ursprünglich von der Bezeichnung *Simple Object Access Protocol*. SOAP-Nachrichten basieren auf XML, und können auch zur Darstellung komplexer Objekte verwendet werden. Im Rahmen von SOAP-RPC Aufrufen werden SOAP-Nachrichten zum Transport der Parameter verwendet. SOAP-Nachrichten werden neben der Anwendung in SOAP-RPC auch in anderen Technologien zur Repräsentation komplexer Objekte in Form von XML-basierten Nachrichten verwendet.

Weitere Verfahren

Es existieren noch eine Reihe weiterer Ansätze, die häufig nur innerhalb bestimmter Rahmenwerke genutzt werden oder nur vereinzelt eingesetzt werden.

Die Java Architecture for XML Binding (JAXB) oder Programme wie XStream[4] ermöglichen es, Java-Objekte in XML-Form zu übertragen und umgekehrt. Ein Objekt wird hierbei an ein XML-Schema gebunden. Dies erfordert jedoch eine Erstellung oder Generierung des Schemas anhand des zu übertragenden Objektes, sowie gegebenenfalls die Übertragung des Schemas an das entfernte System.

Ein alternatives Austauschformat für Daten, welches noch zu erwähnen wäre, ist die JavaScript Object Notation (JSON). Es ist im Ansatz vergleichbar mit XML, ist jedoch kompakter und dient nur als Format zum Austausch von Daten.

[3] `http://www.asn1.org/`
[4] `http://xstream.codehaus.org/`

Datentransport

Lokale Kommunikation

Solange sich Dienste auf demselben System befinden, können Mechanismen zur Inter-Prozess Kommunikation (IPC) angewendet werden. Je nach Plattform existieren unterschiedliche Mechanismen, welche meist von der Plattform selbst zur Verfügung gestellt werden. Hierzu zählen Ansätze wie beispielsweise gemeinsam genutzter Speicher oder Pipes.

Socket

Sockets stellen meist die Grundlage für die Kommunikation über ein IP basiertes Netzwerk dar. Die direkte Kommunikation über Sockets erfolgt meist ohne weiteres Management oder Vorgaben direkt durch den Austausch der Daten über Datenströme. Sockets können im Prinzip unabhängig von der Plattform genutzt werden. Abhängigkeiten entstehen hierbei erst in der Wahl der Repräsentation (siehe Abschnitt A.2) der Daten. Für eine Nutzung eines Socket muss jedoch zu Beginn ein gemeinsames Protokoll zum Aufruf entfernter Methoden und zur Übertragung komplexer Objekte definiert werden. Dies kann zwar an dieser Stelle proprietär durchgeführt werden, es existieren jedoch standardisierte Ansätze, welche gerade diese Aspekte angehen. Diese Ansätze werden im Folgenden beschrieben.

HTTP

Das *Hypertext Transfer Protocol* (HTTP) ergänzt die reine Socketkommunikation durch ein Protokoll zur Datenübertragung. Zum Management der Datenübertragung werden hierzu die eigentlichen Nutzdaten nach einem (relativ einfachen) Schema um Zusatzinformationen ergänzt und übertragen. HTTP bietet zudem den Vorteil, relativ einfach auch in Kombination mit einigen Systemen wie Firewalls oder Proxys einsetzbar zu sein.

SIP

Das *Session Initiation Protocol* (SIP) ist vergleichbar mit dem Konzept von HTTP. SIP kommt aktuell im Bereich Internet-Telefonie und UMTS zum Einsatz. Ursprünglich wurde es als Protokoll für generischen Aufbau von Sitzungen konzipiert. Es kann prinzipiell auch für die Übertragung von Nutzdaten (MIME kodiert - analog zu HTTP) durch ein standardisiertes Verfahren genutzt werden. Das bedeutet, sofern ein Netzwerk so konzipiert und konfiguriert wurde, dass eine Verbindung mittels SIP (über NAT und Firewalls hinweg, beispielsweise durch Application Layer Gateways oder Border Gateway Controller) möglich ist. So können auch Nutzdaten auf prinzipiell dem gleichem Wege übertragen werden.

Gerade durch das Ziel dieser Arbeit ein System zu entwickeln, welches mit bestehenden Kommunikationssytemen interagieren soll, wird SIP als Transportprotokoll interessant. In diesen Nutzdaten kann der angedachte Aufruf entfernter Dienste gekapselt werden. Hierzu existieren Ansätze diese Aufrufe mittels SOAP zu kapseln und über SIP zu übertragen [Dea00]. In diesem Zusammenhang ist es jedoch erforderlich, dass keine Filter oder Systeme wie Gateways [Ack03] ausschließlich Telefonie-Verbindungen zulassen.

XMPP

Das *Extensible Messaging and Presence Protocol* (XMPP[5]) wurde im Rahmen des Instant-Messaging Systems Jabber entworfen und von der IETF standardisiert. XMPP ist vom Ansatz her vergleichbar mit SIP. Es dient der Nachrichten- und Dateiübertragung. XMPP nutzt jedoch ausschließlich TCP und verschickt auf XML basierende Nachrichten.

JMS

Die *Java Message Service* (JMS[6]) API ist eine von Sun herausgegebene Spezifikation für die Kommunikation zwischen Systemen oder Diensten. Über die Spezifikation wird eine Abstraktionsschicht definiert, über die es Möglich ist, über Nachrichten in einem Netz zu kommunizieren. Ein System, welches JMS implementiert, kann auch als Nachrichten-orientierte Middleware bezeichnet werden (siehe Abschnitt 3.3). Im Fokus von JMS steht die asynchrone Kommunikation über Nachrichten mittels Publish/Subscribe oder Point-to-Point.

[5] http://xmpp.org/rfcs/rfc3920.html
[6] http://java.sun.com/products/jms/

Es gibt eine Vielzahl von Implementierungen, welche JMS umsetzen. Die Spezifikation lässt offen, wie der Transport der Daten zu erfolgen hat. Systeme wie beispielsweise Apache ActiveMQ[7] oder OpenJMS[8] setzen hierfür auf Verfahren wie Socket (RMI), HTTP oder XMPP.

Gemeinsam definierte Nachrichtenformate und typisierte Schnittstellen wirken sich jedoch nachteilig auf die Nutzung in dem angestrebten Szenario aus. Der Fokus liegt verstärkt in dem asynchronen, aber zuverlässigen Verteilen von Nachrichten in einem Netzwerk (eine Übermittlung von Nachrichten in Echtzeit ist nicht in der Spezifikation vorgesehen). Daher sind bei JMS je nach Implementierung erhöhte Verzögerung zu erwarten und somit ist JMS für die Verwendung in echtzeitsensitiven Systemen nicht geeignet.

Anfrage Spezifikation

REST

Das *Representational State Transfer* (REST[9]) Konzept sieht vor, auf Basis der von HTTP angebotenen Möglichkeiten, Dienste aufzurufen. Dieser Ansatz beschreibt somit eine einfache, leichtgewichtige, skalierbare Möglichkeit Dienste im Netz verfügbar zu machen und zu nutzen. Der Fokus liegt weniger auf komplexen Interaktionen zwischen Diensten, als auf dem Anbieten und Verwalten von Ressourcen im Netz. Eine Ressource wird mittels URI adressiert und kann über die HTTP Methoden GET, POST, PUT und DELETE geladen, verändert, erstellt oder gelöscht werden.

Rest ist weniger als Standard zu verstehen, sonder eher als eine Vorgabe für einen Architekturstil. Daher wird nicht spezifiziert, wie genau ein Aufruf gestaltet sein muss oder insbesondere wie komplexe Objekte repräsentiert werden. Ein REST Konzept wird daher oft mit XML, JSON oder SOAP zur Repräsentation komplexer Objekte kombiniert.

IIOP

Das *Internet Inter-ORB Protocol* (IIOP) wurde im Rahmen von CORBA spezifiziert. Das Protokoll ermöglicht es, Objekte plattformunabhängig, als binären Datenstrom zu übertragen. Hierzu werden die Beschreibungen der Schnittstellen und der Objekte getrennt von deren Implementierung verarbeitet. Die Beschreibung wird in ein gemeinsames Meta-Format übertragen - der Interface Definition Language (IDL). Diese Beschreibung spezifiziert genau die Schnittstelle und die Objekte, so dass ein Aufruf über diese Beschreibung in einen entsprechenden binären Datenstrom (*Common Data Representation* - CDR) transformiert werden kann. Der aufgerufene Dienst kann anschließend den Datenstrom anhand der Beschreibung in Objekte, welche seiner Plattform entsprechen, umwandeln. Dieser Ansatz ermöglicht eine Kombination von Plattformunabhängigkeit und Effizienz. Es müssen jedoch auch Einschränkungen im Bereich Flexibilität und Portabilität des Codes hingenommen werden, da IDL im Prinzip den *kleinsten gemeinsamen Nenner* zwischen den unterstützten Plattformen beschreibt. Hinzu kommt der Aufwand zur Erzeugung einer IDL-Beschreibung.

RMI

Java beinhaltet zum Aufruf entfernter Methoden das *Remote Method Invocation* (RMI) Protokoll. Grundlegend basiert das RMI Protokoll auf dem *Stub*-Prinzip, bildet also entfernt verfügbare Schnittstellen auf lokale Objekte ab. In der Basiskonfiguration von RMI werden Aufrufe und die Objekte innerhalb dieser Aufrufe mittels Serialisierung als binärer Datenstrom übertragen. In dieser Konfiguration können jedoch nur Verbindungen zwischen Diensten erstellt werden, welche auf Java basieren. Jedoch existiert mit RMI-IIOP[10] inzwischen die Möglichkeit die Daten mittels IIOP zu übertragen, welches die Verbindung zwischen Java-Diensten und CORBA-Diensten ermöglicht. Andererseits hat eine reine RMI Kommunikation jedoch Vorteile in den Bereichen Geschwindigkeit, Speichermanagement (Garbage Collector) und Portabilität von Programmcode.

