Kontextabhängige Zugriffskontrolle für Anwendungen im Ubiquitous Computing

vom
Fachbereich Informatik
der Technischen Universität Darmstadt
genehmigte

Dissertation

zur Erlangung des akademischen Grades einer
Doktor-Ingenieurin (Dr.-Ing.)

von

Diplom-Informatikerin Univ. Marie-Luise Moschgath

geboren am 22. April 1966 in Ochsenhausen

Darmstadt 2002
Hochschulkennziffer D17

Erstreferent:	Prof. Dr. Ralf Steinmetz
Koreferent:	Prof. Dr. Peter Kammerer

Tag der Einreichung:	21. Mai 2002
Tag der Disputation:	01. Juli 2002

Eidesstattliche Erklärung

Hiermit erkläre ich, die vorgelegte Arbeit zur Erlangung des akademischen Grades „Dr.-Ing." mit dem Titel „Kontextabhängige Zugriffskontrolle für Anwendungen im Ubiquitous Computing" selbständig und ausschließlich unter Verwendung der angegebenen Hilfsmittel erstellt zu haben. Ich habe bisher noch keinen Promotionsversuch unternommen.

Darmstadt, den 20. Mai 2002

_____

Dipl.-Inf. Univ. Marie-Luise Moschgath

Inhaltsverzeichnis

Abbildungen

Tabellen

Kapitel 1

Einleitung

1.1 Motivation

Die Vision des „Ubiquitous Computing“ wurde von Mark Weiser geboren, dem 1999 früh verstorbenen Chief Technologist vom Xerox Palo Alto Research Center [Wei91]. In seinem damaligen Wunschdenken wurde der Computer zum reinen Mittel zum Zweck, der eine bestimmte Aufgabe erfüllt. Er verschwindet in der Umgebung und wird im Idealfall nicht mehr als solcher wahrgenommen. Nicht mehr die Maschine steht im Zentrum, sondern der Mensch.

Durch die fortschreitende Miniaturisierung der Computertechnologie stehen wir heute erst am Anfang des „Ubiquitous Computing“-Zeitalters. Bestehende und zu erwartende Technologien ermöglichen es, Prozessoren und kleinste Sensoren mehr und mehr in die Gegenstände unserer alltäglichen Umgebung zu integrieren. Die ersten Schritte sind getan. Armbanduhren, Kleider, Schreibstifte, Tassen und Möbel werden „smart“. Sie besitzen eigene Rechenkapazität und die Möglichkeit mittels spontaner Vernetzung untereinander oder mit Personen zu kommunizieren und gegebenenfalls zu kooperieren.

Ubiquitous Computing bedeutet auch, dass eine neue Generation von Applikationen entsteht, die dem Benutzer immer und überall Zugriff auf beliebige, sich ständig ändernde Informationen und Dienste gewährt. Zu dieser Art von Anwendungen zählen beispielsweise Dienste, wie sie in einem „Intelligenten Haus” angeboten werden. Die Benutzung der Kaffeemaschine oder eines Faxgerätes wird überwacht und gesteuert sowie beliebige Informationen über das Hausinformationssystem dem Bewohner bereitgestellt. Das Öffnen von Türen, das Steuern von Licht oder der Heizung, das Betätigen von Jalousien erfolgt elektronisch gesteuert.

Diese neue Art von Anwendungen stellt neue Anforderungen und Herausforderungen an die Technik. Neben Themen wie Skalierbarkeit, Fehlertoleranz, Konsistenz, Nutzbarkeit, werden die Sicherheit und der Datenschutz eine zentrale Rolle einnehmen. Konnte früher das Licht oder die Heizung eines Raums nur vor Ort

geregelt werden, so kann dies künftig von jedem beliebigen Ort aus erfolgen. Die totale Vernetzung in einem Intelligenten Haus, das seinen Nutzern Informationen und Dienste anbietet, seine Bewohner beobachtet und Daten sammelt, um situationsgerecht auf die Bedürfnisse seiner Bewohner reagieren zu können, birgt verschiedene Probleme für die Sicherheit, den Datenschutz und die Privatsphäre, da der Zugriff auf diese Daten und Dienste von überall erfolgen kann.

Um die erhobenen sensitiven Daten und die Dienste vor unautorisiertem Zugriff zu schützen, ist eine Zugriffskontrolle notwendig. Sie gewährt nur denjenigen Zugriff, die über entsprechende Berechtigungen verfügt. Dazu ist notwendig, eine Authentifizierung durchzuführen, die die Identität des Nutzers eindeutig beweist, und eine Autorisierung, welche überprüft, ob dieser Nutzer die erforderlichen Rechte für den Zugriff auf bereitgestellte Informationen und Dienste besitzt.

Alle bekannten Zugriffsmodelle bzw. deren Realisierung in Zugriffskontrollsystemen basieren auf den Authentisierungsprinzipien „etwas tragen", „etwas wissen" oder auf biometrischen Merkmalen. Das erfordert zum einen, dass der Nutzer bei der Authentifizierung aktiv werden muss, und zum anderen, dass das System über Informationen über diesen Nutzer verfügen muss. Nur durch den Vergleich der gespeicherten Informationen über den Nutzer und dessen Rechte mit den bei der Authentifikation vorgelegten Daten kann eine Authentisierungs- und anschließende Autorisierungsentscheidung getroffen werden.

Dieses Prinzip der Authentifizierung und Autorisierung kann in offenen Systemen, wie sie im Bereich des Ubiquitous Computing angestrebt werden, nur begrenzt angewendet werden. In vielen ubiquitären Anwendungsszenarien, so auch in einem Intelligenten Haus, sind dem System unbekannte Nutzer, die Gäste, zu berücksichtigen, welche auf Dienste und Informationen zugreifen möchten und teilweise müssen. Da das IT-System jedoch nicht über Nutzerdaten über die Gäste verfügt, können gewöhnliche Sicherungsmechanismen nicht greifen.

Ein weiteres Problem einer Zugriffskontrolle für den Anwendungsbereich des Ubiquitous Computing ist dessen Philosophie einer humanzentrierten Sichtweise, in der nicht der Mensch mit den Maschinen agiert, sondern die smarte Umgebung auf den Nutzer, dessen Bedürfnisse und Kontexte reagiert. Dies hat neue Mensch-Maschine-Interaktionsformen zur Folge, die eine aktive Authentisierungs- und Autorisierungshandlung erschweren, ja nicht erwünschen, da sie dem Ziel einer unaufdringlichen Interaktionsform widersprechen.

Eine Möglichkeit, das Problem der unbekannten Nutzer zu lösen, ist der Ansatz des „Web-of-Trust" wie es von PGP her bekannt ist [PGP95]. Dabei besitzen alle Nutzer ein Zertifikat. Ausgehend von wenigen bekannten Nutzern, erhalten die dem System unbekannten Nutzer durch bekannte Nutzer eine Sicherheits- bzw. Vertrauensbewertung, anhand derer eine Sicherheitsüberprüfung durchgeführt werden kann. Ob dieser Ansatz für den Bereich des Ubiquitous Computing den gewünschten Erfolg bringt, wird beispielsweise im Projekt Vigil [KUJ+01] untersucht. Jedoch widerspricht der Ansatz des „Web-of-Trust" der angestrebten Nutzerfreundlichkeit einer ubiquitären Umgebung und hat mit den Problemen existierender Public Key-Infrastrukturen zu kämpfen.

Daher wird in dieser Arbeit ein anderer Weg zum Schutz von sensitiven Daten und Diensten eingeschlagen. Während für bestimmte ubiquitäre Dienste eine aktive Authentifizierung und Autorisierung aus Sicherheitsgründen unabdingbar sein wird, können andere ubiquitäre Dienste durch das einfache Einschränken der Zugriffsrechte auf bestimmte Umgebungswerte, so genannte Kontextinformationen, ausreichend vor Missbrauch geschützt werden. Zu diesen Daten gehören beispielsweise Informationen, die der Nutzer selbst in seiner Umgebung wahrnehmen kann, wie die Anwesenheit anderer Personen im gleichen Raum. Dienste, deren Nutzung lokal sinnvoll ist, jedoch bei (missbräuchlicher) Fernsteuerung sogar ärgerlich sein kann, sind beispielsweise die Regulierung der Heizung und die Steuerung von Licht oder Jalousien. Werden solche Dienste durch die totale Vernetzung für jedermann zugänglich, müssen sie gesichert werden, um zu verhindern, dass sie missbräuchlich verwendet werden können. Dies kann durch die Verwendung von Kontextinformationen erfolgen.

Verschiedene Erweiterungen bekannter Sicherheitsmodelle können Kontextinformationen wie die Zeit, das IT-System, den Ort oder auch den aktuellen Zustand des Software-Systems berücksichtigen und die Rechte der Benutzer nach vordefinierten Regeln weiter einschränken. Die Defizite dieser Ansätze sind jedoch die Beschränkung auf einen oder zumindest sehr wenige Typen von Kontextinformationen, die Abstimmung des Sicherheitsmodells auf die verwendeten Kontexttypen und die Integration dieser Zugriffsbeschränkung in die zu schützende Anwendung.

Unter Kontextinformationen werden in dieser Arbeit die verschiedensten Umgebungswerte verstanden, welche messbar, ermittelbar oder berechenbar sind, und Einfluss auf die Zugriffsrechte haben könnten. Dazu zählen beispielsweise in der Umgebung über Sensoren messbare Werte, Zustandsinformationen des Systems, Zustandsinformationen der Anwendung, vorangehende Ereignisse oder Zeit und Ort. Kontextinformationen vom Typ Zeit sowie Zustandsinformation eines Systems sind Informationen, welche mit einfachen Mitteln zu ermitteln sind, und aus diesem Grund auch heute schon zur Einschränkung der Zugriffsrechte verwendet werden. Die Kontextinformation „Ort“, wie sie in den heutigen Modellen und realisierten Systemen eingesetzt wird, beschränkt sich dabei auf den logischen Standort des Endgeräts im Netz (z.B. internes Netz, Sub-Netz oder externer Zugriff), von dem der Zugriff erfolgt, da außer der Netzstruktur in der Regel keine Raummodelle vorhanden sind.

Die „Defizite“ – mangelnde Kontextinformationen und fehlende Raummodelle – werden durch Ubiquitous Computing aufgehoben. Vernetzte Sensoren, integrierte Prozessoren und Raummodelle, wie sie auch für ortsabhängige Dienste (sog. Location Based Services) notwendig sind, ermöglichen die Erfassung nahezu beliebiger Umgebungsinformationen und schaffen so die Voraussetzungen für eine „kontextabhängige Zugriffskontrolle“.

1.2 Ziele der Arbeit

Das zentrale Ziel dieser Arbeit ist es, ein neues Konzept der Zugriffskontrolle zu entwickeln, das eine dynamische Anpassung von Zugriffsrechten mittels komplexer Kontextinformationen an die aktuelle Umgebung bzw. Situation erlaubt. Das Modell der Zugriffskontrolle soll dabei nicht auf einen speziellen Typ von Kontext-

information abgestimmt sein, sondern so allgemein gehalten werden, dass ein breites Spektrum verschiedenster Kontexttypen berücksichtigt werden kann.

Als Grundbaustein für eine kontextabhängige Zugriffskontrolle für Anwendungen im Ubiquitous Computing ist zu untersuchen, was unter Kontext in dieser Fachdisziplin verstanden wird und welche Typen von Kontextinformationen für eine Zugriffskontrolle zu unterstützen sind. Um Kontextinformationen in einem System verwenden zu können, ist es notwendig, eine geeignete Modellierung zu finden, welche für eine permanente Speicherung und Abfrage der Kontextinformationen geeignet ist.

Grundlage der Realisierung einer solchen Zugriffskontrolle bildet eine Spezifikationssprache zur Beschreibung von kontextabhängigen Zugriffsrechten, die einfach, benutzerfreundlich und einfach zu implementieren ist. Diese Sprache soll unabhängig vom verwendeten Zugriffsmodell sowie dessen Implementierung sein. Es ist zu untersuchen, in wie weit verfügbare Spezifikationssprachen geeignet sind, komplexe Kontextbedingungen auszudrücken, und eine Spezifikationssprache zu entwickeln, welche diese Anforderung erfüllt.

Zur Validierung des vorgeschlagenen Konzepts wird eine Infrastruktur für eine kontextabhängige Zugriffspolitik entwickelt. Um weitestgehende Flexibilität und Einsetzbarkeit zu erreichen, soll die Infrastruktur aus zwei unabhängigen Systemen bestehen: einer Infrastruktur zur Definition, Bestimmung und Ermittlung von Kontextinformationen beliebigen Typs und darauf aufsetzend ein verteiltes System zur Festlegung, Überprüfung und Durchsetzung von Zugriffsrechten.

1.3 Verwandte, aber nicht behandelte Themen

Da der Fokus dieser Arbeit auf der Spezifikation bis hin zur Durchsetzung einer kontextabhängigen Zugriffskontrolle liegt, werden die Aspekte der Authentizität und Integrität von gemessenen Kontextdaten nicht betrachtet. Im Falle von Sensor-ermittelten Daten sind diese Forderungen beim heutigen Stand der Technik nicht befriedigend zu erfüllen. Es können weder über die Sicherheit der Sensoren, deren Zuverlässigkeit und Manipulationssicherheit, noch über die Integrität der Daten für die Übertragungsstrecke vom Sensor zum IT-System zuverlässige Aussagen getroffen werden. Dazu wäre eine Sicherheitsinfrastruktur für das Sensornetzwerk notwendig, das beim heutigen Stand der Sensortechnik mit geringen Ressourcen noch schwierig zu implementieren ist.

Ein erster Ansatz für eine Sicherheitsinfrastruktur bei beschränkten Ressourcen, der sich jedoch nicht durchgesetzt hat, ist das Konzept des „Resurrecting Duckling“ [SA99, Sta00]. Um diesem Problem der nicht-garantierten Authentizität und Integrität von Sensordaten gerecht zu werden, sind sowohl Unsicherheits- wie auch Unschärfefaktoren für Kontextdaten bei deren Verarbeitung zu berücksichtigen.

Da die Integrität und die Authentizität von Sensordaten, die ein Teil der Kontextdaten darstellen, auf welche die Zugriffsentscheidung der kontextabhängigen

Zugriffskontrolle basiert, nicht garantiert werden kann, kann eine kontextabhängige Zugriffskontrolle nicht als Sicherheitsfunktion betrachtet werden, sondern „nur“ als sichernde Maßnahme.

1.4 Aufbau der Arbeit

In Kapitel 2 werden zur Veranschaulichung zwei Anwendungsszenarien beschrieben und daraus die Anforderungen an das zu entwickelnde System abgeleitet.

Kapitel 3 diskutiert die unterschiedlichen Auffassungen zu den Begriffen „Kontext“ und „Kontextabhängigkeit“ wie sie in der Literatur des Ubiquitous Computing verstanden werden. Und stellt die wichtigsten Kontextinfrastrukturen, –Toolkits und –Frameworks vor.

Kapitel 4 gibt eine kurze Einführung in die Themen Zugriffskontrollmodelle und Zugriffspolitiken. Anschließend werden die wichtigsten Vertreter von existierenden Zugriffsmodellen, welche Kontextinformationen in geringem Umfang für ihre Entscheidungsfindung verwenden, vorgestellt sowie einige Politiksprachen zur Zugriffskontrolle.

In Kapitel 5 wird die Sichtweise und das Verständnis der Begriffe „Kontext“ und „Kontextinformation“ in dieser Arbeit vermittelt. Diese bildet die Grundlage für die anschließende Modellierung von Kontextinformationen.

In Kapitel 6 wird die im Rahmen dieser Arbeit entwickelte Politiksprache CDACL (Context-Dependent Access Control Language) vorgestellt. Es werden Konfliktlösungsstrategien erläutert und die Ausdrucksstärke von CDACL gezeigt.

Kapitel 7 beinhaltet eine Beschreibung des Konzepts und der Architektur der entwickelten Kontextinfrastruktur, welche aus den zwei großen Teilkomponenten Lokationssystem und Kontextsystem besteht.

Das Konzept, die Architektur sowie die Implementierung der kontextabhängigen Zugriffskontrolle werden in Kapitel 8 beschrieben.

Kapitel 9 fasst die Arbeit zusammen und skizziert mögliche weiterführende Arbeiten.

Kapitel 2

Problemanalyse

In diesem Kapitel werden verschiedene Anforderungen aufgezeigt, die an eine kontextabhängige Zugriffskontrolle im ubiquitären Bereich zu stellen sind. Zum besseren Verständnis der Problematik werden in Abschnitt 2.1 die Ausgangsplattform, der RaumComputer, beschrieben und in Abschnitt 2.2 ein Beispielszenario. Eine Übersicht über die an das System gestellten Anforderungen wird in Abschnitt 2.3 gegeben.

2.1 RaumComputer

Die in dieser Arbeit entwickelte kontextabhängige Zugriffskontrolle inklusive Kontextsystem wurde vor dem Hintergrund einer ubiquitären Umgebung entwickelt. Diese ubiquitäre Umgebung wird durch einer Hard- und Softwareplattform genannt „RaumComputer“ realisiert, die im Projekt „Co-operative Workplaces – Workspaces of the Future“ von R. Reinema et. al. entwickelt wurde [RMB+99, RBB+00, MHR01, Rei01, Rei02, RT02]. Das Ziel bei der Entwicklung des RaumComputers war es, die Lücke zwischen physischer und virtueller Welt zu verringern eine Symbiose beiden Welten zu erreichen.

Ein RaumComputer ist ein eingebettetes System, welches in jedem Raum eines Gebäudes installiert werden kann und eine Reihe von Diensten speziell für diesen Raum zur Verfügung stellt. Als eingebettetes System realisiert, ermöglicht er die Steuerung des physischen Raums mittels Aktoren aus der virtuellen Welt und die Beeinflussung der virtuellen Welt durch die über Sensoren vermittelten physischen Gegebenheiten. Er stellt Rechenkapazität sowie einen Zugang zur Kommunikationsinfrastruktur zur Verfügung.

Ein RaumComputer ist eine autonome Installationseinheit, welche mit dem Internet bzw. Intranet verbunden ist. Er besteht aus einem eingebetteten PC mit Touchscreen, Lautsprecher, Mikrophon und einem angeschlossenen Chipkartenleser.

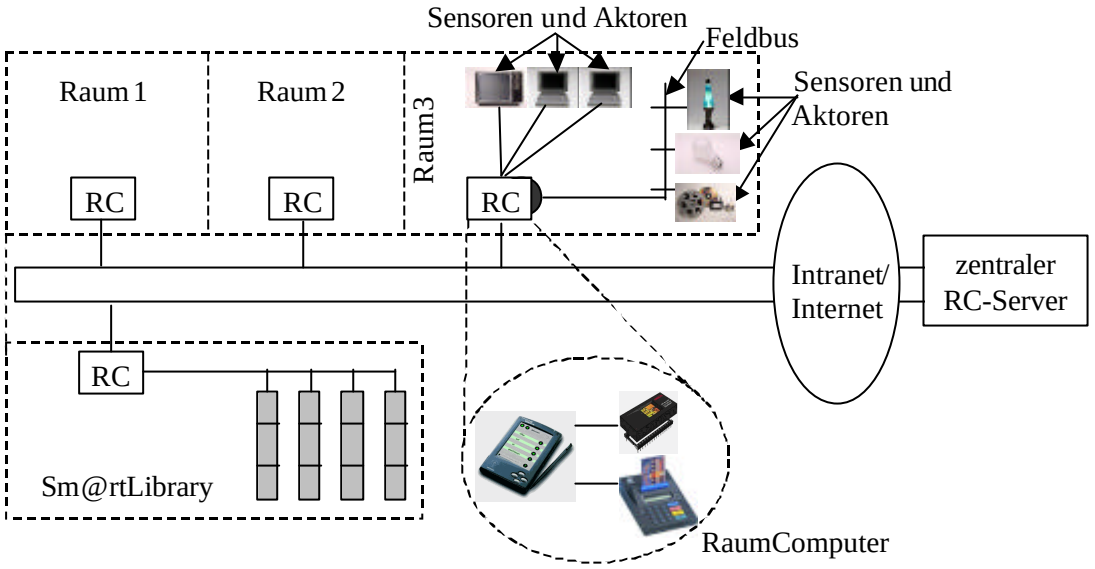

Abbildung 1 Beispiel einer RaumComputer-Installation

Die einzelnen RaumComputer werden über das Intra-/Internet zu einem RaumComputer-Netzwerk verbunden, womit eine Menge von Räumen oder Gebäuden verwaltet werden kann (siehe Abbildung 1). Ein zentraler RaumComputer-Server mit nicht-beschränkender Ressourcenkapazität verwaltet die Konfigurationsdaten der einzelnen, vernetzten Systeme sowie zentrale Dienste.

Durch die Vernetzung der einzelnen RaumComputer über das Intra-/Internet, kann auf die angebotenen Raum-Dienste, die Steuerungen der Aktoren und Sensoren und die durch die Sensoren gesammelten Daten von beliebigen Lokationen zugegriffen und genutzt werden. Durch die Steuerung von Raum-Funktionalitäten wie Licht, Heizung oder Belüftung, durch die Möglichkeiten, die Umgebung und die Bewohner wahrzunehmen und über Aktoren auf Situationen, Bedürfnisse und Notfälle reagieren zu können, kann mit Hilfe des RaumComputer-Netzes das „Intelligente Haus“ realisiert werden. Ein „Intelligentes Haus“ nimmt seine Bewohner und deren Umgebung wahr. Es reagiert situationsgerecht und unterstützt den Bewohner in seinen alltäglichen und beruflichen Aktionen. Es bietet neben Diensten eine Kommunikationsinfrastruktur sowie Informationen an.

2.2 Beispielszenarien

Die folgenden Szenarien sollen verdeutlichen, wie Kontextinformationen in die Handlungen einfließen können. Die Art der Kontextinformation wird, in Klammern gesetzt, hervorgehoben.

Ein Besucher besucht eine Firma zum ersten Mal. Er benötigt für die Besprechung ein Dokument, das er vergessen hat. Das Dokument existiert in elektronischer Form im Intranet seiner Organisation, doch ist der Zugriff darauf ausschließlich über eine gesicherte Verbindung mit starker Authentisierung (→ Sicherheitsstufe) zulässig.

Der Besucher verbindet sich über das lokale Netz vor Ort mit dem Intranet seiner Organisation und holt das Dokument. Anschließend möchte er es auf dem Farbdrucker im Besprechungsraum ausdrucken. Aus Kostengründen steht dieser Dienst nur dem Führungspersonal (→ Status der Person in der Firma) oder temporär den Teilnehmern einer Besprechung (→ temporäre Rolle) während der reservierten Besprechungszeit (→ Zeit) und vom Besprechungsraum (→ Lokation) aus zur Verfügung. Die Freischaltung dieses Druckdienstes erfordert, dass mindestens einer der Besprechungsteilnehmer sich gegenüber dem RaumComputer authentisiert und damit den Besprechungskontext (→ Situationswissen) aktiviert hat. Neben dem Druckerdienst steht im Besprechungsraum auch ein Faxdienst zur Verfügung. Da bei der elektronischen Reservierung des Raums die Besprechung als „vertraulich“ eingestuft wurde, wird nur ein eingeschränkter Faxdienst zugelassen (→ Sicherheitsstufe). Neben den schon für den Drucker genannten Einschränkungen, gilt für den Faxdienst bei vertraulichen Sitzungen, dass jederzeit Dokumente empfangen werden können, jedoch nur gesendet werden kann, wenn bis zu diesem Zeitpunkt der Besprechung keine als vertraulich eingestuften Dokumente kopiert wurden (→ Aktivitäten in der Vergangenheit). Das Kopieren oder Faxen von streng-geheimen Dokumenten ist ausschließlich durch speziell gesicherte Geräte möglich (→ Sicherheitsstufe).

Ein einfacheres Szenario aus dem Alltagsleben: Der Sohn der Familien kommt von der Schule nach Hause. Er möchte sich vor dem Fernseher in seinem Zimmer entspannen, doch steht ihm dieser Dienst erst dann zur Verfügung, wenn er seine Schulaufgaben erledigt hat (→ Aktivitäten in der Vergangenheit) und auch nur bis 20 Uhr (→ Zeit). Nachdem er seine Schularbeiten erledigt hat, möchte er sich den neuesten Abenteuerfilm für Kinder ansehen. Da dieser Film erst ab 12 Jahre freigegeben ist, darf er diesen nicht in seinem Zimmer (→ Lokation), sondern nur im Wohnzimmer in Gegenwart seiner Eltern ansehen (→ Umgebung). Die Mutter steht in der Küche und kocht das Mittagessen. Ein Telefonanruf für sie trifft ein. Da sie jedoch beschäftigt ist (→ Situation, Aktivität), wird der Anruf direkt auf den Anrufbeantworter geleitet. Der Hund des Hauses betritt durch die Hundeklappe das Haus. Da der Nachbarhund weiß, dass es immer gutes Hundefutter gibt, versucht er sich ebenfalls in das Haus zu schleichen. Die Hundeklappe erkennt ihn jedoch und lässt ihn nicht passieren (→ Identität).

Diese beiden Szenarien zeigen, dass Kontextinformationen hilfreich bei der Bereitstellung von Diensten und Informationen aber auch bei der Einschränkung des Zugriffs darauf sein können.

2.3 Anforderungen

An die Realisierung einer allgemeinen kontextabhängigen Zugriffskontrolle, wie sie in dieser Arbeit vorgestellt wird, sind einige Anforderungen zu stellen. Diese An-

forderungen können in allgemeine Systemanforderungen, wünschenswerte Anforderungen, die hilfreich, jedoch nicht notwendig sind, und spezifische Anforderungen, welche durch das Einsatzszenario gegeben sind, eingeteilt werden.

Spezifikation von Kontext und Kontextinformationen

Um eine kontextabhängige Zugriffskontrolle realisieren zu können, ist in einem ersten Schritt festzulegen, was unter den Begriffen „Kontext" und „Kontextinformation" zu verstehen ist. Es ist festzulegen, welche Arten von Kontextinformationen in dieser Arbeit unterstützt werden sollen. Die Minimalanforderungen dabei ist, alle Arten von Kontextinformationen, die bisher in Zugriffskontrollsystemen eingesetzt werden, zu berücksichtigen.

Trennung von Kontextgewinnung und Zugriffskontrollmechanismen

Bisherige Ansätze in Richtung kontextabhängige Zugriffskontrolle verwenden nur in geringem Maße Kontextinformationen und in der Regel nur wenige Arten. Da jedoch die unterschiedlichsten Kontextarten unterstützt werden sollen, wachsen die Komplexität und der Aufwand. Beliebige Kontextinformationen müssen gesammelt, aufbereitet, gespeichert und in die kontextabhängige Zugriffskontrolle einbezogen werden. Da zur Erfüllung dieser Aufgaben sowohl Hardware- wie auch Softwareressourcen in größerem Umfang benötigt werden, ist die erste allgemeine Anforderung an das Gesamtsystem, dass der Bereich der Gewinnung und Aufbereitung von Kontextinformationen von der eigentlichen Zugriffskontrolle getrennt und die Kontextinformationen auch anderen kontextabhängigen Anwendungen zur Verfügung gestellt werden sollen, um die Ressourcen mehrfach nutzen zu können. Ziel dabei ist auch, das Zugriffskontrollsystem übersichtlich und handhabbar zu halten.

2.3.1 Anforderungen an die Kontextinfrastruktur

Betrachtet man die in den vorherigen Abschnitten beschriebenen Anwendungsszenarien, so ist von einer Vielzahl von Sensoren verschiedenen Typs auszugehen und einer hohen Anzahl von kontextabhängigen Anwendungen, die die von den Sensoren ermittelten Kontextinformationen verwenden möchten (siehe Abbildung 2).

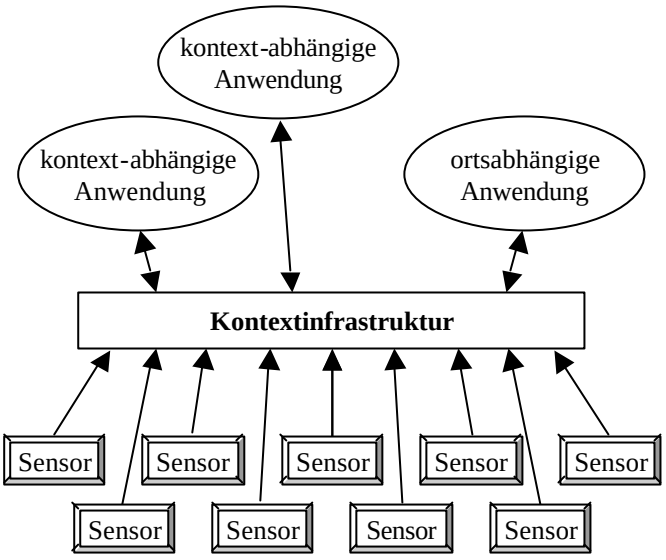

Abbildung 2 Kontextinfrastruktur

2.3.1.1 Allgemeine Anforderungen

Unabhängigkeit der kontextabhängigen Anwendungen von der Sensor-Hardware

Die zentrale Anforderung an die Kontextinfrastruktur muss sein, die Ebene der kontextabhängigen Anwendungen vollständig von der Schicht der Kontext-ermittelnden Sensoren zu trennen. Das entlastet die einzelnen kontextabhängigen Anwendungen davor, selbst Kontextinformationen von den unterschiedlichsten und unzähligen Sensoren abfragen und verarbeiten zu müssen. Die Anwendungen benötigen somit keine Informationen darüber, welche Art von Sensoren in der aktuellen Umgebung vorhanden sind, wie diese arbeiten, welche Daten in welchem Abstraktionsniveau und Datenformat geliefert werden oder wie eine Kommunikation mit diesen Sensoren möglich ist. Dies ist von entscheidender Bedeutung für Anwendungen auf mobilen Geräten, welche sich in beliebiger, sich ständig ändernder Umgebung befinden, oder für Anwendungen, welche von unterschiedlichsten Orten aus entfernt aufgerufen werden können.