R-OSGI

R-OSGi[11] steht für *Remote-OSGI*. Wie der Name schon vermuten lässt, ermöglicht dieser Ansatz die Kommunikation zwischen OSGI-Diensten (siehe hierzu Abschnitt 3.3). Durch den Einsatz von R-OSGI ist es möglich innerhalb von OSGI ist transparent, also gleichermaßen auf lokale, wie auch auf entfernte OSGI-Dienste zuzugreifen. Ein

[7] http://activemq.apache.org/
[8] http://openjms.sourceforge.net/
[9] http://www.oio.de/public/xml/rest-webservices.htm
[10] http://java.sun.com/j2se/1.4.2/docs/guide/rmi-iiop/index.html
[11] http://r-osgi.sourceforge.net/

Aufruf eines entfernten Dienstes in R-OSGI erfolgt vergleichbar mit RMI. Die feste Bindung von R-OSGI an OSGI impliziert die Einschränkung auf Java basierende Dienste.

R-OSGI wurde im Rahmen des Concierge OSGI-Rahmenwerkes umgesetzt und verfolgt daher ebenso das Ziel, in Umgebungen mit eingeschränktem Speicherplatz und verringerter Rechnerleistung eingesetzt werden zu können. R-OSGI ist ein auf OSGI-Dienste eingeschränkter, aber dafür sehr effizienter Ansatz (zum Teil schneller als RMI [RAR07b]) für Kommunikation zwischen Diensten.

XML-RPC

Diese Technologie sieht einen *Remote Procedure Call* (RPC) unter Nutzung von XML vor. Dieser Verfahren ist ein relativ alter, dafür sehr etablierter quasi Standard für den Aufruf entfernter Dienste. Dieses relativ einfache und relativ effiziente Protokoll ist für sehr viele Plattformen verfügbar. Das Problem bei XML-RPC liegt jedoch in der Darstellung komplexer Objekte durch XML. In dem XML-RPC Protokoll sind nur eine sehr kleine Anzahl von Basis-Datentypen für die Parameter vorgesehen. Abhilfe bringt an dieser Stelle eine (eigene, proprietäre) Erweiterung um Mechanismen zur Serialisierung von komplexen Objekten.

Beispielsweise können komplexe Objekte per JAXB oder XStream in XML-Objekte umgewandelt werden, welche dann als String gekapselt in dem XML-RPC eingebettet werden können.

Jabber-RPC

XMPP kann über sogenannte *Extensions* erweitert werden. Eine solche Erweiterung stellt der *Jabber-RPC*[12] dar. Jabber-RPC nutzt das gleiche Prinzip und die gleiche Syntax wie XML-RPC, verwendet jedoch XMPP zum Datentransport.

SOAP RPC

SOAP RPC ist vergleichbar mit XML-RPC, stellt jedoch eher dessen Nachfolger beziehungsweise dessen Erweiterung dar. Im Gegensatz zu XML-RPC stehen bei SOAP umfangreichere Methoden zur Anfrage von Diensten und Repräsentation von komplexen Objekten zur Verfügung. Diese zusätzlichen Funktionen machen eine Umsetzung von SOAP etwas aufwendiger als die von XML-RPC. So ist der Umfang der verfügbaren Implementierungen für unterschiedliche Systeme etwas geringer. Es existieren eine Reihe von Systemen zur vereinfachten Integration und Nutzung von SOAP. Systeme wie beispielsweise Apache AXIS bieten eine einfache Nutzung von SOAP, sowie Werkzeuge zur Erstellung der WSDL Beschreibung, beziehungsweise zur Generierung von Stubs anhand der WSDL-Beschreibung.

JAX-WS

Die von Sun entworfene JAX-RPC Spezifikation sieht einen Aufruf entfernter Dienste vor, welcher vom Ansatz her vergleichbar mit XML-RPC ist. Bei JAX-RPC war die Nutzung von WSDL vorgesehen. Ursprünglich wurden von JAX-RPC auch nur primitive Datentypen unterstützt, was eigene Methoden zur Serialisierung komplexer Objekte notwendig machte. JAX-RPC wurde inzwischen durch die Spezifikation JAX-WS weitestgehend ersetzt, welche eine Verwendung von SOAP oder JAXB, sowie WSDL vorsieht.

SNMP

Das *Simple Network Management Protocol* (SNMP) wurde dahingehend konzipiert, Gerätschaften wie beispielsweise Router, Server oder Drucker in einem Netzwerk zu überwachen und zu steuern. Es basiert auf binär kodierten (mittels ASN.1, BER) Nachrichten, welche direkt per UDP über Sockets versendet werden. Anhand des *Object-Identifiers* (OID) kann der Typ des Endgerätes beziehungsweise seiner Schnittstelle bestimmt werden. Über den OID wird hierzu auf globale Liste von Beschreibungselementen (*Management Information Base* - MIB) referenziert, welche die Beschreibung für das jeweilige Endgerät in einem auf ASN.1 basierenden Format enthält. Die Kommunikation mittels SNMP ist durch die Nutzung von (durch BER kodierten) binären Nachrichten (je nach Anwendungsbereich) eine relativ effiziente [PDMQ04, LAV+05] Alternative.

Im Rahmen von SNMP können eigene Anwendungen erstellt, sowie eigene MIB Beschreibungen integriert werden. Es existieren in diesem Zusammenhang Ansätze, SNMP direkt als Plattform für RPC zu verwenden [DRT04]. Ein Problem besteht jedoch in dem Aufwand, ein MIB eines Dienstes zu erstellen und bereitzustellen. Durch den Einsatz zusätzlicher Rahmenwerke ist es möglich die Erstellung und Bereitstellung zu automatisieren, jedoch ist die Anwendung im Vergleich zu anderen Ansätzen relativ unflexibel.

[12] `http://xmpp.org/extensions/xep-0009.html`

Dienst Lokalisation

SLP

Das *Service Location Protocol* (SLP) [VGPK99] ist ein plattformunabhängiges Protokoll zur Auffindung von Diensten in IP basierten Netzen. Es basiert auf einem eigenen binären Nachrichtenformat und ermöglicht die Adressierung von Diensten über URIs.

Sogenannte *Directory Agents* (DA) können die Rolle eines Registrars beziehungsweise Brokers einnehmen und bieten die Möglichkeit Anfragen zu Cachen. SLP kann sowohl mit DAs, als auch ohne eingesetzt werden.

DAs oder andere Dienste (falls kein DA gefunden wurde) werden durch Multicast-Nachrichten ermittelt. Nachfolgende Nachrichten können dann per Unicast ausgetauscht werden.

Ein Dienst kann neben der URI auch über weitere Eigenschaften beschrieben werden. Bei der Suche nach passenden Diensten kann ein Filter im *Lightweight Directory Access Protocol* (LDAP[13]) Format angewendet werden, um über diese Eigenschaften zu suchen.

SSDP

Das *Simple Service Discovery Protocol* (SSDP[14]) wird von UPnP verwendet, um andere UPnP-Dienste in IP-Netzen aufzufinden. Die Nachrichten basieren auf HTTP-Anfragen und enthalten eine Reihe einfacher Angaben (Filter) über den gesuchten Dienst. Die Nachrichten werden per Multicast im Netzwerk verteilt, wobei auf eine zentrale Instanz bei diesem Protokoll verzichtet wird. Bei der Suche nach Diensten erhält jeder UPnP-Dienst die Nachricht und muss diese gegebenenfalls beantworten. Als Resultat erhält der anfragende Dienst unter anderem die URI gefundener Dienste.

PDP

Das *Peer Discovery Protocol* (PDP) [BLMM04] ist Teil des JXTA Peer-to-Peer Konzeptes. Es verfolgt durch seine Verbindung zu Peer-to-Peer einen anderen Ansatz als die bisher beschriebenen Systeme. Basierend auf der Annahme bereits mit dem Peer-to-Peer Netz verbunden zu sein, hat dieses Protokoll zum Ziel, andere Peers mit bestimmten Eigenschaften zu finden. In diesem Fall besteht die Suche darin, Peers zu finden, welche bestimmte Dienste ausführen. Das Protokoll beschreibt diese Suchanfrage durch eine Nachricht im XML Format und sendet sie an alle Peers, welche dem Anfragenden Peer bekannt sind (vergleichbar mit einem Broadcast).

Bluetooth SDP

Es existieren inzwischen Sensornetze, welche Bluetooth zur Kommunikation nutzen [Yüc05]. Der Bluetooth Protokoll Stack [Blu04] beinhaltet das *Service Discovery Protocoll* (SDP). Es ermöglicht das Auffinden von Diensten innerhalb der Reichweite der Bluetooth-Schnittstelle.

Ein Bluetooth-Dienst wird über einen eindeutigen Bezeichner (Universally Unique Identifier - UUID) adressiert. Hierzu registriert ein Dienst sich am lokalen SDP, welches die UUID, sowie weitere Attribute welche den Dienst beschreiben, in seine Service Discovery Database (SDDB) aufnimmt.