Speicherung, Aufbereitung und Bereitstellung von Kontextinformationen

Das System muss in der Lage sein, die von Sensoren ermittelten Kontextdaten aufzunehmen, nach Bedürfnissen zu filtern, weiterzuverarbeiten und zu speichern. Sowohl die aktuellen Daten wie auch gespeicherte Historie-Kontextdaten sollen den jeweiligen Anwendungen zur Verfügung gestellt werden. Die kontextabhängigen Anwendungen sollen die Kontextinformationen sowohl über eine Abfrage- wie auch über einen Benachrichtigungsdienst erhalten können (sog. Pull- und Push-Mechanismen).

Berücksichtigung von Datenschutzrichtlinien für sensitive Daten

In einer ubiquitären Umgebung werden neben allgemeinen Umgebungsvariablen wie beispielsweise Temperatur oder der CO_2-Gehalt der Luft auch Daten über Personen erhoben, welche üblicherweise dem Datenschutz unterliegen. Diese Daten sind gegen in geeigneter Weise vor unberechtigtem Zugriff zu schützen, ohne die Funktionalität der Kontextinfrastruktur zu beeinträchtigen.

Allgemeine funktionale Anforderungen

An die Kontextinfrastruktur werden einige allgemeine funktionale Anforderungen gestellt. Dazu zählen:

- *Skalierbarkeit*: Um die zu erwartende sehr hohe Anzahl von Sensoren bewältigen zu können, muss die Kommunikation sowie das Kontextsystem hoch skalierend sein.
- *Unterstützung von Mobilität*: Das Kontextsystem soll sowohl mobile Sensoren wie auch mobile Anwendungen unterstützen.
- *Plattformunabhängigkeit*: Da unterschiedliche Endgeräte unterstützt werden sollen, muss Plattformunabhängigkeit garantiert werden.
- *Geringe Antwortzeiten*: Benötigt eine ortsabhängige Anwendung Informationen über den eigenen Aufenthaltsort (bzw. des Geräts, auf dem diese Anwendung läuft) oder von Personen und Objekten in der Nähe, so sollten die Antwortzeiten akzeptabel sein. Lange Antwortzeiten verzögern die Weiterarbeit der Anwendung und werden in der Regel von Benutzern nicht akzeptiert.

2.3.1.2 Spezifische Anforderungen

Spezifikationssprache

Es sollte möglich sein, auf einfache Weise weitere Kontextinformationen zu spezifizieren. Dazu ist eine Untersuchung geeigneter Spezifikationssprachen notwendig. Die Spezifikationssprache sollte

- *benutzerfreundlich,*
- *flexibel,*
- *ausdrucksstark* und
- *einfach zu verarbeiten* sein.

Definierte Schnittstellen

Kontextabhängige Anwendungen müssen über eine definierte Standard-Schnittstelle auf beliebige, vorgegebene Kontextinformationen zugreifen können. Es müssen verschiedene Abfragen möglich sein und verschiedene Abstraktionsebenen der Kontextinformationen, welche unterschiedliche Auflösungsgenauigkeiten widerspiegeln, unterstützt werden. Definierte Schnittstellen müssen auch für Sensoren

bereitgestellt werden, damit das Hinzufügen neuer Sensoren wenige bis keine Änderungen im Kontextsystem bedarf.

Berücksichtigung der Kommunikations- und Rechnerinfrastruktur

Durch die Architektur des RaumComputer-Netzes, auf das das zu entwickelnde System aufbaut soll, sind Ressourcenbeschränkungen bei Soft- und Hardware gegeben. Diese Beschränkungen müssen in der Architektur des Kontextsystems spezielle Berücksichtigung finden, um allgemeine Anforderungen wie kurze Antwortzeiten oder Skalierbarkeit zu ermöglichen.

Spezifische funktionale Anforderungen

Neben den genannten allgemeinen funktionalen Anforderungen existieren einige für das zu realisierende System spezifischen Anforderungen:

- *Modularität*: das Kontextsystem sollte so aufgebaut sein, dass eine Erweiterung mit weiteren Raummodellen und die Abbildung der verschiedenen Raummodelle ineinander möglich sind.
- *Unterstützung unterschiedlicher Granularität*: Unterschiedliche Sensoren besitzen verschiedene Genauigkeit bei der Auflösung der Kontextdaten. Dies ist bei den Schnittstellen wie auch in der Architektur der Software zu berücksichtigen.
- *Fehlertoleranz*: Da sowohl Netze ausfallen und damit die Kommunikationsverbindung unterbrochen werden kann, aber evtl. auch Typen von Sensoren eingesetzt werden, die nicht zuverlässig arbeiten, müssen evtl. auftretende Fehler berücksichtigt und Mechanismen zur Konsistenzprüfung und -sicherung eingebaut werden.

2.3.2 Kontextabhängige Zugriffskontrolle

Die Aufgaben der Rechtevergabe und der Zugriffskontrolle bestehen darin, Mechanismen zur Vergabe von Zugriffsrechten zur Verfügung zu stellen und bei einem Zugriff auf ein zu schützendes Objekt die Autorisierung des jeweiligen Subjekts zu prüfen. Allgemeine Anforderungen für diese Aufgaben sind bereits durch verschiedene Kriterienkataloge [DoD85, BSI89, ITSEC91, CC99] zur Bewertung der Sicherheit von IT-Systemen festgelegt.

2.3.2.1 Allgemeine Anforderungen

Zu diesen allgemeinen Anforderungen gehören:

- *Eindeutige Identifizierung*: Die Rechteverwaltung hat die eindeutige und fälschungssichere Identifikation von Subjekten und Objekten zu gewähren.
- *Konflikterkennung*: Die Rechtevergabe muss Konflikte bei der Rechtevergabe erkennen, anzeigen und wenn möglich Mechanismen zur automatischen Konfliktlösung bereitstellen. Das Ziel dabei ist zu verhindern, dass das System in einen inkonsistenten Rechtezustand fällt.

- *Gesicherte Autorisierungsinformationen*: Die Autorisierungsinformationen – dazu zählen die Zugriffsrechte und die Informationen, die zur Prüfung der Zugriffsrechte verwendet werden – sind vor unautorisierter Manipulation zu schützen und fälschungssicher zu speichern.
- *Kontrolle aller Zugriffsversuche*: Es ist sicherzustellen, dass alle Zugriffsversuche kontrolliert werden und die Kontrolle nicht umgangen werden kann.
- *Unmittelbare Kontrolle*: Zwischen der Prüfung der Zugriffsrechte und dem eigentlichen Zugriff dürfen keine Aktionen liegen, die einen Entzug des Zugriffsrechts zur Folge haben könnten.

Allgemeine funktionale Anforderungen

Auch für den Bereich der Zugriffskontrolle gelten die funktionalen Anforderungen:

- *Skalierbarkeit*,
- *Plattformunabhängigkeit* und
- *geringe Antwortzeiten*.

2.3.2.2 Spezifische Anforderungen

Entwicklung einer geeigneten Zugriffspolitiksprache

Eine kontextabhängige Zugriffskontrolle erfordert eine Zugriffspolitiksprache, die fähig ist, die Zugriffsrechte zu spezifizieren. Um eine Zugriffspolitik durchsetzen zu können, ist diese in einer maschinenlesbaren Form zu beschreiben.

Eine Anforderung an diese Arbeit ist es, eine Politiksprache für eine kontextabhängige Zugriffskontrolle zu entwickeln, die folgenden Kriterien genügt:

- Die Politiksprache sollte sowohl maschinen- wie auch menschenlesbar sein, damit sowohl eine einfache Verarbeitung durch die Zugriffskontrollmechanismen wie auch eine einfache, benutzerfreundliche Festlegung einer Zugriffspolitik möglich sind.
- Sie sollte geeignet sein, die gängigsten Sicherheitsmodelle auszudrücken, um auch eine Domänen-übergreifende Kommunikation zu ermöglichen.
- Die Sprache sollte plattformunabhängig sein, damit sie auf verschiedenen Systemen einsetzbar ist. Eine Anwendung in einem neuen System sollte weder eine Änderung der Sprache erfordern oder einen Übersetzungsprozess.
- Die Politiksprache muss eindeutig sein und darf keine Interpretationslücken öffnen, so dass eine Intervention eines Administrators bei der Abarbeitung notwendig wird.

Anwendungsunabhängigkeit

Bei der Entwicklung und Implementierung der Zugriffskontrollmechanismen ist darauf zu achten, dass diese weitestgehend unabhängig von den zu kontrollierenden Anwendungen, den zu kontrollierenden Objekten sowie der Soft- und Hardware-Plattform sind. Das Ziel sollte sein, Mechanismen zu realisieren, die ohne großen Änderungsaufwand in beliebigen Anwendungsszenarien einsetzbar sind.

Berücksichtigung der Kommunikations- und Rechnerinfrastruktur

Wie schon für das Kontextsystem gilt auch für die Zugriffskontrolle, dass die Anforderungen durch die zugrunde liegenden Architektur des RaumComputer-Netzes mit seinen Ressourcenbeschränkungen bei Soft- und Hardware beachtet werden müssen. Nur durch spezielle Berücksichtigung dieser Restriktionen bei der Konzeption der Architektur des Zugriffskontrollsystems können allgemeine Anforderungen wie kurze Antwortzeiten oder Skalierbarkeit erfüllt werden.

Fehlertoleranz

Eine kontextabhängige Zugriffskontrolle hat das Problem, dass evtl. zum Zeitpunkt der Rechteüberprüfung nicht alle benötigten Kontextinformationen zur Verfügung stehen. Gründe dafür können ein temporärer Ausfall des Kontextsystems oder bestimmter Sensoren sein, oder dass bestimmte Informationen an bestimmten Lokationen nicht erhoben werden (können). Das Zugriffskontrollsystem muss in der Lage sein, in geeigneter, kontrollierter Weise auf diese Situation zu reagieren, ohne die Funktionsfähigkeit des Gesamtsystems zu beeinträchtigen.

Kapitel 3

Kontext und kontextabhängige Anwendungen

Kontext und Kontextabhängigkeit sind keine neuen Konzepte. Seit 40 Jahren werden diese Konzepte in verschiedenen Fachbereichen der Informatik untersucht. Am bekanntesten sind dabei die Sprachverarbeitung oder die Mensch-Maschine-Interaktion. Neuer ist der Einsatz im Bereich Mobilkommunikation und des Ubiquitous Computing.

Seit Jahren propagiert die Künstliche Intelligenz, mehr „Intelligenz“ zur Verbesserung der Kommunikation zwischen Mensch und Maschine und des Designs von Benutzungsschnittstellen einzusetzen. In Ansätzen verwenden beide Disziplinen – die Sprachverarbeitung oder die Mensch-Maschine-Interaktion - Kontextinformationen, doch erst durch die technologische Entwicklung sind wir heute in der Lage, durch geeignete Sensortechnologien Kontextinformationen in größerem Umfang zu erfassen, auszuwerten und anzuwenden.

Kontextabhängigkeit wird als eine der Schlüsseltechnologien bei der Entwicklung von ubiquitären Anwendungen angesehen. Sie beschreibt die Möglichkeit einer Anwendung oder eines Geräts seine Umgebung wahrzunehmen, darauf zu reagieren oder sich daran anzupassen. Um besser verstehen zu können, wie Kontextinformationen verwendet und wie kontextabhängige Anwendungen erstellt werden können, werden in den folgenden Abschnitten die unterschiedlichen Auffassungen der Begriffe „Kontext“ und „Kontextabhängigkeit“, wie sie in der Literatur zu finden sind, vorgestellt. Dabei werden ausschließlich Definitionen und Festlegungen aus den Bereichen Ubiquitous Computing und Mobilkommunikation betrachtet, die sich von denen aus den Bereichen Künstliche Intelligenz oder Philosophie unterscheiden. Einige interessante Abhandlungen über den Begriff „Kontext“ in den genannten anderen Disziplinen sind beispielsweise in [Guh91, MB94, BBM95, Bre99, Mot95] zu finden.

In Abschnitt 3.1 werden Erläuterungen und Definitionen für die Begriffe „Kontext“ und „Kontextabhängigkeit“ vorgestellt. Der Unterschied von „Kontext“ zu den im Ubiquitous Computing häufig verwendeten Begriffen „Situation“ und „Environment“ wird in Abschnitt 3.2 genauer beleuchtet. In Abschnitt 3.3 wird ein Überblick über die relevanten Arbeiten aus den Bereichen Kontextinfrastrukturen, Frameworks und Toolkits gegeben.

3.1 Kontext und Kontextabhängigkeit

Der Begriff „Kontextabhängigkeit“ (engl. context awareness) wurde für den Bereich Ubiquitous Computing 1994 von Bill N. Schilit eingeführt [SAW94]. Er klassifizierte damit eine neue Art von Anwendungen, die die Gegebenheiten in der Umgebung eines (mobilen) Anwenders aufnehmen und in die Verarbeitung einbeziehen. Diese Anwendungen überwachen die Umgebung, in der sie ablaufen, und adaptieren ihr Verhalten entsprechend des Kontexts und seiner Veränderungen. Typischerweise wird unter Kontext die Lokation, die Identität des Anwenders sowie welche Personen oder Ressourcen sich möglicherweise in der näheren Umgebung befinden verstanden. Prinzipiell ist Kontext jedoch nicht auf die genannten Kategorien eingeschränkt. Er kann beliebige physische und logische Umgebungsvariablen umfassen.

Eine neuere Definition des Begriffs „Kontext“ stammt von Dey und Abowd. In [DA00a] definieren sie Kontext als „any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves“. Nach ihrer Definition ist eine Anwendung genau dann kontextabhängig, wenn „it uses context to provide relevant information and/or services to the user, where relevancy depends on the user’s task“.

Eine etwas andere Sichtweise vertreten Lieberman und Selker in [LS00]. Sie definieren Kontext als „everything that affects the computation except the explicit input and output“. Unter explizitem Input wird jegliche Benutzer-Interaktion verstanden, unter implizitem Input alles andere. Eine Konsequenz dieser Definition – die die Autoren selbst ziehen – ist, dass Kontext davon abhängt, wo die Grenzen des betrachteten Systems gezogen werden. Während bei Dey et al. der Kontext bzw. die Situation im Mittelpunkt stehen, stellen Lieberman et al. das betrachtete System ins Zentrum der Betrachtung.

Schmidt et al. [SBG99] beschreiben Kontext als „a situation and the environment a device or user is in“. Diese Definition verwendet weitere unklare Begriffe für den Versuch einer Erklärung. Dass dies ein fragwürdiger Versuch einer Definition ist, zeigt die Begriffsverwirrung bzgl. „Kontext“, „Situation“ und „Environment“ in der Literatur, in der diese Begriffe oftmals äquivalent verwendet werden (siehe dazu Abschnitt 3.2).

Rakotonirainy et al. [RLF00] schränken den Begriff „Kontext“ gegenüber der Definition von Dey et al. wieder etwas ein. Ihrer Meinung nach spielt nicht nur die Relevanz der Kontextinformation für die Anwendung eine Rolle, sondern auch die Verfügbarkeit. Wechselt ein Anwender oder eine Anwendung ihre Umgebung, so ist nicht garantiert, dass die gleichen Kontextinformationen zur Verfügung stehen.

Deshalb definieren sie Kontext als „information that can be used to characterize the situation of an entity and can be obtained by the entity, where an entity can be a person, place, physical or computational object“.

3.2 Kontext, Situation und Environment

In der Literatur tauchen neben den Begriffen „kontextabhängig“ (engl. context-aware) und „Kontext“ auch die Begriffe „situationsabhängig“ (engl. situation-aware oder situated) und „Umgebung“ (engl. environment) auf. Der Unterschied bzw. die Ähnlichkeit soll anhand einiger Definitionen verdeutlicht werden.

Im Duden Fremdwörter neu [DFn01] sind folgende Definitionen für diese Begriffe zu finden:

Kontext <*lat.*> *der;* -[e]s, -e: 1. (Sprachw.) a) der umgebende Text einer sprachlichen Einheit; b) (relativ selbstständiges) Text- od. Redestück; c) der inhaltliche (Gedanken-, Sinn)zusammenhang, in dem eine Äußerung steht, u. der Sach- u. Situationszusammenhang, aus dem heraus sie verstanden werden muss; vgl. Kontext. 2. Zusammenhang.

Situation <*lat.-mlat.-fr.*> *die;* -, -en: 1. [Sach]lage, Stellung, [Zu]stand. 2. Lageplan (Geogr.). 3. die Gesamtheit der äußeren Bedingungen des sozialen Handelns u. Erlebens (Soziol.).

Environment <*engl.*> *das;* -*s*, -*s*: Kunstform, die eine räumliche Situation durch Anordnung verschiedener Objekte u. Materialien (z. B. Sand, Blütenstaub) herstellt (Kunstw.).

DER BROCKHAUS multimedial 2001 [DBm01] enthält im Gegensatz zu [DFn] nur kurze Definitionen:

Kontext *[lateinisch]*, Zusammenhang, Umfeld.

Situation *[französisch]* die, Kartographie: Lage, Grundriss.

bzw. in der Philosophie :
… der einmalige, unwiederholbare Augenblick, in dem sich für den Einzelnen die unmittelbare konkrete Wirklichkeit darstellt und ihn zur Entscheidung zwingt (Situationsethik).

Environment: das (Ambiente), Ausdrucksform der bildenden Kunst in der 2. Hälfte des 20.Jahrhunderts, die aus Assemblage und Combine-Painting entwickelt wurde und wichtige Impulse aus der Happeningbewegung erhielt. Das Environment besteht aus einer räumlich definierten Anordnung verschiedenartiger Materialien und/oder (Gebrauchs-)Gegenstände und bezieht den Betrachter unmittelbar ein.

Betrachtet man den Begriff „Environment“ als einen einfach aus dem Englischen übernommenen Begriff – wie es in der Informatik üblich ist -, so findet man in Langenscheidts Handwörterbuch [LH01] folgende kurze und prägnante Übersetzung:

environment Umgebung *f*; Umwelt *f*

Vergleicht man diese Definitionen der Begriffe „Kontext“, „Situation“ und „Environment“, so erkennt man, dass die Begriffe sich im Wesentlichen nur im Betrachtungswinkel unterscheiden. Bei dem Begriff „Kontext“ steht der betrachtete Text, die Handlung bzw. die Entscheidung im Mittelpunkt, beim Begriff „Situation“ die Umgebung, welche jedoch Einfluss auf das Kernstück Text/die Handlung/die Entscheidung nimmt. Die Übersetzung des englischen Begriffs „Environment“ betrachtet nur die Umgebung und schließt damit das Kernstück aus, während der in der Kunst anzutreffende eingedeutschte Begriff explizit den Betrachtenden wieder mit einbezieht. Interessant dabei ist auch die Betonung des Räumlichen.

Da der Betrachtungswinkel bei der Verwendung der Begriffe „Kontext“, „Situation“ und „Environment“ im Ubiquitous Computing in den wenigsten Fällen eine Rolle spielt und die Unterschiede in der Bedeutung marginal sind, können diese Begriffe synonym verwendet werden.

3.3 Relevante Arbeiten

In den folgenden Abschnitten werden verschiedene Kategorisierungsversuche für Kontextinformationen und kontextabhängige Anwendungen in der Literatur des Ubiquitous Computing vorgestellt.

3.3.1 Kategorien von Kontextinformationen

Für die maschinelle Verarbeitung von Kontextinformation ist eine Modellierung notwendig, deren Grundlage die Klassifikation darstellt. Daher werden im Folgenden einige Klassifikationsversuche für Kontextinformationen aus der Literatur des Ubiquitous Computing vorgestellt.

Schilit et al. [SAW94] identifizieren als die drei wichtigsten Aspekte von Kontextinformationen:

- die Identität des Benutzers,
- die Personen und
- die Ressourcen, die sich in der Nähe befinden.

Sie betonen jedoch, dass Kontext mehr als nur die Lokation des Benutzers umfasst. Licht- und Geräuschverhältnisse, Netzverbindungen, Kommunikationskosten, Bandbreite sowie das soziale Umfeld können ebenso von Interesse sein.

Im Gegensatz zu Schilit et al. unternehmen Schmidt et al. in [SBG99] einen konkreteren Versuch zur Klassifikation von Kontextinformationen, welcher stark an

objektorientierte Programmiermodelle angelehnt ist. Sie stellen folgendes Arbeitsmodell für Kontext auf:

- Ein Kontext wird durch einen eindeutigen Namen identifiziert.
- Für jeden Kontext ist eine Menge von Eigenschaften relevant.
- Für jede relevante Eigenschaft ist ein Wertebereich implizit oder explizit durch den Kontext festgelegt.

Ausgehend von diesen Grundfeststellungen propagieren sie ein hierarchisches Modell von Kontexteigenschaften. Auf der obersten Ebene werden zwei Kategorien unterschieden, der menschliche Faktor und der Umgebungsfaktor, auf der zweiten Ebene jeweils drei Unterkategorien (siehe Abbildung 3). Auf jeder weiteren Ebene ist die weitere Unterteilung von den jeweiligen Kontext-Eigenschaften abhängig.

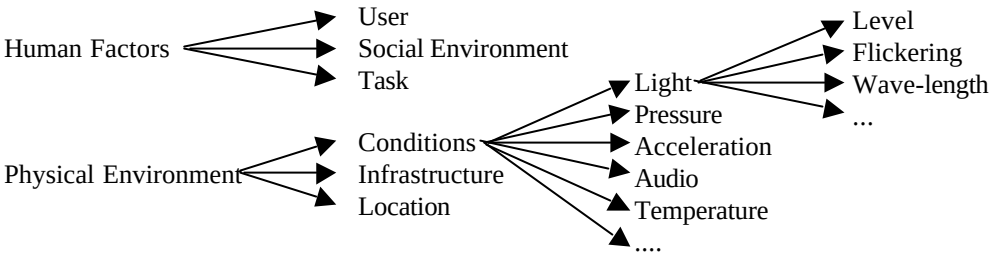

Abbildung 3 Context Feature Space [SBG99]

Für Dey und Abowd [DA00a] ist Kontextinformation das, was die W-Fragen beantworten: „who, what, when, and where“. Sie zählen Ortsinformation, Identität, Zeit und Aktivität zu den primären Kontextinformationen, aus denen die sekundären Kontextinformationen, wie beispielsweise E-Mail oder Beruf einer Person, abgeleitet werden können.

Salber [Sal00] kategorisiert Kontext nach Art des „Kontext-Lieferanten“ und den Attributen. Zu möglichen Lieferanten zählen der Menschen, Orte, physikalische und virtuelle Objekte; die Attribute können Ortsinformation, Identität, Aktivität oder Zustand sein.

Einen weiteren Versuch einer Kategorisierung unternehmen Chen und Kotz in [CK00]. Sie unterscheiden fünf Kategorien von Kontext, wobei sie die ersten drei von Schilit et al. übernommen haben:

- *Computing Context*: dazu zählt beispielsweise die Art der Netzverbindung, Kommunikationskosten, Bandbreite oder die sich in der Nähe befindenden Ressourcen wie Rechner und Drucker,
- *User Context*: umfasst Benutzerprofile, Aufenthaltsort, sich in der Nähe befindende Personen oder das soziale Umfeld,
- *Physical Context*: umfasst Licht-, Geräuschverhältnisse oder Temperatur,

- *Time Context*: umfasst Tageszeit, Woche, Monat, etc. und
- *Context history*: die sich aus vorangegangenen Konstellationen von Kontextinformationen ableitet.

Während die Kategorisierung von Schilit et al. sehr einengend ist, orientieren sich Schmidt et al. zum einen sehr stark an einer möglichen technischen Umsetzung und zum anderen nehmen sie von Rechner nur schwer erfassbare Parameter wie bspw. das soziale Umfeld in ihre Klassifikation auf. Die umfassendste Klassifikation stammt von Chen und Kotz. Hier wird beispielsweise Zeit als Klasse und nicht nur als Informationseinheit betrachtet und die Historie des Systems als Kontextinformation spezifiziert. Was jedoch in keinem der Klassifikationsversuche deutlich wird, sind die Beziehungen zwischen den unterschiedlichen Kategorien von Kontextinformationen. Der Versuch, diesen Mangel zu beheben, wird in Kapitel 5 dieser Arbeit unternommen.

3.3.2 Kategorien von kontextabhängigen Anwendungen

Um das Phänomen kontextabhängige Anwendungen besser verstehen und damit eine bessere Unterstützung bei deren Entwicklung anbieten zu können, ist eine Kategorisierung kontextabhängiger Anwendungen nützlich.

Einer der ersten Kategorisierungsversuche stammt von Schilit et al. [SAW94]. Sie kategorisieren kontextabhängige Anwendungen in vier Gruppen mit zwei orthogonalen Dimensionen: ob Informationen bereitgestellt oder Kommandos ausgeführt werden, und manuelle oder automatische Erhebung. Die Kategorien sind:

- *proximate selection:*

  Darunter werden Anwendungen verstanden, welche die Sicht auf Personen, Dinge oder Dienste in ihrer Nähe erlauben bzw. solche von besonderer Relevanz hervorheben.

- *automatic contextual reconfiguration:*

  Der interessante Aspekt dabei ist, wie Kontextinformation dazu verwendet werden können, unterschiedliche Systemkonfigurationen zu erkennen und zu adaptieren, um die Zusammenarbeit verschiedener Systeme zu erleichtern.

- *contextual information and commands:*

  Darunter fallen Anwendungen, die abhängig vom aktuellen Kontext dem Benutzer Informationen bereitstellen oder Anweisungen ausführen.

- *context-triggered actions:*

  Kontext-getriggerte Anwendungen gehorchen einfachen IF-THEN-Regeln. Ändert sich der Kontext so, dass eine Regel aktiviert wird, wird die Anwendung entsprechend der Konklusion der Regel abgeändert oder eine Aktion ausgelöst.

Eine andere Kategorisierung nehmen Peter Brown et al. in [BBL+00] vor. Sie teilen kontextabhängige Anwendungen in folgende sechs Gruppen ein:

- *proactive triggering:*

  Diese Gruppe von Anwendungen zeichnet den Kontext des aktuellen Nutzers auf und stellt diesem Informationen oder Dienste bereit, die aktuell relevant erscheinen.

- *streamlining interaction:*

  Um die Kommunikation zwischen mehreren Personen zu unterstützen, ermöglicht diese Art von Anwendungen eine einfache begleitende Kommunikation. Die Geräte kennen ihren eigenen Kontext und bieten die passenden Dienste an. Ziel ist es, eine parallel stattfindende Mensch-zu-Mensch-Kommunikation in geeigneter Form technisch zu unterstützen.

- *memory for past events:*

  Der aktuelle Kontext eines Anwenders wird ermittelt und archiviert, um für spätere Auswertungen verfügbar zu sein.

- *reminders for future contexts:*

  Diese Gruppe von Anwendungen erlaubt die Festlegung von zukünftigen Situationen (Konstellationen von Kontextinformationen) und Reaktionen. Tritt eine der spezifizierten Situationen ein, so reagiert die Anwendung wie festgelegt.

- *optimizing patterns of behavior:*

  Diese Anwendungskategorie zeichnet ihren Kontext und das Verhalten der vorhandenen Akteure auf, um es zu einem späteren Zeitpunkt zu analysieren und um Verbesserungsvorschläge für die dann aktuelle Situation machen zu können.

- *sharing experiences:*

  Kontext wird hier zur Unterstützung von Gruppenarbeit eingesetzt. Er wird erfasst und anderen Gruppenmitgliedern zur Verfügung gestellt.

Eine weitere Klassifikation kontextabhängiger Anwendungen bietet Hans-W. Gellersen [Gel00]. Er teilt kontextabhängige Anwendungen ein in:

- ortsbezogene,
- objektbezogene und
- situationsbezogene Systeme.