Anders als bei den meisten anderen Ansätzen fragt das anfragende System per Broadcast alle erreichbaren Systeme an, welche wiederum mit den Einträgen aus ihrer SDDB antworten. Die Kommunikation erfolgt direkt mit den Geräten und es ist kein Mechanismus wie Broker, Publish/Subscribe, Advertisements oder Notifikation vorgesehen.

Andere Ansätze

Es existieren noch eine Reihe weiterer Ansätze für bestimmte andere Anwendungsbereiche. Viele dieser Ansätze bestehen jedoch bisher nur in der Theorie oder haben sich bisher nicht durchgesetzt:

- **Salutation:** Ist ein Protokoll zur Auffindung von Diensten, welches vergleichbar mit den Technologien UPnP oder Jini. Das Projekt wurde in vielen Arbeiten referenziert, wird jedoch nicht weiter gepflegt.
- **INS/Twine:** ist ein Ansatz, welcher auf Peer-to-Peer Konzepten aufsetzt und daher ohne zentrale Instanz auskommt [BBK02a]. Interessant an diesem Ansatz ist die Fokussierung auf die Suche über die Attribute. Das Intentional-Naming-System (INS) [AWSBL99] definiert hierzu einen Standard, Geräte über ihre Attribute beziehungsweise Fähigkeiten zu adressieren.

[13] `http://tools.ietf.org/html/rfc4510`

[14] `ftp://ftp.pwg.org/pub/pwg/ipp/new_SSDP/draft-cai-ssdp-v1-03.txt`

A.3 Rahmenwerke zur Kommunikation

In Abschnitt 4.2.3 wurden verschiedene Rahmenwerke und Systeme zur Kommunikation zwischen verteilten Systemen vorgestellt. In diesem Abschnitt werden die betrachteten Technologien genauer erläutert.

CORBA

Die *Common Object Request Broker Architecture* ermöglicht (durch die Nutzung von IIOP) den Plattform übergreifenden Zugriff auf verteilt im Netzwerk verfügbare Objekte (siehe Abschnitt 3.3 objektorientierte Middleware). Das Konzept von CORBA verfolgte die Vision eines Internet weiten Objekt-Busses, also eine verstärkte Anbindung und Wiederverwendung bestehender Objekte über Netzwerke hinweg. Unter anderem verursacht durch die Komplexität von dem Konzept von CORBA (welche auch durch die Nutzung von IDL verursacht wurde) hat sich diese Idee nicht etablieren können.

Jini, JavaSpaces

Die *Java Intelligent Network Infrastructure* (JINI[15]) erweitert RMI um ein Rahmenwerk zur Registrierung und Suche von Diensten in IP basierten Netzen.

Ein *Lookup-Service* nimmt hierbei die Rolle eines zentralen Registrars ein, welcher in Jini benötigt wird. Das *Discovery&Join-Protokoll* von Jini bietet die Möglichkeit, den Lookup-Service per Multicast zu ermitteln (wenn die Adresse vorab nicht bekannt ist) und sich anschließend bei dem Lookup-Service anzumelden oder Anfragen zu stellen.

Eine Suchanfrage kann (analog zu einer Anfrage auf den RMI-Namensdienst) über die Service-ID erfolgen. JINI erweitert diese Möglichkeit um weitere Parameter wie eine Zuordnung zu Gruppen (durch den JoinManager) und Attribute, welche eingesetzt werden können, um einen Dienst zu beschreiben. Dies ermöglicht die Suche nach Diensten über deren Eigenschaften.

Das Konzept Java Spaces[16] setzt auf dem von Jini auf und verwendet Teile davon. Das Konzept sieht eine indirekte Kommunikation zwischen verteilten Diensten vor, indem alle Dienste auf eine Art gemeinsamen Datenraum zugreifen. Dienste können aus diesem Datenraum Objekte beziehen und nutzen, was sehr leichtgewichtige Dienste ermöglicht (benötigte Objekte werden zur Laufzeit aus dem Datenraum nachgeladen).

Ähnlich wie bei OSGI liegt ein Mehrwert von Jini und JavaSpaces unter anderem darin, Code verteilt zu adressieren und nutzbar zu machen. Leider scheint Jini jedoch aktuell nicht gepflegt oder weiter entwickelt zu werden.

AGENT, JADE

Die Grundidee von *Agenten* ist vergleichbar mit denen von Diensten. Die Kommunikation erfolgt im Allgemeinen über den Austausch von Nachrichten (also asynchrone Kommunikation). Die FIPA[17] hat mit der Agent Communication Language (ACL[18]) Spezifikationen für die Nachrichtenstruktur vorgegeben (nicht aber über deren Transport). Die eigentliche Nachricht wird in einer ACL Nachricht gekapselt, um Interoperabilität zwischen verschiedenen Systemen, die Verarbeitung der Nachricht, sowie deren Transport zu ermöglichen.

Agenten Systeme sind ursprünglich entworfen worden, um Dienste möglichst lose gekoppelt und ohne zentrale Instanzen zu betreiben, was gerade beim Einsatz in dynamischen Umgebungen von Vorteil ist. Im Fokus steht häufig auch das Prinzip der mobilen Agenten, welche das Nachladen von Code (ähnlich zu OSGI) und damit das dynamische Verteilen der Aufgaben im Netz ermöglichen soll. Diese lose gekoppelten Agenten sollen anhand ihrer Aufgaben dynamisch verbunden werden, um so ein verteiltes System zu bilden. Dieses verteilte System soll auch komplexe Abläufe und Funktionen durchführen können, indem es die einzelnen Aufgaben des Prozesses dynamisch an die zuständigen Agenten verteilt. Diese dynamische Koppelung von Agenten macht ein sogenanntes *matchmaking* notwendig. Hierzu enthält die ACL Struktur Referenzen auf Ontologie-Elemente.

Das *Java Agent DEvelopment Framework* (JADE[19]) ist eine häufig eingesetzte, FIPA-konforme Implementierung. In JADE sind (relativ komfortable) Funktionen zur Nutzung einer Ontologie enthalten. Diese Ontologie kann genutzt werden um die Funktion eines Agenten zu spezifizieren. Jeder Agent kann gegebenenfalls eigene Elemente zu der Ontologie hinzufügen (vergleichbar mit Abschnitt 4.5.4). Die Verteilung von Nachrichten erfolgt anhand dieser Spezifikation (vergleichbar zu *content based routing*), was zur Folge hat, dass ein Agent nur Nachrichten erhält, die

[15] `http://www.jini.org/`
[16] `http://java.sun.com/developer/technicalArticles/tools/JavaSpaces/`
[17] eine IEEE-Standardisierungskommission
[18] `http://www.fipa.org/repository/aclspecs.html`
[19] `http://jade.tilab.com/`

ihn betreffen und die er verarbeiten kann. Die in JADE genutzte Ontologie kann mit der *Web Ontology Language* (OWL) kombiniert werden [New06].

Jede JADE Umgebung (Agent Plattform - AP) besitzt einige Basis-Dienste. Hierzu zählt der *Agent Discovery Service* (ADS). In JADE ist ein Multicast-Ansatz integriert, es können jedoch auch andere Mechanismen zur Dienst Lokalisation genutzt werden (siehe Abschnitt A.2). Zu den Basis Diensten zählt auch der *Directory Facilitator* (DF). Dieser Dienst ist vergleichbar mit einem Namensdienst, wird jedoch nur bei Bedarf gestartet (wenn kein anderer DF im Netz gefunden wurde). Durch das Verbinden von vielen AP können Peer-to-Peer-ähnliche Strukturen geschaffen werden, wodurch dieser Ansatz vergleichbar mit dem von JXTA [LKM06] ist.

JADE nutzt lokale Methodenaufrufe und RMI als Basis für die Kommunikation zwischen Agenten. Zusätzlich wurden Transportprotokolle wie HTTP/IIOP integriert. Über eine offene Schnittstelle können weitere Transportprotokolle genutzt werden. So ist beispielsweise auch eine Integration von JXTA möglich, womit auch NAT- und Firewall-Problematiken umgangen werden können [LKM06].

JADE liefert ein komplettes Rahmenwerk mit Lifecylemanagement, sowie Tools und GUIs zum Überwachen der Agenten und der Kommunikation. In JADE wird Sicherheit durch den Einsatz von Policy-gesteuerten AAA-Mechanismen (*Authentication*, *Authorization*, *Accounting*) gewährleistet. Es wurde zudem für Einsatz auf ressourcenbeschränkten, mobilen Endgeräten konzipiert. Die Implementierung von JADE zeigt gute Leistungen [CQV01]. Agenten, vor allem mobile Agenten, finden bisher meist Anwendung im wissenschaftlichen Umfeld, wie beispielsweise in Sensornetzen [KBKA05]. Teile der Funktionen von JADE können sinnvoll für diese Arbeit genutzt werden. Bei anderen Funktionen wie beispielsweise die Ontologie, der Nachrichten-basierten Kommunikation und dem Content-Based-Routing ist fraglich, ob diese sich für Anforderungen der Suche als geeignet erweisen.

Agentenbasierte Systeme können auch Nachteile mit sich bringen, wie beispielsweise durch redundante Datenhaltung oder durch schwierige, wenn nicht sogar unmögliche Evaluation.

UPnP

Universal Plug and Play (UPnP) verbindet mehrere Technologien miteinander und dient der Steuerung von Geräten in einem Netzwerk. UPnP wird häufig lediglich zum Auffinden von Diensten genutzt, bietet aber einen sehr generischen Ansatz mit Funktionalitäten, welche darüber hinaus auf die darauf folgende Nutzung des gefundenen Dienstes eingehen. UPnP verfolgt einen mit Jini vergleichbaren Ansatz [New05]. Zur Lokalisation von Diensten wird das SSDP (*Simple Service Discovery Protocol*) verwendet. Geräte (oder Dienste) im Netz beschreiben sich selbst mittels einer XML-Beschreibung, welche per HTTP abgerufen werden kann. Die Beschreibung enthält neben einigen Basisinformationen auch URLs, über die weitere Beschreibungen über den Dienst bezogen werden können. Hierzu zählt vor allem auch die Beschreibung der Schnittstelle des Dienstes. Die Datentypen, welche in den Parametern verwendet werden, können neben standard Datentypen (aus der XML-Schema Definition - XSD) auch herstellerspezifische Datentypen beinhalten (siehe Abschnitt 2.5.1 *Defining and processing extended data types* der UPnP Spezifikation v.1.1), indem auf eigene Namensräume und XML-Schemata verwiesen wird. Die Spezifikation von UPnP sieht vor, die *Interfaces* anschließend anhand der Beschreibung über SOAP abzuwickeln. UPnP wurde ursprünglich von Microsoft entwickelt, ist inzwischen jedoch auch als Open Source-Implementierung erhältlich und wird von einer Reihe von Endgeräten bereits unterstützt.

WCF

Das *Windows Communication Framework* (WCF) ist Teil von .NET und besteht aus einer Zusammenfassung verschiedener Technologien. Einigen Komponenten, wie beispielsweise das *Distributed Component Model* (DCOM), welche früher zur Kommunikation zwischen Windows-Diensten entworfen wurden, wurden zur Abwärtskompatibilität beibehalten. DCOM war ein Ansatz, welcher mit CORBA vergleichbar war und mit diesem konkurrierte. Beide nutzten zur Beschreibung ihrer Schnittstellen IDL.

Das WCF in der Version, wie sie aktuell in .NET 3.5 enthalten ist, beinhaltet unter anderem *.NET Remoting*, welches Verfahren für Transaktionen, Sicherheit und Message Queues zwischen Diensten bereitstellt. Durch die Nutzung von .NET Remoting können je nach Anwendungsfall unterschiedliche Technologien für den Anbindung eines Dienstes genutzt werden. So kann für lokale Aufrufe von Diensten eine direkte Kommunikation zwischen den Diensten erfolgen (über den *Remoting-Channel IPC*[20]). Für eine plattformunabhängige Anbindung eines Dienstes wird eine Anbindung über SOAP (wahlweise mit MTOM), JSON, REST oder anderer Verfahren ermöglicht[21]. Um externe Dienste geeignet zu adressieren, werden sogenannte (Service-, Data-) *Contracts* genutzt. Diese sind vergleichbar mit WSDL und können auch dorthin exportiert werden.

[20] http://www.dotnetframework.de/lserver/artikeldetails.aspx?b=3199

[21] http://msdn.microsoft.com/en-us/library/cc512374.aspx

WCF bietet einen sehr großen Funktionsumfang, ist in der Regel jedoch nur auf Windows basierten Systemen einsetzbar (existierende Ansätze zur Integration auf anderen Plattformen wie Mono[22], DotGNU[23] und Rotor[24] unterstützen bisher nur ältere Versionen von .NET beziehungsweise nur einen eingeschränkten Funktionsumfang des WCF) und sind auch im Vergleich mit vielen andere Ansätzen relativ schwergewichtig.

JavaEE-EJB

Die *Java Platform Enterprise Edition* (JavaEE) enthält die Spezifikation für die *Enterprise Java Beans*(EJB). EJB sind vergleichbar mit Diensten.

EJB und .NET Enterprise Services fokussieren als Grundlage für dienstorientierte Architekturen im Umfeld von Unternehmensanwendungen einen ähnlichen Anwendungsbereich. EJB erhalten vom JavaEE-Rahmenwerk ein standardisiertes Umfeld zur Ausführung bereitgestellt. Die Funktionen die dabei von der JavaEE vorgesehen sind, sind sehr umfangreich und beinhalten einige bereits erwähnte Komponenten. Unter anderem ist es möglich, Dienste zur Darstellung interaktiver Webseiten (Java Server Pages - JSP) zu betreiben.

Eine der dabei angebotenen Funktionalitäten ist die Java EE Connector Architecture (JCA). Das Problem bei der Kommunikation zwischen Diensten besteht häufig in der Verbindung von Diensten unterschiedlicher Hersteller. Ein JCA Konnektor ermöglicht die Kommunikation zwischen solchen heterogenen Systemen auf einheitlicher Ebene. Mit der Spezifikation *Java API for XML - Web Services* (JAX-WS) wurde in JavaEE eine API zur Erstellung von Web-Services eingeführt.

Devices Profile for Web Services

Ein noch recht neuer Ansatz, um Netzdienste auch auf mobilen Endgeräten zu ermöglichen, ist der OASIS Standard Devices Profile for Web Services (DPWS) Version 1.1[25]. Der Standard setzt auf den Technologien Simple Object Access Protocol (SOAP) und WSDL auf und wurde speziell für ressourcenbeschränkte Endgeräte entworfen. Er beinhaltet zudem Mechanismen zur Auffindung von Diensten über Multicast-Nachrichten und Funktionen zur Behandlung von Ereignissen. Aktuell stehen Implementierungen des DPWS-Stacks für C und Java zur Verfügung. Die Arbeit [Hil09] gibt einen guten Überblick über DPWS und dessen Funktionen, sowie eine Aufwandsbetrachtung in ressourcenbeschränkten Szenarien.

MundoCore

MundoCore ist eine Middleware, welche an der TU-Darmstadt entwickelt wird [Ait06]. Sie ist Gegenstand aktueller Forschungsaktivitäten und stellt keinen Standard dar. MundoCore wurde für den Anwendungsbereich Ubiquitous Computing konzipiert. Es wurde ursprünglich in Java implementiert, ist aber auch in C++ oder Python verfügbar.

Die Kommunikation basiert auf logischen Kanälen, auf welchen Nachrichten mit Hilfe von Publish-Subscribe-Mechanismen gesendet bzw. empfangen werden können. Dienste im Netz werden über den Kanal adressiert, auf denen diese sich registriert haben. Es ist möglich, unterschiedliche Kommunikationsparadigmen auf diese kanalbasierte Kommunikation abzubilden. Je nach Bedarf können Kanäle direkt für die dedizierte Kommunikation zwischen zwei Diensten eingesetzt werden (unicast), oder es können mehrere oder alle Dienste über einen Kanal kommunizieren (multi- oder anycast). Hierzu werden in der Regel alle Nachrichten über Broadcast verschickt und jeder Dienst reagiert nur auf die Nachrichten, welche den Kanälen angehören, welche dieser abonniert hat.

Die Architektur von MundoCore erlaubt die Verwendung unterschiedlicher Technologien für Transport und Repräsentation der Daten, sowie für die Lokalisation von Diensten im Netzwerk. So kann der Transport von Daten via TCP/IP, gemeinsam genutzten Speicher oder Bluetooth-Protokollen erfolgen. Vergleichbar mit dem Konzept von OSGI können Dienste zur Laufzeit nachgeladen und ausgeführt werden. Die Repräsentation der Daten kann binär oder über XML/SOAP erfolgen.

MundoCore wurde speziell für den Einsatz in ressourcenbeschränkten Systemen konzipiert. Die MundoCore Bibliothek benötigt rund 90kByte und kann auch unter Java Micro Edition (Java ME) verwendet werden.

Zur Verwendung von MundoCore wird ein Preprozessor benötigt, welcher spezielle Tags im Code zur Generierung von Stubs nutzt. Dies erzeugt jedoch bei der Entwicklung einen Zusatzaufwand.

[22] http://www.mono-project.com/WCF
[23] http://www.dotgnu.org/pnet.html
[24] http://msdn.microsoft.com/de-de/library/cc405435.aspx
[25] http://docs.oasis-open.org/ws-dd/dpws/1.1/cs-01/wsdd-dpws-1.1-spec-cs-01.html

JXTA

Im Gegensatz zu den anderen Ansätzen basiert JXTA auf dem Peer-to-Peer Konzept, benutzt also die Annahme eine Verbindung zu einem logischen Verbund von Systemen zu besitzen als Grundlage. Das JXTA Protokoll ist offen und ermöglicht somit die Umsetzung eigener Dienste, sowie deren Kommunikation über JXTA [SE05]. Die Lokalisation von Diensten erfolgt über das *Peer Discovery Protocol* (PDP). Das auf XML basierende Nachrichtenformat und eine Reihe von Implementierungen für unterschiedliche (auch mobile) Systeme erlauben eine plattformübergreifende Kommunikation mittels JXTA.

A.4 Technologien zur semantischen Beschreibung von Netzdiensten

In dieser Arbeit werden Technologien zur semantischen Beschreibung aus dem Bereich der Netzdienste zur Beschreibung von Sensoren herangezogen. Dieser Abschnitt erläutert genauer die verschiedenen betrachteten Technologien.

SAWSDL

Die Web Services Description Language[26] ist ein etablierter Standard aus dem Bereich Schnittstellen-Beschreibung für Netzdienste. WSDL basiert auf XML in Kombination mit dem XML Schema (XSD) und bietet die Möglichkeit, Schnittstellen unabhängig von dem verwendeten System oder Netzwerk zu beschreiben. WSDL für sich genommen dient jedoch nur zur rein syntaktischen Beschreibung von Schnittstellen.