Ortsbezogene Systeme verwenden die Lokationsinformation von Systemkomponenten, objektbezogene die Identität von Personen und Objekten und situationsbezogene die Informationen über die Aktivitäten in der Systemumgebung.

Daniel Salber beschreibt in [Sal00] seine drei Kategorien von Context-Aware Applications:

- *Presenting information and services:*

  Darunter sind Anwendungen zu verstehen, welche dem Benutzer an die aktuelle Umgebung angepasste Informationen oder Dienste anbieten.

- *Automatically executing a service:*

  Diese Art von Anwendungen reagiert selbständig auf Veränderungen in der Umgebung und passt sich dieser an, um so optimal für und auf den Benutzer reagieren zu können.

- *Attaching context information for later retrieval:*

  Diese Gruppe von Anwendungen versieht aufgezeichnete Daten mit relevanten Kontextinformationen, um zu einem späteren Zeitpunkt nicht nur auf die Daten, sondern auch auf den Erfassungskontext zugreifen zu können.

Alle vorgestellten Versuche der Kategorisierung von kontextabhängigen Anwendungen zeigen, welche unterschiedlichen Ausprägungen möglich sind. Eine Bewertung ist nur schwer möglich, da alle Kategorisierungsversuche unter einem individuellen Blickwinkel erfolgten, und damit den unterschiedlichen Arten von kontextabhängigen Anwendungen unterschiedliche Wichtigkeit zuordneten.

So unterschiedlich die Kategorisierungsversuche auf den ersten Blick auch erscheinen mögen, so können doch als Schlussfolgerung drei zentrale Funktionalitäten kontextsensitiver Systeme identifiziert werden:

- das Beobachten der Umgebung durch eine Anwendung und die Anpassung an dieselbe,
- das aktive Anstoßen einer Anwendung von außen durch Eintreten eines bestimmten Kontexts, um eine Aktion auszulösen oder eine erneute Anpassungen der Anwendung an einen neuen Kontext zu veranlassen, und
- das Aufzeichnen von Kontextinformationen für Auswertungen zu einem späteren Zeitpunkt.

3.3.3 Kontextarchitekturen, Frameworks und Toolkits

Unzählige Arbeiten zeigen, dass Kontextabhängigkeit ein zentrales Thema des Ubiquitous Computing ist. Die Mehrzahl der Projekte verwendet jedoch nur sehr wenige unterschiedliche Kontextklassen und Kontextinformationen und noch dazu in sehr geringem Umfang. Die Ursache dafür liegt im Fehlen von unterstützenden Strukturen, sowohl bei der Erstellung von kontextabhängigen Anwendungen, wie auch im Betrieb. In der Mehrzahl der Anwendungen werden die Kontextinformationen speziell für die jeweilige Applikation ermittelt und in proprietärer Weise dieser zur Verfügung gestellt. Dieses Vorgehen zeigt, dass die Konzeption, die Entwicklung und die Wartung von kontextabhängigen Anwendungen im Gegensatz zu herkömmlichen Anwendungen sehr schwierig und aufwendig ist.

Dieses Manko der fehlenden Unterstützung wurde sehr früh erkannt und in einigen Ansätzen versucht zu beheben. In verschiedenen Forschungsprojekten wurden allgemeine Architekturen, Frameworks oder Toolkits entwickelt, die die Entwicklung kontextabhängiger Anwendungen und das Erfassen von Kontextinformationen unterstützen. Die wichtigsten Ansätze werden in den folgenden Abschnitten vorgestellt.

3.3.3.1 Schilits Systemarchitektur für context-aware mobile computing

Aufbauend auf dem PARCTAB Mobile Computing System [AGS+93, SAG+93], welches am Xerox Palo Alto Research Center entwickelt wurde, realisierte William Noah Schilit als einer der ersten ein System zur Unterstützung von kontextabhängigen mobilen Anwendungen [Sch95].

Die Systemarchitektur spiegelt Schilits Auffassung von Kontext und Kontextabhängigkeit wider (siehe Abschnitt 3.3.1 und 3.3.2). Seiner Auffassung nach, verwenden kontextabhängige Anwendungen im Wesentlichen Informationen über Personen, Dinge und deren Beziehungen zueinander. Informationen über eine Person sind beispielsweise deren Vorlieben; Informationen über Dinge beispielsweise ihr Status, wie „an“, „aus“, „untätig“ oder „beschäftigt“.

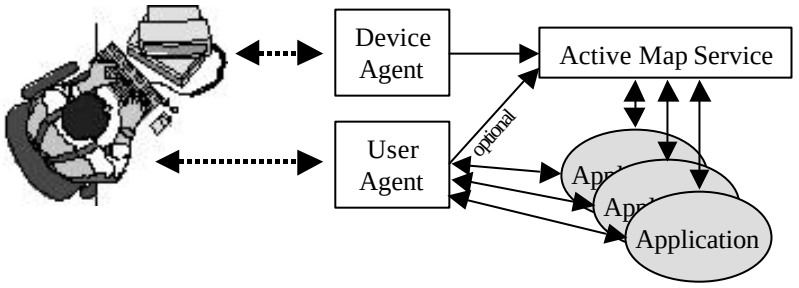

Abbildung 4 Schilits Systemarchitektur

In Schilits Systemarchitektur wird jede Person von einem User Agent repräsentiert, jedes Gerät von einem Device Agent (siehe Abbildung 4). Der Unterschied des User Agent zum Device Agent liegt darin, dass jeder Benutzer seinen Agent individuell konfigurieren kann. Um dem Datenschutz und dem Schutz der Privatsphäre zu genügen, kann jeder Anwender selbst bestimmen, welche Informationen an wen weitergegeben werden.

Ein Device Agent repräsentiert genau ein Gerät und dessen Zustand. Ändert sich der Zustand eines Gerätes, so wird diese Information an den Active Map Service propagiert. Ein Device Agent ist spezifisch für das Gerät, das er repräsentiert.

Der Active Map Service (AMP) sammelt die Informationen über die verschiedenen Geräte der einzelnen Regionen. Diese benötigt er, um Anfragen zum Beziehungskontext, wie beispielsweise „wer befindet sich in der Nähe von wem“, beantworten zu können. Der AMP alleine kennt die räumlichen Beziehungen, da nur hier alle Informationen zusammenlaufen. Jeder AMP kann dabei mehrere Regionen verwalten. Für eine Region ist jedoch genau ein AMP zuständig.

Schilits Arbeit ist eine der ersten in diesem Themenbereich. Sie ist richtungsweisend, da sie wesentliche Grundlagen erarbeitet und den ersten Schritt in Richtung Systemunterstützung für kontextabhängige Anwendungen darstellt. Doch ist Schilits Auffassung von Kontext sehr eng gefasst. Er unterstützt weder Zeit noch Aktivität als Kontextinformation. Auch erfolgt keine Aufzeichnung von Kontextinformationen, wodurch eine Auswertung zu einem späteren Zeitpunkt nicht möglich ist. Seine Agenten unterstützen lediglich physische Sensoren und sind speziell auf die Anwendungen zugeschnitten. Durch den rein zentralistischen Ansatz skaliert dieses System nicht und kann nicht für größere Mengen von Kontextinformationen ausgebaut werden.

3.3.3.2 Stick-e Document Framework

Das stick-e document Framework wurde an der University of Kent unter Federführung von Prof. Brown entwickelt. Es ist ein Framework zur Unterstützung des Entwicklungsprozesses von speziellen kontextabhängigen Anwendungen [Bro96a, Bro96b, Pas97]. Das Framework stellt Mechanismen bereit, sowohl für die Spezifikation eines Kontexts, bei dessen Eintreten eine Aktion ausgelöst wird, wie auch für die auszuführenden Aktionen.

Ein stick-e-Dokument ähnelt einem WWW-Dokument. Es besteht aus kleineren Komponenten, den so genannten „stick-e notes“. Stick-e notes stellen eine elektronische Variante der bekannten Post-It Notes dar und können vergleichbar eingesetzt werden. Wie Post-It Notes an Gegenstände geheftet werden können, so können stick-e notes mit einem Kontext assoziiert werden. Stick-e notes bestehen aus einem Inhalt und einer Kontextspezifikation. Trifft der spezifizierte Kontext ein, so wird der Inhalt des stick-e notes ausgelöst. Der Inhalt kann aus einer Nachricht, Anweisungen oder einem Programm bestehen. Eine Nachricht kann beispielsweise im Internet veröffentlicht, dem Benutzer auf einem beliebigen Gerät angezeigt oder aber an andere Personen weitergeleitet werden. Anweisungen oder Programme werden ausgeführt.

Ein besonderes Kontextelement bei stick-e notes sind die so genannten „Geister“ (engl. spirits). Mit „Geist“ ist ein imaginärer Begleiter eines Benutzers gemeint, der vom Benutzer frei definiert werden kann, um einen bestimmten Kontext, ein Thema oder aber auch eine reale Person zu repräsentieren, die nicht über das System automatisch erfasst und damit auch nicht über eine eigene virtuelle Repräsentation verfügt. Der Nutzen dieser Geister liegt darin, dass bestimmte Kontexte definiert werden können, denen dann stick-e notes angeheftet werden können.

Um die Portabilität der stick-e-Nachrichten zu gewährleisten, werden diese in SGML spezifiziert [Bro96a]:

<note some attributes>
<required>
<with> J.D. Bovey <or> X. Chen
<at> co-ordinates of location
<content>
This is the content.
</note>

Das Schlüsselwort <required> leitet die Beschreibung des Kontextes ein. Über das Schlüsselwort <with> kann spezifiziert werden, in wessen Anwesenheit der Inhalt ausgelöst wird und mit <at> an welchem Ort. Weitere Details zur Spezifikation sind in [Bro96b] zu finden.

Die Architektur des stick-e notes-Frameworks besteht aus folgenden Komponenten:

- *SEPREPARE*: unterstützt die Erzeugung von stick-e notes,
- *SEMANAGE*: verwaltet die Notes,
- *SETRIGGER*: triggert die stick-e notes, deren spezifizierter Kontext wahr wird,
- *SESHOW*: speichert die aktivierten Notes und präsentiert sie dem Benutzer.

Pascoe et al. [PRM98, PRM99] führten die Arbeit am stick-e document framework weiter, mit dem Fokus, eine verbesserte Mensch-Maschinen-Interaktion zu erreichen.

Das stick-e document framework unterstützt die Entwicklung von kontextabhängigen Anwendungen in der Art, dass es möglich wird, Anwendungen ohne Programmierkenntnisse kontextabhängig zu gestalten. Der Nachteil ist jedoch, dass das Einsatzgebiet sehr stark auf eine spezielle Art von Anwendungen eingeschränkt wurde. Es ist nicht möglich, komplexere Kontextinformationen zu verwenden, wie beispielsweise räumliche Beziehungen oder die Historie von Aktionen.

3.3.3.3 Situated Computing Service

Der Situated Computing Service (kurz: SitComp Service) wurde von Hull et al. in den Hewlett Packard Laboratories entwickelt [HNB97]. Der SitComp Service stellt einen Teil einer Middleware dar, welcher den Zugriff auf lokale Sensoren unterstützt.

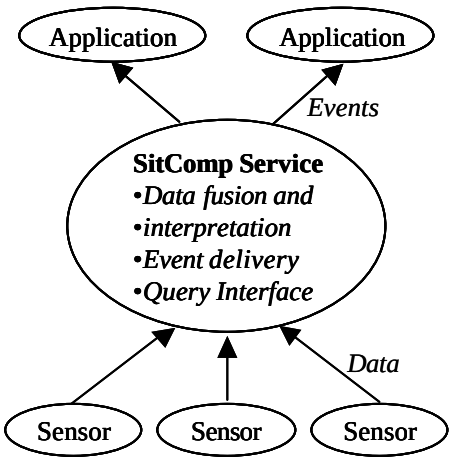

Abbildung 5 SitComp Service

Der SitComp Service abstrahiert und kapselt die lokalen Sensordaten, indem er diese durch Sensor Fusion kombiniert und zu höherwertigen Kontextinformationen interpretiert (siehe Abbildung 5). Diese können über Ereignisse (engl. Events) an interessierte Anwendungen geschickt oder über ein standardisiertes Interface abgefragt werden.

Der SitComp Service wurde für Anwendungen entwickelt, welche auf einem einzelnen Rechner laufen. Die Architektur erlaubt keinen Fernzugriff, eine sehr gravierende und nicht-akzeptable Einschränkung. Auch bietet die Architektur keine Unterstützung für die Archivierung von Daten oder der Auswertung der Historie.

3.3.3.4 Generic Context Server

Der Generic Context Server wurde, wie das Context Toolkit, am Georgia Institute of Technology entwickelt [SA98]. Die Zielsetzung von Daniel Salber und Gregory D. Abowd war die Entwicklung einer Infrastruktur zur Speicherung, Bereitstellung und Archivierung von Kontextinformationen.

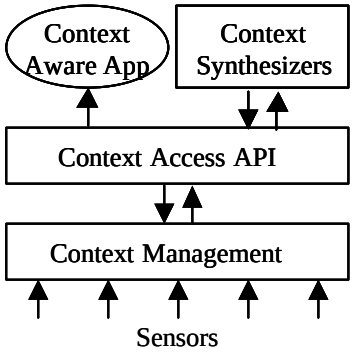

Abbildung 6 Context Server

Die Infrastruktur setzt sich aus mehreren Kontextservern zusammen. Auf jedem stationären oder mobilen Rechner, auf dem Kontextdaten gesammelt werden, läuft ein Kontextserver. Dieser besteht aus folgenden funktionalen Komponenten (siehe Abbildung 6):

- *Context Management:* sammelt und speichert die Kontextinformationen,
- *Context Access API:* stellt eine Schnittstelle für Applikationen zur Verfügung, um lokalen oder Fernzugriff auf die Kontextdaten zu ermöglichen,
- *Context Synthesize:* generiert durch Abstraktion oder Komposition aus einfachen Kontextdaten höherwertige Kontextinformationen.

Die Kommunikation mit anderen Kontextservern erfolgt über HTTP, die Anfragen und Antworten sind in XML spezifiziert. Um dieses realisieren zu können, beinhaltet jeder Kontextserver einen HTTP-Server sowie einen XML-Parser.

Die Vorteile dieses Systems sind die Skalierbarkeit durch die Verwendung von verteilten Kontextservern und die Verwendung von anerkannten Standards für den Austausch von Nachrichten. Die Archivierung der Kontextdaten erfolgt lokal bei den einzelnen Kontextservern, da eine zentrale Komponente fehlt. Dies erfordert größere Kapazitäten an Ressourcen für die einzelnen Stationen. Kontextinformationen verschiedener Stationen in Bezug zu setzen, ist durch diesen ausschließlich verteilten Ansatz sehr aufwendig, da die einzelnen Informationen für die Verarbeitung von allen Kontextservern abgefragt werden müssen.

3.3.3.5 CALAIS

CALAIS (Context And Location Aware Information Service) ist ein weiteres System zur Unterstützung kontextabhängiger Anwendungen. Es wurde von Giles John Nelson am Clare College entwickelt [Nel98]. Ziele dieser Arbeit waren

- die Untersuchung von Sensor-Technologien, um Orts- und Kontextinformationen zu gewinnen,
- die Bereitstellung von Mechanismen zur Verteilung der Informationen in einem verteilten System,
- das Management von Ortsinformationen sowie
- die Unterstützung der Anwendungen durch eine Abstraktionsschicht, welche die Details der Sensortechnologien verbirgt.

Als Sensor-Technologien werden Active Badges und Ultrasound Badges eingesetzt. Weitere Kontextinformationen werden über einen Workstation Activity Monitor, einen Door Status Service, einem Telephone Handset und einem Motion Detection System ermittelt.

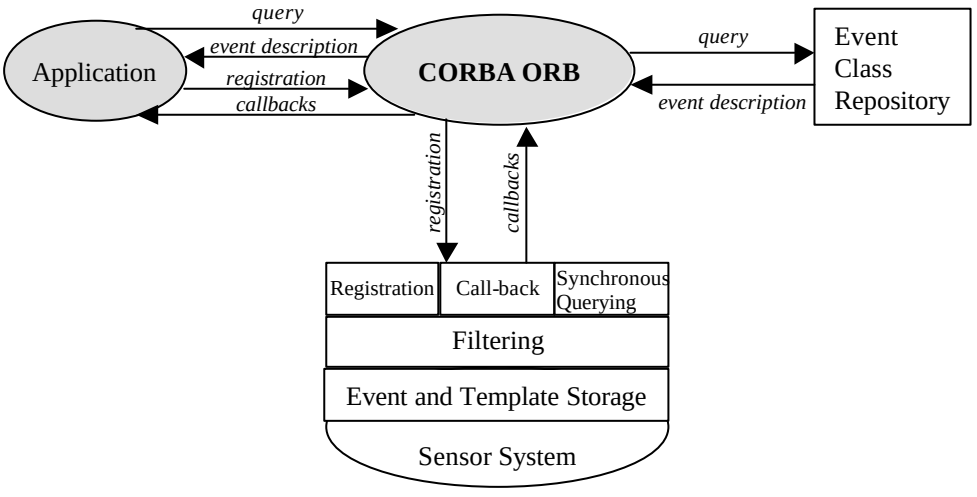

Abbildung 7 CALAIS Event Service

Die Verteilung der Kontextdaten erfolgt über Event-Mechanismen (siehe Abbildung 7). Dabei setzt das CALAIS-System auf den CORBA Standard (Common

Object Request Broker Architecture) der OMG (Object Management Group), da dieser ein standardisiertes Interface, die OMG interface definition language (IDL), sowie Ortstransparenz bietet. Jeder Sensor wird um ein in IDL spezifiziertes CORBA-Interface erweitert, welches die Attribute und die Event-Klassen beschreibt. Die eigentlichen Details des Sensors bzw. der Sensordaten werden dadurch verborgen. Das CALAIS-System unterstützt so die verteilte Erfassung von Kontextinformationen und stellt Abfrage- und Benachrichtigungsmechanismen bereit.

Durch die Verwendung von CORBA als Middleware ist vielfältige Funktionalität geboten, doch ist es nicht für Embedded Systems konstruiert. Der zentralistische Ansatz schränkt die Skalierbarkeit des Systems ein. Das Hinzufügen von neuen Sensoren wird nicht unterstützt. CALAIS bietet keine Mechanismen zur Speicherung oder der Interpretation von Kontextinformationen, sondern erleichtert ausschließlich den Zugriff auf Sensordaten.

3.3.3.6 CoolTown

CoolTown ist ein Forschungsprojekt der Hewlett-Packard Laboratories in Palo Alto. Es basiert auf fünf Glaubensgrundsätzen [Cooltown]:

1. Die Vielfalt von eingebetteten und mobilen Geräten wird zunehmen.
2. Das World Wide Web wird das Netz der Zukunft sein.
3. Jedes Ding wird seine Web-Präsenz besitzen und so spontane und personalisierte Dienste ermöglichen.
4. Durch Annäherung der physischen und der virtuellen Welt können die Vorteile der Web-Dienste die reale Welt verbessern.
5. Die Ökosysteme der Service Provider werden auf kreative und produktive Art zusammenwachsen.

CoolTown ist eine Infrastruktur, welche die Realisierung kontextabhängiger Anwendungen und Dienste für den Endbenutzer durch die Repräsentation jedes realen Objekts (beispielsweise Personen, Orte oder Geräte) durch eine Web-Seite unterstützt. Jede Web-Seite aktualisiert sich dynamisch, indem sie Informationen über das jeweils repräsentierte Objekt sammelt.

Durch die Verwendung des WWW als Plattform, kommen Standards wie URLs oder HTTP zum Einsatz. Eine weite Verbreitung ist garantiert, da für die gängigen Hard- und Software-Plattformen Implementierungen verfügbar sind, und grundlegende Dienste wie Sicherheit und Abrechnungsverfahren für das WWW existieren.

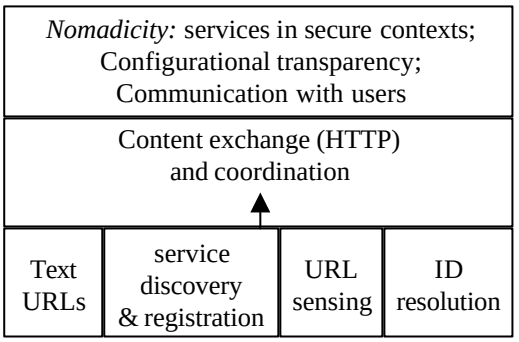

Abbildung 8 CoolTown Infrastruktur

Die Infrastruktur von CoolTown ist in Schichten aufgebaut (siehe Abbildung 8) [KBM+01]. Die unterste Schicht stellt Dienste zum Auffinden und Ermitteln der Web-Präsenzen von Personen, Dingen und Orten bereit. Eine Abstraktion der Sensordaten erfolgt über die URLs sowie durch die Web-Seiten. Das Auffinden von Diensten erfolgt über einen bereitgestellten Discovery-Mechanismus. Die mittlere Schicht wird durch das Web selbst realisiert und dient dem Austausch von Informationen über HTTP. Die oberste Schicht realisiert die Dienste zur Unterstützung der Web-Repräsentationen von Orten und von mobilen Personen.

Die Vorteile dieses speziellen Systems liegen in der Verwendung von weit verbreiteten Standards. Jedoch ist das System auf eine bestimmte Klasse von kontext-abhängigen Anwendungen spezialisiert. Es wird keine Unterstützung von einfachen Sensordaten angeboten.

3.3.3.7 Context Toolkit

Der umfassendste Ansatz einer unterstützenden Architektur für kontextabhängige Anwendungen ist das Context Toolkit, das am Georgia Institute of Technology von Daniel Salber, Anind K. Dey und Gregory D. Abowd entwickelt wurde [SDA99, DA00b, Dey00]. Das Hauptziel dieses Projektes ist die Bereitstellung eines Frameworks, das die Entwicklung und den Einsatz von kontextabhängigen Anwendungen unterstützt.

Die Architektur des Context Toolkit besteht aus drei Arten von Komponenten: den Context Widgets, den Context Interpreters und den Context Aggregators (vgl. Abbildung 9). Context Widgets sind verantwortlich für das Sammeln von Informationen über die Umgebung durch beliebige Software- und Hardware-Sensoren. Sie vermitteln zwischen den kontextabhängigen Anwendungen und der Umgebung. Ein Context Widget kapselt die „rohen“, unverarbeiteten Kontextdaten durch die Bereitstellung einer geeigneten Schnittstelle, so dass eine Anwendung diese Kontextdaten verwenden kann, ohne über Detailkenntnisse der darunter liegenden Sensor-Technologien verfügen zu müssen. Ein Context Widget wird nicht exklusiv einer Anwendung zugeordnet, sondern zwischen den verschiedenen Anwendungen geteilt.

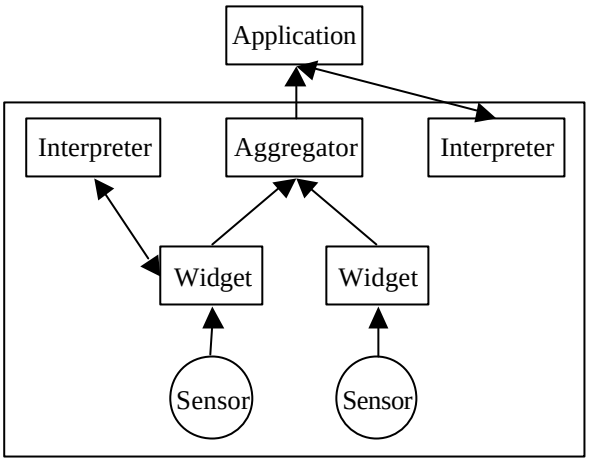

Abbildung 9 Context Toolkit

Context Interpreter verarbeiten rohe oder einfache Sensordaten zu komplexeren Kontextinformationen. Ein Beispiel dafür ist die Umrechnung von Raum-Lokationsdaten zu Gebäude-Lokationsdaten. Die Context Aggregation dient der Verbindung verschiedener Kontextinformationen einer Entität zu Regelbedingungen wie beispielsweise: „wenn sich der Mann in der Küche befindet und Essen kocht, dann ist die Frau zufrieden“.

Mit diesen Abstraktionsebenen - Widgets, Interpreters und Aggregator - erreicht das Context Toolkit die Kapselung der Sensordaten, erleichtert den Zugriff auf die Kontextdaten und stellt verschiedene Mechanismen zur Verarbeitung der Sensordaten für beliebige kontextabhängige Anwendungen bereit. Es unterstützt auch die Speicherung von Kontextdaten für spätere Auswertungen. Nachteil dieses System ist jedoch die fehlende Skalierbarkeit. Durch die strenge Zuordnung eines Widgets zu einem Sensor wird es bei hoher Sensorzahl problematisch. Jedoch ist hervorzuheben, dass zur Gewährleistung des Datenschutzes als Sicherheitsmechanismus eine Zugriffskontrolle für die gespeicherten Daten eingebaut wurde, ein Sicherungsmechanismus, der in den zuvor vorgestellten Systemen nicht vorhanden ist.

3.3.3.8 MUSE

MUSE ist ein relativ junges Projekt am UCLA Department of Computer Science MUSE]. Ziel von MUSE ist die Bereitstellung einer Middleware zur Unterstützung der Entwicklung von kontextanhängigen Anwendungen und Diensten für eine mit Sensoren bestückte Umgebung.

Die MUSE Infrastruktur verwendet Jini [Jini01] als Service-Plattform. Sie besteht aus Hardware- und Software-Komponenten, die sich als Jini-Services im Netz anmelden, sobald sie eingefügt werden. Es werden drei Dienst-Arten unterschieden: der Lookup-Service, die Sensor-Services und die Fusion-Services.

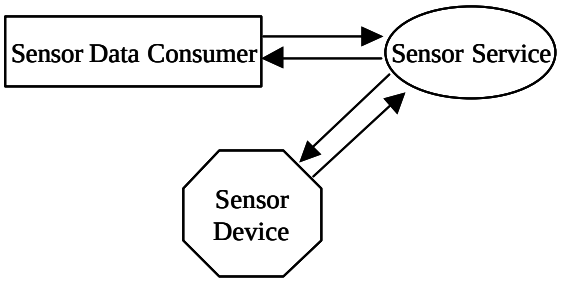

Abbildung 10 Sensor-Service in MUSE

Der Lookup-Service registriert die angebotenen Dienste. Ein Sensor-Service repräsentiert einen Sensor und kann über das Netz abgefragt werden (siehe Abbildung 10). Der Fusion-Service sammelt Daten von Sensoren und interpretiert diese. Er stellt somit den Anwendungen höherwertige Kontextinformationen zur Verfügung.

Diese Architektur unterstützt weder die Abstraktion noch die Aggregation von Sensordaten. Auch erfolgt keine Speicherung für eine mögliche spätere Auswertung oder Weiterverarbeitung. Durch eine Eins-zu-eins-Zuordnung von Sensor zu Sensor-Service ist – wie schon im Context Toolkit – die Skalierbarkeit nicht gegeben.

Kapitel 4

Zugriffskontrolle

In diesem Kapitel werden einige grundlegende Begriffe aus dem Bereich Zugriffskontrolle eingeführt. In Abschnitt 4.1 wird der Begriff „Sicherheitspolitik“ erläutert und in Abschnitt 4.2 ein Überblick über die wichtigsten Sicherheitsmodelle gegeben. Anschließend werden in Abschnitt 4.3 die relevanten Arbeiten präsentiert und bewertet, die erste Ansätze in Richtung einer kontextabhängigen Zugriffskontrolle darstellen.

4.1 Sicherheitspolitik

Eine der ersten Definitionen für den Begriff „Security Policy“ oder deutsch „Sicherheitspolitik” stammt von J. A. Goguen und J. Meseguer aus dem Jahre 1982. Für sie legt eine Sicherheitspolitik „… the security requirements for a given system” fest [GM82]. In den unter dem Namen Orange Book bekannten „Trusted Computer System Evaluation Criteria“ (TCSEC) [DoD85] wird eine Sicherheitspolitik als eine „Sammlung von Gesetzen, Regeln und Methoden, die festlegen, wie eine Organisation sensitive Informationen verwalten, schützen und zu verteilen hat“ und eine Zugriffspolitik als eine „Sammlung von Regeln, die vom System verwendet werden, um zu entscheiden, ob der Zugriff eines bestimmten Subjekts auf ein spezielles Objekt erlaubt werden kann“.

Wie diese Definitionen schon zeigen, werden Sicherheitspolitiken auf verschiedenen Ebenen definiert (siehe Abbildung 11):

- auf Organisationsebene,
- auf Systemebene und
- auf technischer Ebene.

Auf Organisationsebene werden unter einer Sicherheitspolitik allgemein die Festlegung der strategischen Sicherheitsziele einer Organisation und deren Erfüllung verstanden. Eine Sicherheitspolitik auf Systemebene legt die Regeln und Vorschriften für die Verarbeitung sensitiver Informationen für ein konkretes System fest. Auf

technischer Ebene ist zu spezifizieren, wie Hard- und Software die Ressourcen des Systems nutzen dürfen.