In diesem Zusammenhang wurde WSDL sukzessiv dahingehend erweitert, auch semantische Beschreibungselemente zu unterstützen. WSDL-Semantics[27] und darauf aufbauend SAWSDL[28] sind solche Erweiterungen von WSDL. SAWSDL hat den Vorteil, dass es durch seine Abwärtskompatibilität innerhalb bestehender WSDL-basierender Systeme genutzt werden kann. Das Ziel dieses Ansatz ist eine einfache, generische und dadurch leichtgewichtige (im Vergleich zu OWL-S) Form der semantischen Beschreibung. Der Fokus liegt verstärkt in der Beschreibung der Methoden und der dabei benötigten Eingabe- und Ausgabeparameter, was bei einfachen Formen der Dienstfindung hilfreich ist. Für eine umfassende Form der automatisierten Komposition von Diensten ist der Umfang der durch SAWSDL angebotenen semantischen Beschreibungselemente jedoch oft nicht ausreichend [MPW07].

SAWSDL spezifiziert keine Form oder Sprache zur Darstellung von semantischen Modellen wie beispielsweise Ontologien. Stattdessen bietet es Mechanismen die es ermöglichen aus WSDL Beschreibungen heraus auf semantische Modelle zu referenzieren.

WSMO, WSML

Die WSMO stellt ein Meta-Modell für WSML dar. WSML definiert eine formale Sprache zur semantischen Beschreibung von Web Services. Zudem beinhaltet sie eine Sprache zur Auswertung mittels logischer Ausdrücke und Regeln. WSMO und WSML benutzen eine eigene Syntax, beziehungsweise eine eigene Form von Auszeichnungssprache, welche ohne XML auskommt. Es kann aber bei Bedarf auf XML zurückgreifen, um Beschreibungen zu exportieren oder zu importieren. Eine WSMO-Beschreibung eines Dienstes beinhaltet folgende Beschreibungselemente:

- Mittels der Aussagen im Bereich *Capabilities* kann über Vorbedingungen und Annahmen festgelegt werden, ob ein Dienst genutzt werden kann.
- Die *Choreography* beschreibt die Funktionsweise eines Dienstes aus Sicht des anfragenden Systems. In diesem Bereich können gerade auch komplexere Abläufe der Interaktion beschrieben werden.
- Im Bereich *Orchestration* wird aus der Sicht des Dienst Anbieters beschrieben, wie der Dienst mit anderen Diensten kombiniert wird, um seine Funktionalitäten anzubieten.

Mediatoren bilden das verbindende Element zwischen zwei Komponenten (beziehungsweise Diensten oder extern verfügbaren Ontologien). Um eine mögliche Heterogenität der Schnittstellen zu überbrücken, spielen Mediatoren eine vermittelnde Rolle. WSML besitzt ein umfangreiches, solides Grundmodell und ermöglicht einen universellen Einsatz, ohne Einschränkung der Anwendungsgebiete.

[26] `http://www.w3.org/TR/wsdl`
[27] `http://www.w3.org/Submission/WSDL-S/`
[28] Semantic Annotations for WSDL Working Group: `http://www.w3.org/2002/ws/sawsdl/`

RDF(S), OWL-S, OWL

Als Grundlage für die Beschreibung von Objekt-Beziehungen wird häufig das Resource Description Framework (RDF) in Kombination mit XML als *Ausdrucksschema* angewandt. RDF spezifiziert nur die Syntax von Aussagen. Eine Aussage besteht aus drei Komponenten:

$$Subjekt \longrightarrow Prädikat \longrightarrow Objekt$$

Dieses RDF-Triplet bildet die Grundlage und Kernkomponente für Wissensrepräsentation, wobei Subjekt und Objekt Elemente darstellen und das Prädikat die Beziehung vom Subjekt zum Objekt bezeichnet. Die Elemente werden über *Uniform Resource Identifier*[29] eindeutig bezeichnet.

RDF für sich genommen spezifiziert nur diese Syntax. Daher existieren keine Vorgaben, was die Prädikate betrifft. Es besteht somit die Möglichkeit eigene Prädikate zu definieren oder öffentlich verfügbare Definitionen zu nutzen.

Das RDF-Schema (RDFS) ist eine solche öffentlich verfügbare Definition eines gemeinsamen Vokabulars für die Elemente und Prädikate[30]. RDFS bildet wiederum die Grundlage für OWL, welche das Vokabular von RDFS erweitert und zusätzliche Sprachkonstrukte definiert. Diese Konstrukte dienen einer gezielteren Beschreibung von Elementen, wie beispielsweise der Einschränkung auf bestimmte Datentypen.

OWL-S erweitert wiederum OWL um Elemente zur Beschreibung von Netzdiensten. Hierzu ist durch OWL-S eine Ontologie definiert worden, welche eine Reihe von Beschreibungselementen vorsieht, welche es möglich machen, geeignete Dienste zu bestimmen und entsprechend automatisch in den Verarbeitungsprozess zu integrieren. Wie in Abbildung A.1 dargestellt ist, teilt sich die Service-Ontologie wiederum in drei untergeordnete Ontologien auf:

- Das *Profile* beschreibt die Dienstleistung, die von dem Dienst erbracht wird.
- Über das *Grounding* wird beschrieben, wie die Schnittstellen des Dienstes aufzurufen sind.
- Im *Model* wird die Interaktion mit dem Dienst und der Ablauf der Aufrufe beschrieben.

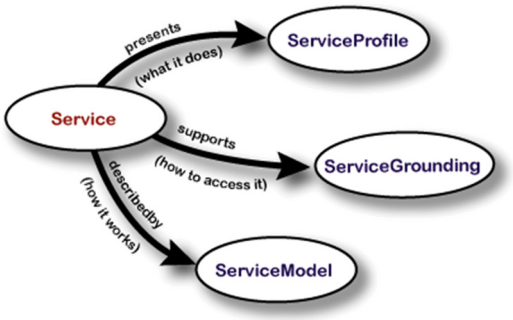


Abbildung A.1: Beschreibungselemente von OWL-S

A.5 OWL-S Beschreibung

Im folgenden Teil wird die OWL-S Beschreibung eines Sensors beispielhaft aufgezeigt. In diesem Fall handelt es sich um den Terminkalender-Sensor, welcher die Nutzung der Informationen über die Termine aus Outlook heraus möglich macht.

OWL-S Service

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!--HEADER-PART BEGIN-->
<!DOCTYPE uridef[
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
<!ENTITY owl "http://www.w3.org/2002/07/owl">
<!ENTITY xsd  "http://www.w3.org/2001/XMLSchema">
<!ENTITY actor "http://www.daml.org/services/owl-s/1.1/ActorDefault.owl">
```

[29] RFC3986 - Uniform Resource Identifier (URI): Generic Syntax

[30] http://www.w3.org/TR/rdf-schema

```
<!ENTITY service "http://www.daml.org/services/owl-s/1.1/Service.owl">
<!ENTITY process "http://www.daml.org/services/owl-s/1.1/Process.owl">
<!ENTITY profile "http://www.daml.org/services/owl-s/1.1/Profile.owl">
<!ENTITY mcGrounding "urn:kom.sn.common.v1:KOMSSN-MCGrounding.owl">

<!ENTITY my_service "urn:sn.sensors.outlooksensor.v1:OWLS-Service">
<!ENTITY my_profile "urn:sn.sensors.outlooksensor.v1:OWLS-Profile">
<!ENTITY my_process "urn:sn.sensors.outlooksensor.v1:OWLS-Process">
<!ENTITY my_grounding "urn:sn.sensors.outlooksensor.v1:OWLS-Grounding">
<!ENTITY komssn "http://kom.sn.common.v1/KOMSSN-NetworkOntology.owl">
<!ENTITY komssnowls "urn:kom.sn.common.v1:KOMSSN-OWLS.owl">
]>
<rdf:RDF
xmlns:rdf=     "&rdf;#"
xmlns:rdfs=    "&rdfs;#"
xmlns:owl=    "&owl;#"
xmlns:xsd=     "&xsd;"
xmlns:service= "&service;#"
xmlns:process= "&process;#"
xmlns:profile= "&profile;#"
xmlns:actor= "&actor;#"
xmlns:mcGrounding= "&mcGrounding;#"
xmlns:komssn= "&komssn;#"
xmlns:my_service= "&my_service;#"
xmlns:my_process= "&my_process;#"
xmlns:komssnowls= "&komssnowls;#"
xml:base= "&my_service;"
>
<owl:Ontology>
       <owl:versionInfo>$Id: OWLSGroundingEmitter.java,v 1.1 naveen </owl:versionInfo>
       <rdfs:comment> ---Add INFO--- </rdfs:comment>
       <owl:imports rdf:resource="&rdf;" />
       <owl:imports rdf:resource="&rdfs;" />
       <owl:imports rdf:resource="&owl;" />
       <owl:imports rdf:resource="&service;" />
       <owl:imports rdf:resource="&profile;" />
       <owl:imports rdf:resource="&process;" />
       <owl:imports rdf:resource="&my_process;" />
       <owl:imports rdf:resource="&my_profile;" />
       <owl:imports rdf:resource="&actor;" />
       <owl:imports rdf:resource="&mcGrounding;" />
<owl:imports rdf:resource="&komssn;" />
<owl:imports rdf:resource="&komssnowls;" />
</owl:Ontology>
<!--HEADER-PART END-->

<service:Service rdf:ID="OutlookSensor_Service">
<service:describedBy rdf:resource="&my_process;#OutlookSensor_Process"/>
    <service:presents rdf:resource="&my_profile;#OutlookSensor_Profile"/>
<service:supports rdf:resource="&my_grounding;#OutlookSensor_Grounding"/>
</service:Service>
</rdf:RDF>
```

Der OWL-S Service Teil der Beschreibung dient lediglich des Verweises auf die anderen drei Teile: Dem Process, dem Profile und dem Grounding.

Zur Verkürzung der Darstellung wird in den folgenden Beschreibungen der als *HEADER-PART* gekennzeichnete Teil ausgelassen, da er innerhalb des Systems gleichbleibend ist.

OWL-S Process

Der *Process* Teil der Beschreibung ermöglicht die Beschreibung der Funktionsweise des Dienstes. In diesem Fall handelt es sich um einfache, atomare Aufrufe (engl. *atomic process*).

```
<!--Definitions for Atomic Process : OutlookSensor_getNextAppointment-->
<!--Output-->
<process:Output rdf:ID="OutlookSensor_getNextAppointment_OUT0">
    <process:parameterType rdf:datatype="xsd;#anyURI">
        #nextAppointment
    </process:parameterType>
    <komssnowls:KOMSSN_Datatype rdf:datatype="&xsd;#anyURI">&komssn;#UserData@Appointment</komssnowls:KOMSSN_Datatype>
</process:Output>
<!--Process-->
<process:AtomicProcess rdf:ID="OutlookSensor_getNextAppointment">
<process:hasOutput rdf:resource="#OutlookSensor_getNextAppointment_OUT0"/>
<komssnowls:expirationTime rdf:datatype="&xsd;#long">10000</komssnowls:expirationTime>
</process:AtomicProcess>
<!--Definitions for Atomic Process : OutlookSensor_hasAppointmentNowWithSubject-->
```

```
<!--Output-->
<process:Output rdf:ID="OutlookSensor_hasAppointmentNowWithSubject_OUT0">
    <process:parameterType rdf:datatype="xsd;#anyURI">
        #appointmentSubjects
    </process:parameterType>
    <komssnowls:KOMSSN_Datatype rdf:datatype="&xsd;#anyURI">&komssn;#UserData@Appointment</komssnowls:KOMSSN_Datatype>
</process:Output>
<!--Process-->
<process:AtomicProcess rdf:ID="OutlookSensor_hasAppointmentNowWithSubject">
<process:hasOutput rdf:resource="#OutlookSensor_hasAppointmentNowWithSubject_OUT0"/>
<komssnowls:expirationTime rdf:datatype="&xsd;#long">10000</komssnowls:expirationTime>
</process:AtomicProcess>
<!--Definitions for Atomic Process : OutlookSensor_hasAppointmentNow-->
<!--Output-->
<process:Output rdf:ID="OutlookSensor_hasAppointmentNow_OUT0">
    <process:parameterType rdf:datatype="xsd;#anyURI">
        #appointment
    </process:parameterType>
    <komssnowls:KOMSSN_Datatype rdf:datatype="&xsd;#anyURI">&komssn;#UserData@Appointment</komssnowls:KOMSSN_Datatype>
</process:Output>
<!--Process-->
<process:AtomicProcess rdf:ID="OutlookSensor_hasAppointmentNow">
<process:hasOutput rdf:resource="#OutlookSensor_hasAppointmentNow_OUT0"/>
<komssnowls:expirationTime rdf:datatype="&xsd;#long">10000</komssnowls:expirationTime>
</process:AtomicProcess>
</rdf:RDF>
```

OWL-S Profile

Der Profile Abschnitt ermöglicht eine Einordnung des Dienstes. Hierzu wird die Kategorie des Dienstes (*service category*), sowie die semantischen Beschreibung der Ein- und Ausgabeparameter der Methoden definiert.

```
<profile:Profile  rdf:ID="OutlookSensor_Profile">
<service:presentedBy rdf:resource="&my_service;#OutlookSensor_Service"/>
<profile:serviceName>OutlookSensor</profile:serviceName>
<profile:textDescription>ServiceInterface of OutlookSensor </profile:textDescription>
<profile:serviceCategory>
<profile:ServiceCategory rdf:ID="OutlookSensor_Category">
<profile:categoryName rdf:datatype="&xsd;#string">KOMSSN_Supplier_Category</profile:categoryName>
<profile:value rdf:datatype="&xsd;#string">AbstractSensor</profile:value>
</profile:ServiceCategory>
</profile:serviceCategory>
<profile:hasOutput rdf:resource="&my_process;#OutlookSensor_getNextAppointment_OUT0"/>
<profile:hasOutput rdf:resource="&my_process;#OutlookSensor_hasAppointmentNowWithSubject_OUT0"/>
<profile:hasOutput rdf:resource="&my_process;#OutlookSensor_hasAppointmentNow_OUT0"/>
</profile:Profile>
```

OWL-S Grounding

Der Grounding Teil der Beschreibung liefert notwendige Informationen zum Aufruf der Schnittstelle, wie beispielsweise der konkreten (syntaktischen) Datentypen.

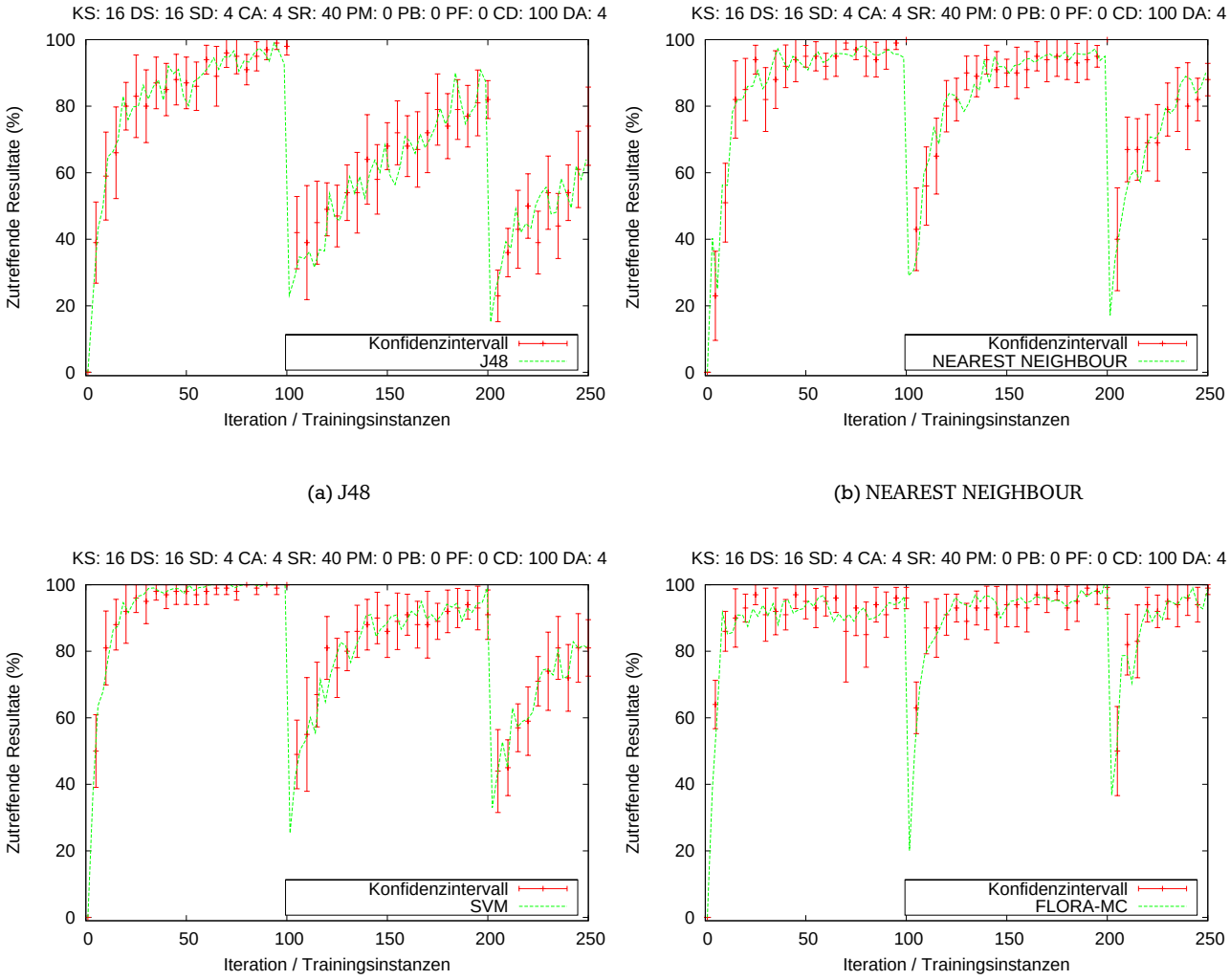
```
<mcGrounding:MundoCoreGrounding rdf:ID="OutlookSensor_Grounding">
       <service:supportedBy rdf:resource="&my_service;#OutlookSensor_Service" />
       <mcGrounding:hasAtomicProcessGrounding rdf:resource="#OutlookSensor_getNextAppointment_Grounding"/>
       <mcGrounding:hasAtomicProcessGrounding rdf:resource="#OutlookSensor_hasAppointmentNowWithSubject_Grounding"/>
       <mcGrounding:hasAtomicProcessGrounding rdf:resource="#OutlookSensor_hasAppointmentNow_Grounding"/>
       <mcGrounding:javaClassName rdf:datatype="&xsd;#string">OutlookSensor</mcGrounding:javaClassName>
</mcGrounding:MundoCoreGrounding>
<mcGrounding:MCAtomicProcessGrounding rdf:ID="OutlookSensor_getNextAppointment_Grounding">
       <mcGrounding:owlsProcess rdf:resource="&my_process;#OutlookSensor_getNextAppointment"/>
       <mcGrounding:javaMethodName rdf:datatype="&xsd;#string">getNextAppointment</mcGrounding:javaMethodName>
       <mcGrounding:processOutput>
               <mcGrounding:OutputParameter>
                       <mcGrounding:owlsOutputParameter rdf:resource="&my_process;#OutlookSensor_getNextAppointment_OUT0"/>
                       <mcGrounding:javaReturnType rdf:datatype="&xsd;#string">java.lang.Object</mcGrounding:javaReturnType>
               </mcGrounding:OutputParameter>
       </mcGrounding:processOutput>