Ein wesentlicher Teil der Sicherheitspolitik auf technischer Ebene stellt die Sicherheitspolitik zur Autorisierung von Zugriffsoperationen oder kurz die Zugriffskontrollpolitik dar. Diese spezielle Art einer Sicherheitspolitik wird in den folgenden Abschnitten genauer betrachtet. In der Literatur wird anstelle des Begriffs „Zugriffskontrollpolitik“ oftmals einfach auch der Oberbegriff „Sicherheitspolitik“ verwendet.

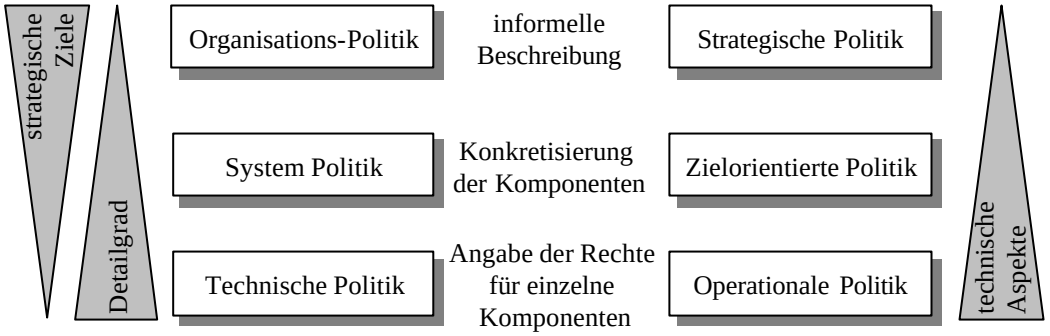

Abbildung 11 Sicherheitspolitik

Sicherheitspolitiken sind Bestandteil einer Sicherheitsarchitektur. Sie formulieren Schutzanforderungen für die Objekte und benötigen für ihre Durchsetzung konkrete Sicherheitsmechanismen. Für eine Sicherheitspolitik zur Autorisierung von Zugriffsoperationen wird die Einhaltung des Referenzmonitorprinzips erwartet [DoD85]. Dieses besagt, dass jeder Zugriff auf ein Objekt kontrolliert werden muss und keine Möglichkeit zur Umgehung dieses Mechanismus existieren darf.

Während eine Sicherheitspolitik auf Organisationsebene (engl. corporate security policy) überwiegend informell beschrieben wird, was aufgrund der Komplexität nicht anders möglich ist, werden zur Festlegung von Sicherheitspolitiken auf der System- und der technischen Ebene (engl. system security policy und technical security policy) weitestgehend formale Methoden eingesetzt. Um Sicherheit für ein IT-System erreichen zu können, müssen neben der Formulierung einer Politik, deren automatische Überprüfung und Durchsetzung erreicht werden. Dies ist nur möglich durch Abstraktion des zu betrachtenden IT-Systems in einem Sicherheitsmodell und dessen formale Beschreibung in einer Sicherheitspolitiksprache (engl. security policy language) anhand derer eine Validierung der Sicherheitspolitik möglich ist.

Da das Ziel bei der Erstellung eines Sicherheitsmodells die formale Überprüfung der Sicherheitspolitik ist, werden das Sicherheitsmodell wie auch die verwendeten Mittel zur Formalisierung auf die Ideen und Belange der Politik zugeschnitten sein. Das zeigt sich deutlich in der Vielfalt der Modelle. Die bekanntesten sind dabei Bell/LaPadula [BL73], Clark/Wilson [CW87], RBAC [FK92] und Chinese-Wall [BN89].

Die Zugriffskontrolle ist ein wesentlicher Bestandteil der Systemsicherheit. Unter Zugriffskontrolle werden alle Maßnahmen, Methoden und Einrichtungen verstanden, die den Zugriff auf eine Ressource auf Autorisierung überprüft und diesen gestattet

oder zurückweist. Eine Zugriffskontrolle basiert auf einem Zugriffskontrollmodell (engl. access control model). Ein Zugriffskontrollmodell ist die formale Beschreibung von Sicherheitseigenschaften, im Falle der Zugriffskontrolle von Vertraulichkeit und Integrität. Um ein Zugriffskontrollmodell realisieren zu können, ist die Spezifikation in einer Zugriffskontrollpolitik-Sprache notwendig. Diese dient dem Zugriffskontroll-mechanismus als Grundlage für die Zugriffsentscheidung.

So vielfältig die im Laufe der Zeit entwickelten Sicherheitsmodelle sind, so vielfältig sind auch die hierfür bereitgestellten Beschreibungssprachen. Da eines der Ziele dieser Arbeit die Spezifikation von Bedingungen für die Autorisierung von Zugriffsoperationen abhängig vom Kontext des Zugriffs, jedoch unabhängig von dem darunter liegenden Sicherheitsmodell ist, werden im Folgenden einige gängigen Sicherheitsmodelle und Politiksprachen vorgestellt. Die Auswahl ist bei weitem nicht vollständig. Es wurde eine Auswahl der Ansätze getroffen, die zum Verständnis dieser Arbeit von Relevanz sind. Eine umfassendere Auflistung ist beispielsweise in [Eck00, Fis91, Opp97, Sch99, SS94] zu finden.

4.2 Sicherheitsmodelle

Sicherheit ist eine Eigenschaft, die empirisch nicht verifizierbar ist. Man möchte aber die Möglichkeit haben, bestimmte Sicherheitseigenschaften soweit wie möglich formal zu beweisen. Mithilfe eines Zugriffskontrollmodells - oder als Oberbegriff Sicherheitsmodell - werden die Prinzipien, nach denen die Entscheidung über die Zulässigkeit eines Zugriffs getroffen wird, beschrieben. Sie dient somit als Grundlage zur Entwicklung von sicheren IT-Systemen.

Ein Sicherheitsmodell besteht aus einer Menge von Subjekten, den aktiven Komponenten, einer Menge von Objekten, den passiven Komponenten, einer Menge von Operationen auf diesen Objekten und aus Zuständen, die in „sichere“ und „unsichere“ unterteilt werden. Ausgehend von einem sicheren Initialzustand darf das System durch die Durchführung von erlaubten Operationen auf Objekten die Menge der sicheren Zustände nicht verlassen.

Sicherheitsmodelle können nach verschiedenen Kriterien klassifiziert werden. Klassische Modelle sind die benutzerbestimmbare oder wahlfreie Zugriffskontrolle (engl. discretionary access control; DAC) und die systembestimmte Zugriffskontrolle (engl. mandatory access control; MAC). Weitere Unterscheidungsmöglichkeiten sind zwischen Zustandsmodellen, die den Aspekt der Datensicherheit modellieren, und Informationsflussmodellen [Gas88], oder zwischen zustandsbehafteten und zustands-losen bzw. statischen und dynamischen Modellen [Opp97]. Eine ausführliche Be-schreibung von Klassifikationsmöglichkeiten für Sicherheitsmodelle ist in [Eck96] zu finden.

4.2.1 Discretionary Access Control

Bei einer wahlfreien Zugriffskontrolle wird durch Administrationsrechte festgelegt, wer welche Zugriffe autorisieren kann. Der bekannteste Vertreter einer wahlfreien Zugriffskontrolle ist die eigentümerbasierte Rechtevergabe. Diese erlaubt den Be-

nutzern nach ihrem eigenen Ermessen Zugriffsrechte auf Objekte unter ihrer Kontrolle an andere Benutzer des Systems weiterzugeben oder diese wieder zu entziehen [TCSEC85]. Die Vorstellung, die dahinter steckt ist, dass ein Benutzer der „Eigentümer“ des Objekts ist und daher über die vollständige Kontrolle darüber verfügen sollte. Die Eigentumsrechte erlangt er im Allgemeinen bei der Erzeugung des Objekts [SCF+96].

Die Idee des DAC-Modells wurde erstmals von Butler W. Lampson im Jahre 1971 in [Lam71] vorgestellt. Sein Vorschlag ist, das Problem der Rechtezuordnung als eine große Zugriffsmatrix zu betrachten, der „Lampson Access Matrix“, wobei die Spalten den Objekten *O* und die Reihen den Subjekten *S* des Systems zuzuordnen sind (Abbildung 12). Legt die Menge *R* die möglichen Zugriffsrechte fest, so definiert die Matrix M_t: $|S| \times |O| \rightarrow 2^R$ die Rechte der Subjekte an den Objekten zum Zeitpunkt *t*. Im Beispiel der Abbildung 12 hat das Subjekt „editor” Lese- und Schreibrechte auf der Datei „.profile”, das Subjekt „netscape” jedoch keine.

	Objekt 1 ~/.profile	*Objekt 2* ~/.netscape	*Objekt 3* editor	*Objekt 2* netscape
Subjekt 1: editor	*read,write*	-	*kill*	-
Subjekt 2: netscape	-	*read,write*	*kill*	*kill*

Abbildung 12 Beispiel für Lampson Access Matrix

Viele Sicherheitskonzepte heutiger Systeme basieren auf dem Zugriffsmatrix-Modell. Dabei ist die Zugriffsmatrix in den meisten Systemen nicht statisch, sondern kann zur Laufzeit modifiziert werden, beispielsweise durch den Eigentümer eines Objekts, der nach belieben Rechte an andere Benutzer erteilen oder entziehen kann.

Da eine Zugriffsmatrix sehr komplex und unhandlich werden kann, wird bei einer Implementierung die Matrix meist in Spalten oder Zeilen zerlegt. Die spaltenweise Zerlegung ergibt Zugriffskontrolllisten (engl. Access Control Lists; ACLs), eine zeilenweise Zerlegung die so genannten Fähigkeitslisten (engl. Capability Lists).

Zugriffskontrolllisten speichern zu jedem Objekt, welche Subjekte darauf zugreifen können. Es ist daher einfach zu ermitteln, welche Subjekte Zugriff auf ein bestimmtes Objekt haben. Schwierig ist jedoch festzustellen, auf welche Objekte ein Subjekt welche Rechte hat und damit verbunden auch das Problem, einen Benutzer aus dem System zu entfernen, da zur Lösung dieses Problems jedes Objekt kontaktiert werden muss. Der Vorteil von ACLs liegt jedoch in der einfachen Verwaltung der Zugriffsrechte sowie deren einfache und effiziente Implementierung für kleinere Systeme. Mit steigender Komplexität des Systems verschlechtert sich jedoch die Handhabung [FGL93].

Während eine ACL die Rechte an Objekte bindet, werden bei Fähigkeitslisten die Rechte an die Subjekte gebunden. Fähigkeitslisten speichern zu jedem Subjekt, auf welche Objekte welche Zugriffe erlaubt sind. Es ist daher leicht festzustellen, auf welche Objekte ein Subjekt welche Zugriffsrechte hat, schwierig ist aber die Frage,

welche Subjekte auf ein Objekt Zugriff haben. Das Entfernen von Objekten aus dem System erfordert somit die Benachrichtigung aller Subjekte.

In objektorientierten Systemen werden Objektreferenzen oftmals als Capabilities betrachtet: bei der Erzeugung eines Objekts erhält das erzeugende Subjekt automatisch die Objektreferenz, mit der es auf das Objekt Zugriff hat. Seine Rechte am Objekt kann es durch die Weitergabe der Objektreferenz an andere Benutzer weiterreichen.

Die Discretionary Access Control, die zu den ältesten Zugriffsstrategien zählt, hat eine entscheidende Schwäche: Informationen können von einem Objekt zu einem anderen Objekt kopiert werden, das sich nicht im Besitz des Eigentümers des Originals befindet. Eine Kontrolle des Informationsflusses ist somit nicht möglich. Diese Schwäche kann von Trojanischen Pferden genutzt werden, um sensitive Informationen an Unberechtigte weiterzuleiten.

4.2.2 Mandatory Access Control

Vor dem Hintergrund, dass das Problem der Trojaner rapide stieg, wurde das Modell der Mandatory Access Control (MAC) entwickelt, das die Vertraulichkeit von Daten gewährleisten soll [Den76]. Bei einer Mandatory Access Control oder deutsch systembestimmte Zugriffskontrolle werden die Zugriffsrechte von einer dedizierten Instanz (beispielsweise dem Sicherheitsadministrator oder dem Betriebssystem) durch strikte Regeln als Sicherheitsattribute den Objekten zugewiesen.

Ein früher Vertreter dieses Konzepts ist das formale, hierarchische Bell-LaPadula Modell, das zwischen 1973 und 1976 von D. Elliott Bell und Leonard J. LaPadula zum Schutz der Vertraulichkeit entwickelt wurde [BL73, BL75]. Zur Gewährleistung der Vertraulichkeit und der Kontrolle des Informationsflusses wird eine Menge von Sicherheitsklassen festgelegt. Dabei wird jedem Subjekt eine Sicherheitsklasse, auch Clearance genannt, zugeordnet und jedem Objekt eine Sicherheitsklassifikation.

Der Zugriff auf Objekte wird im Bell-LaPadula-Modell durch zwei Regeln beschränkt: der Simple-Security und der *-Eigenschaft (vgl. Abbildung 13). Die Simple-Security-Regel, auch no-read-up-Regel genannt, legt fest, dass ein Lesezugriff eines Subjekts *s* auf ein Objekt *o* nur dann erlaubt ist, wenn *s* das entsprechende Zugriffsrecht besitzt und die Sicherheitsklassifikation des Objekts kleiner oder gleich der Sicherheitsklasse des Subjekts ist. So darf in Beispiel der Abbildung 13 das Subjekt *Müller* mit der Sicherheitsklasse *B* prinzipiell nur auf Objekte der Sicherheits-klassifikation *vertraulich* und *unklassifiziert* lesend zugreifen.

Die *-Eigenschaft oder no-write-down-Regel erlaubt eine Schreiboperation auf einem Objekt nur dann, wenn die Sicherheitsklassifikation s_O des Objekts gleich der Sicherheitsklasse s_S des Subjekts, eine append-Operation (dt. anfügen), wenn s_S kleiner oder gleich s_O ist. Diese beiden systembestimmten Regeln, die no-read-up- und die no-write-down-Regel, gewährleisten, dass nur Informationsflüsse von unten (geringste Sicherheitsstufe) nach oben (höchste Sicherheitsstufe) entlang der partiellen Ordnung der Sicherheitsklassen erfolgen kann.