</mcGrounding:MCAtomicProcessGrounding>
<mcGrounding:MCAtomicProcessGrounding rdf:ID="OutlookSensor_hasAppointmentNowWithSubject_Grounding">
       <mcGrounding:owlsProcess rdf:resource="&my_process;#OutlookSensor_hasAppointmentNowWithSubject"/>
       <mcGrounding:javaMethodName rdf:datatype="&xsd;#string">hasAppointmentNowWithSubject</mcGrounding:javaMethodName>

       <mcGrounding:processOutput>
               <mcGrounding:OutputParameter>
                       <mcGrounding:owlsOutputParameter rdf:resource="&my_process;#OutlookSensor_hasAppointmentNowWithSubject_OUT0"/>
                       <mcGrounding:javaReturnType rdf:datatype="&xsd;#string">sn.common.SearchObject</mcGrounding:javaReturnType>
```

```
            </mcGrounding:OutputParameter>
        </mcGrounding:processOutput>

</mcGrounding:MCAtomicProcessGrounding>
<mcGrounding:MCAtomicProcessGrounding rdf:ID="OutlookSensor_hasAppointmentNow_Grounding">
        <mcGrounding:owlsProcess rdf:resource="&my_process;#OutlookSensor_hasAppointmentNow"/>
        <mcGrounding:javaMethodName rdf:datatype="&xsd;#string">hasAppointmentNow</mcGrounding:javaMethodName>

        <mcGrounding:processOutput>
                <mcGrounding:OutputParameter>
                        <mcGrounding:owlsOutputParameter rdf:resource="&my_process;#OutlookSensor_hasAppointmentNow_OUT0"/>
                        <mcGrounding:javaReturnType rdf:datatype="&xsd;#string">sn.common.SearchObject</mcGrounding:javaReturnType>
                </mcGrounding:OutputParameter>
        </mcGrounding:processOutput>

</mcGrounding:MCAtomicProcessGrounding>
```

A.6 Weiterführende Ergebnisse der Analysen der Lernverfahren

In diesem Abschnitt werden weitere Analysen der Verfahren aus den Abschnitten 5.4, 5.5 und 5.6 aufgezeigt.

Exemplarisches Szenario mit Konfidenzintervallen



(a) J48

(b) NEAREST NEIGHBOUR

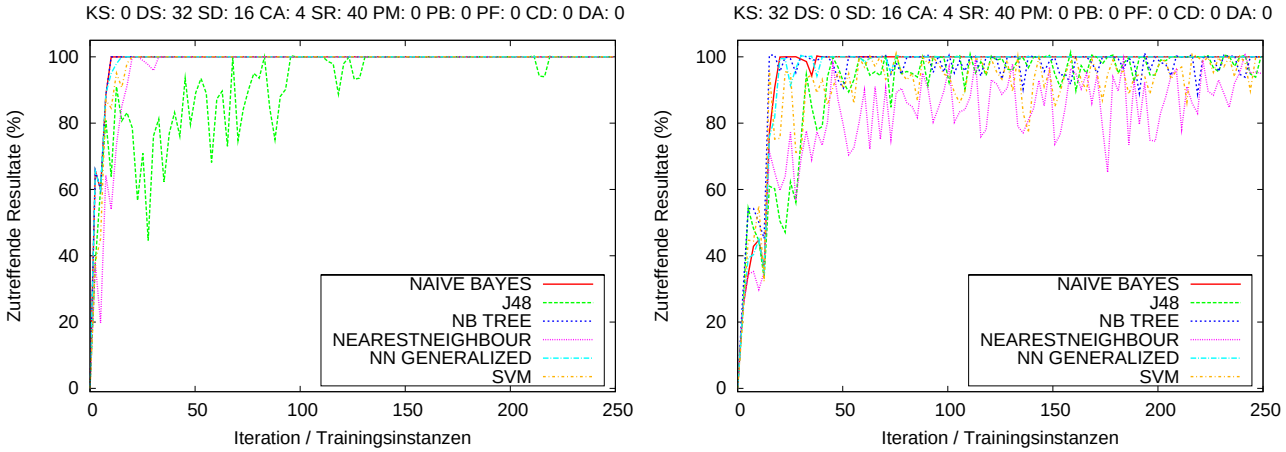
(c) SVM

(d) FLORA-MC

Abbildung A.2: Qualitätsanalyse mit Konfidenzintervallen

In diesem Abschnitt findet sich die Qualitätsanalyse eines exemplarischen Szenarios mit zusätzlicher Darstellung der Konfidenzintervalle, auf welche bisher zur Verbesserung der Lesbarkeit verzichtet wurde. Die Abbildungen A.2 zeigen die Konfidenzintervalle (für 95% Wahrscheinlichkeit und 10 Stichproben) einiger Verfahren aus Abbildung 5.40. Die anderen betrachteten Szenarien weisen vergleichbare Konfidenzintervalle auf.

Verhältnis diskreter und kontinuierlicher Sensoren



(a) Diskrete Sensoren

(b) Kontinuierliche Sensoren

Abbildung A.3: Vergleich verschiedener Arten von Sensoren

In der Abbildung A.3 werden die Arten von Sensoren variiert. Bei den bisher betrachteten Szenarien waren die Werte der 32 Sensoren gleichmäßig auf diskrete und kontinuierliche Datentypen verteilt. In der Abbildung A.3 werden jeweils einmal ausschließlich diskrete und einmal kontinuierliche Datentypen herangezogen. Während alle Verfahren mit Ausnahme von J48 auf diskreten Datentypen sehr gute Ergebnisse liefern, ist bei kontinuierlichen Daten erkennbar, dass gerade NEARESTNEIGHBOUR und SVM vergleichsweise schlechtere Ergebnisse erzielen.

Domäne diskreter Werte

Die Auswirkung der Größe der Domäne von diskreten Werten auf den Zeitbedarf zur Generierung von Modellen ist in Abbildung A.4 dargestellt.

Während einzelne Verfahren wie NEAREST NEIGHBOUR und J48 keine nennenswerte Veränderung aufweisen, steigt der Aufwand bei den Verfahren NB-TREE, SVM und NN GENERALIZED etwa linear mit der Größe der Domäne.

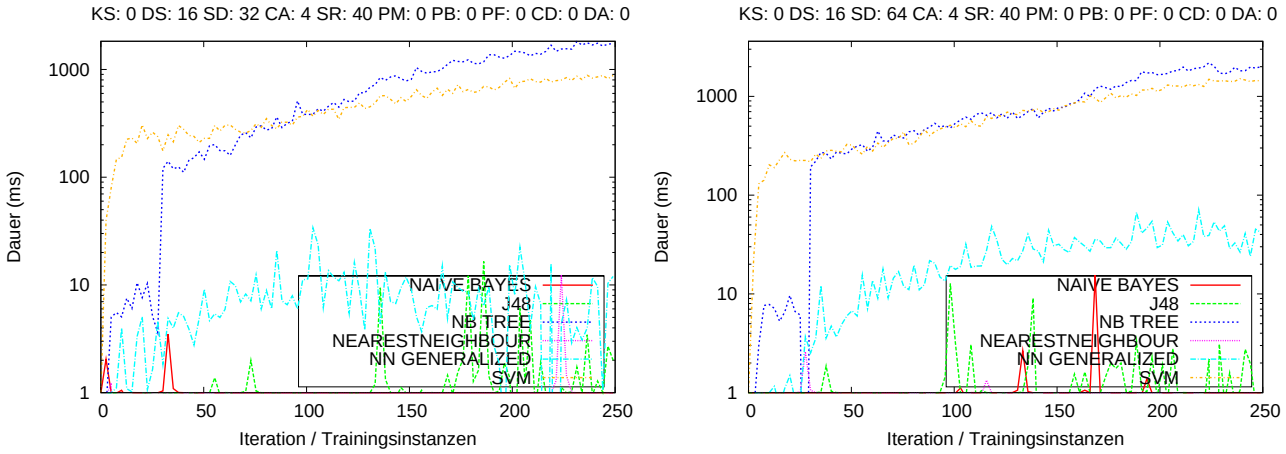
Menge von Sensoren mit Relation zum Konzept

Wie schnell relevante Sensoren zur Entscheidungsfindung bestimmt werden können, beeinflusst ebenfalls die Geschwindigkeit der Lernverfahren. Die Ergebnisse des Vergleichs verschiedener Größen von Mengen relevanter Sensoren sind in Abbildung A.5 dargestellt.

Verdoppelt sich die Größe der Menge relevanter Sensoren, so reduziert sich der Aufwand der Verfahren NB TREE, SVM und J48 um bis zu 50%. Alle anderen Verfahren zeigen dagegen keine nennenswerte Änderung.

Weiterführende Analysen der WAH

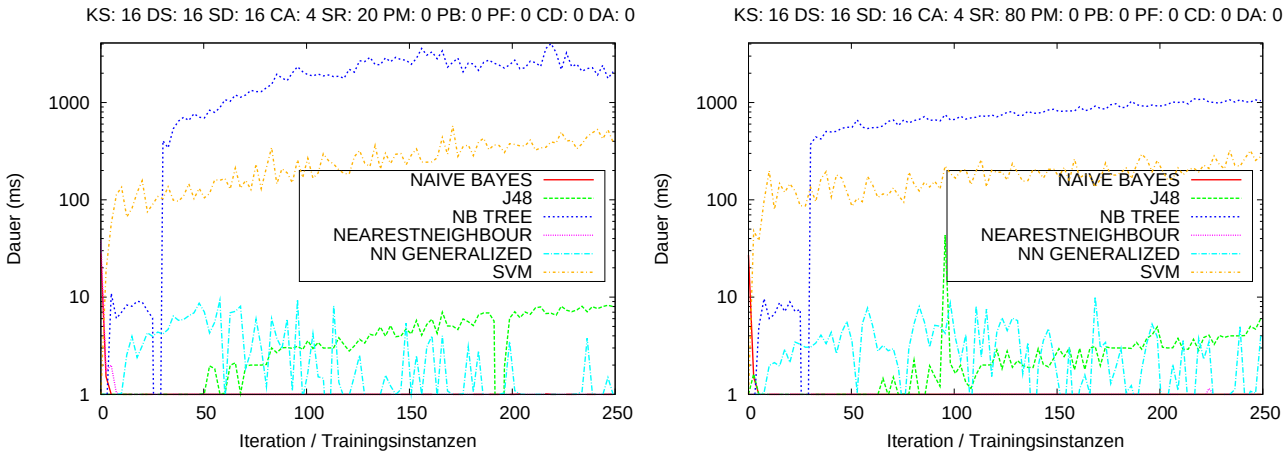
In Abbildung A.6 wird die WAH in den Szenarien mit mehreren kombinierten Eigenschaften genutzt. Hierbei kann ein Vergleich der Genauigkeit mit und ohne WAH durchgeführt werden. Die Basisklassifikatoren ohne WAH wurden in Abschnitt 5.4.7 dargestellt und auf das kombinierte Szenario angewandt. Es ist zu erkennen, dass unter erschwerten Bedingungen keine Verbesserung mehr durch die WAH erzielt werden können, jedoch auch keine deutliche Verschlechterung der Ergebnisse erfolgt. Dieses Verhalten begründet sich auf dem Vorgehen der WAH, welche die Genauigkeit der Klassifikation als Parameter zur Bestimmung der Fenstergröße nutzt. Zudem werden Grenzwerte für die Bestimmung der verschiedenen Phasen des Lernens (siehe auch Abschnitt 5.5.4) genutzt. Unter den erschwerten Bedingungen erreicht die gemessene Genauigkeit nicht mehr die erforderlichen Grenzwerte zur



(a) Diskreter Wertebereich: 32

(b) Diskreter Wertebereich: 64

Abbildung A.4: Vergleich verschiedener Größen des diskreten Wertebereichs

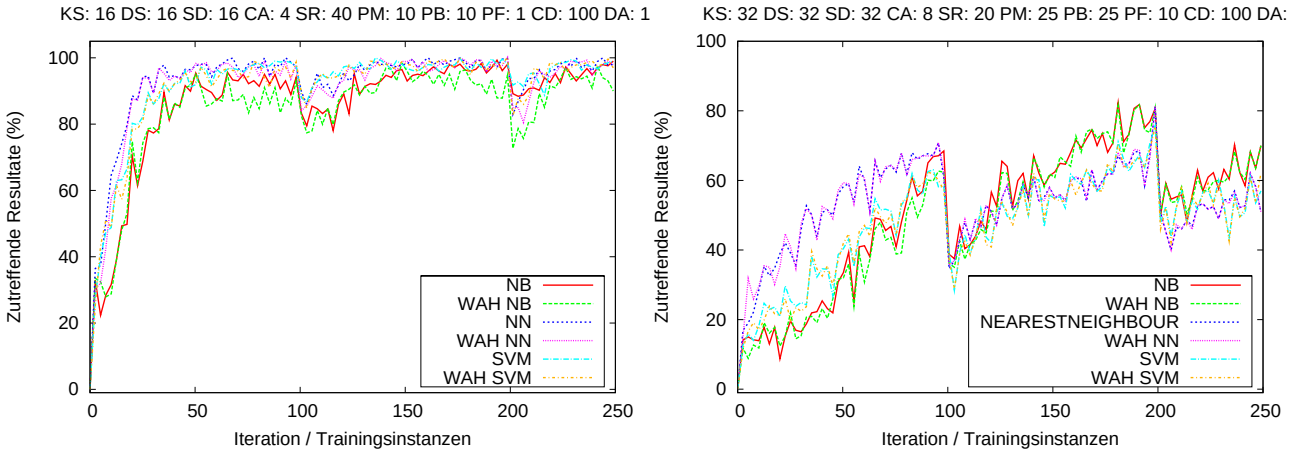


(a) Relation: 20%

(b) Relation: 80%

Abbildung A.5: Vergleich verschiedener Mengen an Sensor Relationen zum Konzept

Bestimmung einer stabilen Phase. Hierdurch geht das Fensterverfahren von einer Fortdauer der Lernphase aus und vergrößert weiterhin die Anzahl der Instanzen im Fenster.



(a) Mäßig Bedingungen

(b) Schwere Bedingungen

Abbildung A.6: Vergleich verschiedener Szenarien und WAH

B Akronyme

ASN.1 *Abstract Syntax Notation One*
CDR *Common Data Representation*
CORBA *Common Object Request Broker Architecture*
DES *Description Element Set*
DPWS *Devices Profile for Web Services*
FLORA *FLOating Rough Approximation*
FLORA-MC *FLOating Rough Approximation-Multivariate Classification*
HTTP *Hypertext Transfer Protocol*
IIOP *Internet Inter-ORB Protocol*
IDL *Interface Definition Language*
IMS *Instant Messaging*
JADE *Java Agent DEvelpment Framework*
J2EE *Java Platform, Enterprise Edition*
JMS *Java Message Service*
R-OSGI *Remote-OSGI*
OWL *Web Ontology Language*
OWL-S *Semantic Markup for Web Services*
PDP *Peer Discovery Protocol*
QoC *Quality of Context*
QoD *Quality of Decision*
QoF *Quality of Feedback*
QoI *Quality of Information*
QoM *Quality of Modell*
QoS *Quality of Service*
RDF *Resource Description Framework*
REST *Representational State Transfer*
RPC *Remote Procedurce Call*
RMI *Remote Method Invocation*
SAWSDL *Semantic Annotations for WSDL*
SIP *Session Initiation Protocol*
SLP *Service Location Protocol*
SNMP *Simple Network Management Protocol*
SOAP *Simple Object Access Protocol*
SSDP *Simple Service Discovery Protocol*
SWRL *Semantic Web Rule Language*
UPnP *Universal Plug and Play*
VoIP *IP-Telefonie*
WCF *Windows Communication Foundation*
WSDL *Web Services Description Language*
WSML *Web Service Modeling Language*
WSMO *Web Service Modeling Ontology*
XML *Extensible Markup Language*
XMPP *Extensible Messaging and Presence Protocol*

C Lebenslauf

Persönliche Daten

Johannes Schmitt

Geboren am 31. Januar 1979 in Frankfurt am Main
Staatsangehörigkeit deutsch
Familienstand verheiratet, 2 Kinder

Akademischer Werdegang

seit 12/2004	Technische Universität Darmstadt Doktorand am Fachbereich Elektrotechnik und Informationstechnik
10/1999-11/2004	Technische Universität Darmstadt Studium zum Diplom-Informatiker
8/1990-6/1998	Freiherr-vom-Stein Gymnasium Frankfurt am Main Erlangung der allgemeinen Hochschulreife

Berufliche Tätigkeiten

Projektleitung der Kooperationen mit Siemens SEN:
SELECTIONS (*SElf LEarning CommunicaTION Services*) (Phasen 1 bis 3)
SMART (*Sensor-based Monitoring for Adv. Reachability in Telecommunication*) (Phasen 1, 2)
Mitarbeiter am BMBF-Projekt:
PROWIT (Prozess-orientierter Web 2.0-basierter integrierter Telekommunikationsservice)

seit 12/04	Wissenschaftlicher Mitarbeiter der Forschungsgruppe Ubiquitous Computing Fachgebiet Multimedia Kommunikation (KOM) Technische Universität Darmstadt
seit 11/04	Senior Entwickler KIMK - Kommunikations- und Informationstechnologie, Medien und Kunst GmBH, Seeheim-Jugenheim

Technical Program Chair Mitglied

WCNC2009	Services and Applications Track 2009 IEEE Wireless Communications and Networking Conference.
PIMRC 2008	Applications, Services and Business Approach, 2008 IEEE International Symposium on Personal, Indoor and Mobile Radio Communications.

D Erklärung laut §9 der Promotionsordnung

Ich versichere hiermit, dass ich die vorliegende Dissertation allein und nur unter Verwendung der angegebenen Literatur verfasst habe.

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.