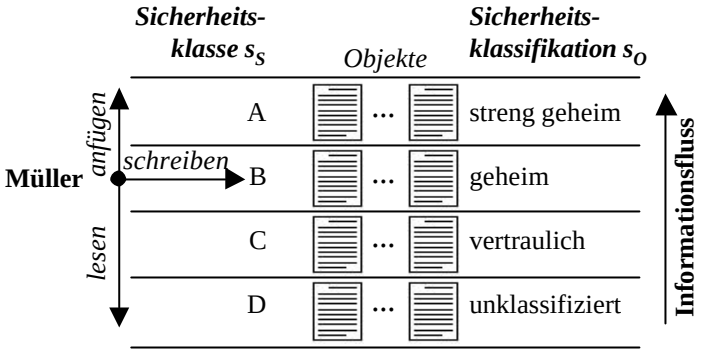

Abbildung 13 Read-up- und write-down-Regel im Bell-LaPadula-Modell

Systembestimmte Zugriffskontrollmodelle wurden ursprünglich für das militärische Umfeld entwickelt, um den hohen Anforderungen an Vertraulichkeit zu genügen. Diese Modelle fanden jedoch durch ihre geringe Benutzerfreundlichkeit im nicht-militärischen Bereich kaum Beachtung. Es gibt jedoch auch Versuche, dieses Prinzip in kommerziellen Systemen einzusetzen [CW87, DG97]. David D. Clark und David R. Wilson stellten 1987 ein Modell vor, das – im Gegensatz zum Bell-LaPadula-Modell – nicht für militärische sondern für kommerzielle Sicherheitsanforderungen ausgelegt war. Das semiformale, nicht-hierarchische Clark-Wilson-Modell dient dem Schutz der Integrität von Daten [CW87].

4.2.3 Role-based Access Control

Das Konzept der Role-based Access Control (RBAC), oder deutsch des rollenbasierten Zugriffsmodells, stammt aus dem Jahre 1992 und wurde von David. F. Ferraiolo und Richald Kuhn entwickelt [FK92]. Weiterentwicklungen stammen beispielsweise von Ferraiolo, Cugini und Kuhn [FCK95] sowie Sandhu [SCF+96].

Das Ziel des RBAC-Modells ist der Schutz der Integrität der Daten. Entwickelt für den Einsatz im Geschäftsbereich, stellte sich die Frage „wer darf welche Daten lesen“. Wie die Erfahrung lehrte, hing die Beantwortung diese Frage in Organisationen meist von der Stellung – sprich der Rolle – des Benutzers innerhalb der Organisation ab. Die Vergabe von Rechten durch die Sicherheitsadministration basiert somit auf einer geeigneten Festlegung von vordefinierten Rollen in der Organisation und den damit verbundenen Rechten. Ein einzelner Benutzer erhält Rechte durch die statische oder dynamische Zuordnung von Rollen. Der Entzug von Rechten erfolgt durch das Entfernen von Rollen.

Die Basis des RBAC-Modells bilden – wie der Name auch sagt – die Rollen. Rollen dienen der Gruppierung und der Kategorisierung von Personen. Eine Rolle ist somit eine Menge von Transaktionen, die die Mitglieder dieser Gruppe (indirekt) ausführen dürfen. Die Mitglieder der Rollen sind die Benutzer des Systems, also Personen. Ein Benutzer kann Mitglied einer oder mehrerer Rollen sein. Ein aktiver Prozess, welcher im Namen eines Benutzers Transaktionen initiiert, wird Subjekt genannt. Eine Transaktion ist eine Handlung oder Handlungsfolge, die auf ein oder mehrere Objekte angewendet wird. Transaktionen sind somit sowohl Rollen wie auch Objekten zugeordnet. Objekte sind beliebige Ressourcen eines IT-Systems.

Rollen können hierarchisch organisiert werden. Wenn ein Benutzer Mitglied einer Rolle r_i ist, welche weitere Rollen enthält, so darf das den Benutzer repräsentierende Subjekt zusätzlich zu den Transaktionen der Rolle r_i auch die Transaktionen der enthaltenen Rollen ausführen.

Um eine Transaktion durchführen zu können, muss das Subjekt in einer Rolle aktiv sein, die die Rechte an dieser Transaktion enthält. Um eine Rolle aktivieren zu können, muss das Subjekt bzw. der Benutzer für diese Rolle autorisiert sein. Ein Benutzer kann gleichzeitig in einer beliebigen Anzahl für ihn autorisierter Rollen aktiv sein, sofern diese nicht durch Regeln zur Aufgabentrennung (engl. separation of duty) gegenseitig ausgeschlossen werden.

4.2.4 Chinese Wall-Modell

Ein Beispiel für eine zustandsbehaftete Zugriffskontrolle ist das Chinese-Wall-Modell von Brewer und Nash [BN89], eine Kombination eines wahlfreien und systembestimmten Zugriffskontrollemodells. Dieses Modell wurde entwickelt, um den Anforderungen an Vertraulichkeit der Daten und der Kontrolle von Informationsflüssen bei Börsen- oder Banktransaktionen zu genügen.

Ein Berater, welcher über Insiderwissen einer Firma verfügt, sollte keine konkurrierenden Firmen unter Ausnutzung seiner Insiderkenntnisse beraten und dadurch in einen Interessenkonflikt geraten. Das Chinese-Wall-Modell gewährt einem Benutzer den Zugriff auf interne Informationen nur dann, wenn in der Vergangenheit kein Zugriff auf Informationen konkurrierender Kunden erfolgte. Das Modell ist eine Erweiterung des Zugriffsmatrix-Modells. Die Objekte werden in eine hierarchische Struktur eingeteilt, auf dessen oberster Stufe die Konfliktklassen stehen.

4.3 Relevante Arbeiten im Bereich kontextabhängiger Zugriffskontrolle

Das Konzept einer kontextabhängigen Zugriffskontrolle ist nicht neu. Es existieren verschiedene Modelle bzw. Implementierungen, welche in irgendeiner Weise den Zugriffskontext in die Entscheidungsfindung der Autorisierung mit einbeziehen. Die bekanntesten Kontextinformationen, welche als Sicherheitsattribute verwendet werden, sind neben der Identität des Benutzers und einer Gruppen- oder Rollenzugehörigkeit, die Zeit und die Lokation [CCP11-98]. Bekanntestes Beispiel für eine zeitabhängige Zugriffskontrolle sind Firewalls und für eine ortsabhängige Zugriffskontrolle Web-Server.

Komplexere kontextabhängige Zugriffskontrollmodelle sind beispielsweise das regionorientierte Modell, das zustandsorientierte Modell, das Generalisierte RBAC-Modell, das Inhalts-basierte Modell oder das Historie-basierte Modell. Jeder dieser Ansätze verwendet eine oder (sehr) wenige Arten von Kontextinformationen wie Zeit, Ort oder Systemzustand oder ist auf ein bestimmtes Anwendungsszenario abgestimmt. Keines der oben genannten und in den folgenden Abschnitten detaillierter vorgestellten Zugriffsmodelle ist flexibel genug, beliebige Typen von Kontextinfor-

mationen einzubeziehen, wie es in dieser Arbeit verlangt wird, oder besitzt eine auf die Ziel-Anwendung abgestimmte Implementierung für den umsetzenden Zugriffskontrollmechanismus.

4.3.1 Regionorientierte Zugriffskontrolle

Albert Held stellt in seiner Dissertation [Hel98] ein Konzept für eine ortsabhängige Zugriffskontrolle in verteilten mobilen Systemen vor. Dieses Konzept trägt besonders den Anforderungen eines mobilen Benutzers oder Terminals Rechnung. Dazu zählen beispielsweise geringe Bandbreite, unzuverlässige Netzanbindung, Ad-hoc Netze ohne Anbindung an eine Netzinfrastruktur, aber auch unterschiedliche Konfigurationen und Topologien der Netze des jeweiligen Aufenthaltsorts. Zur Lösung von Sicherheitsproblemen wie Zugriffskontrolle und unterschiedliche Sicherheitspolitiken, die sich ergeben, wenn der Benutzer bzw. das Terminal sich außerhalb des Firmennetzes und innerhalb eines fremden Netzes befindet, wird ein Verfahren vorgestellt, das eine Steuerung der Zugriffsmöglichkeiten abhängig vom jeweiligen Aufenthaltsort realisiert.

Der Ansatz geht von einer von Subjekten und Objekten unabhängigen Zugriffsstruktur aus. Diese Struktur ist als azyklischer gerichteter Graph abgelegt (vgl. Abbildung 14). Die Knoten des Graphen repräsentieren so genannte *Regionen*, die für physische oder logische Lokationen stehen. Die Rechte sind an *Regionen* gebunden. Subjekte und Objekte erhalten Rechte durch die Zuordnung zu einer Region, ähnlich dem Rollenkonzept in RBAC. Wird der Aufenthaltsort des mobilen Geräts geändert, wird auch die Zugriffsstruktur durch das Hinzufügen oder Entfernen von Kanten zwischen Teilgraphen geändert. Eine Änderung auf Subjekt- oder Objektebene ist nicht notwendig.

Systemmodell

Die Grundlage des ortsabhängigen Zugriffskontrollmodells bildet die Abstraktion eines realen verteilten Netzes: Alle Rechner – egal ob Server oder Endgerät, ob drahtgebunden oder mobil - werden als *Stationen* modelliert, welche durch eine Anzahl von Attributen charakterisiert werden. Die Stationen werden verschiedenen, disjunkten Bereichen, so genannten *Domains*, zugeordnet. Jede Domäne verfügt über einen Domain Manager, der die Informationen über den Zustand dieser Domäne und den dazugehörigen Stationen verwaltet. Eine Station befindet sich zu einem Zeitpunkt entweder in genau einer Domäne oder ist vom System abgekoppelt. Dabei wird zwischen der Home-Domain, der Foreign-Domain und der Current-Domain unterschieden.

Das Regionenmodell

Das Regionenmodell besteht aus *Subjekten*, *Objekten* und dem *Regiongraph* (siehe Abbildung 14). Subjekte sind Benutzer des Systems und Objekte beliebige Ressourcen, die einer Zugriffskontrolle unterliegen. Der Regiongraph ist ein gerichteter azyklischer Graph, der die von den Subjekten und Objekten unabhängige Zugriffsstruktur beschreibt. Die Knoten repräsentieren die Regionen, die Kanten

Verbindungen (Links) zur Weitergabe von Zugriffsrechten. Ein Pfad im Regiongraph wird *Regionpfad* genannt.

Ein Objekt ist genau einer Region zugeordnet. Somit kann eine Region als Menge von Objekten interpretiert werden. Die *Regionen* sind durch Links verbunden. Zwischen zwei Regionen kann höchstens ein Link existieren. Man unterscheidet zwei Arten von Links: transitive und nicht-transitive. Während transitive Links die Rechte weitergeben, unterbrechen nicht-transitive Links den Regionpfad.

Das Erteilen von Zugriffsrechten erfolgt durch die Zuordnung eines Subjekts zu einer Region. Ein Subjekt kann höchstens an eine Region gebunden werden. Durch das Setzen einer Region erhält der Benutzer die Zugriffsrechte auf alle Objekte, die sich in der gesetzten Region und den Regionen befinden, die über die gesetzten Kanten erreichbar sind, entsprechend der Kennzeichnung der Kanten. Durch die Modifikation des Regiongraphen durch Hinzufügen oder Löschen von Regionen oder Kanten, werden die Zugriffsrechte geändert.

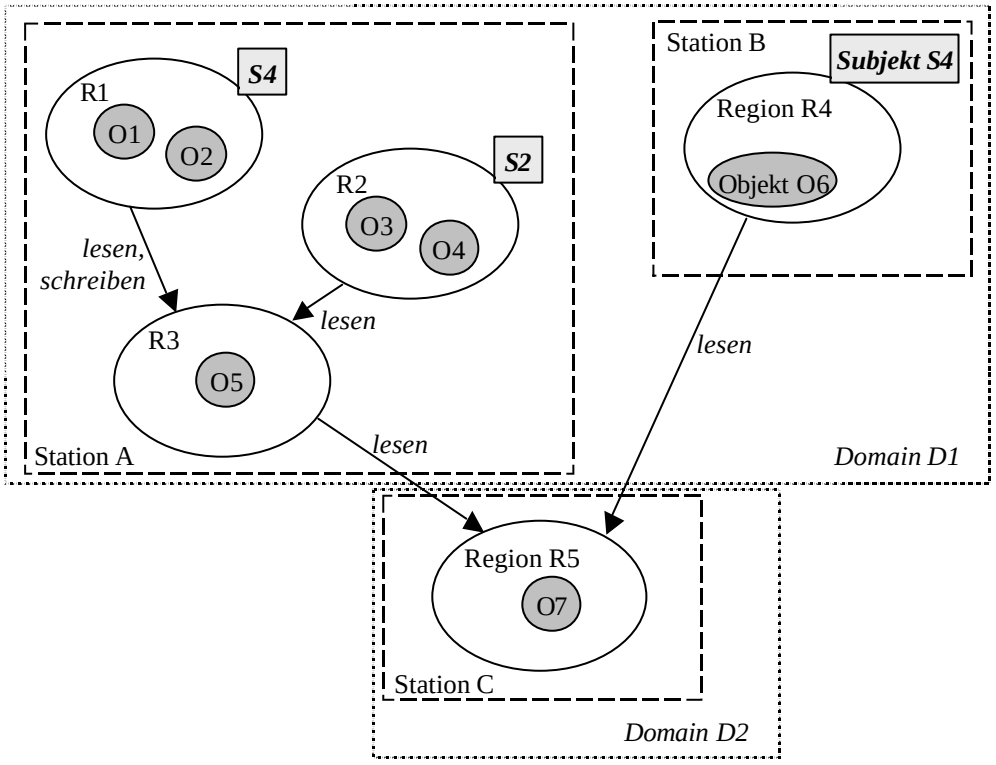

Abbildung 14 Regiongraph mit Zuordnung von Regionen zu Stationen

Die Verteilung der Informationen über die Zugriffsrechte im realen System wird durch die Zuordnung von Teilen des Regiongraphen zu Stationen realisiert. Jede Region ist eindeutig einer Station zugeordnet. Eine Station kann jedoch mehrere Regionen enthalten. Jede Station verwaltet den auf ihr gespeicherten Teil des Regiongraphen, wodurch ein autonomes Arbeiten ohne Verbindung zu anderen Stationen möglich ist.

Verlässt eine Station eine Domäne, werden alle von den Regionen dieser Station ausgehenden Links gelöscht. Bei der Anmeldung in einer neuen Domäne wird der

Graph „neu verlinkt“. Zuvor gelöschte Links sowie neue werden gezogen, abhängig von den nun vorhandenen Bedingungen und den damit verbundenen Rechten. Dazu werden die Informationen benötigt, welche festlegen, von welchen Domänen aus auf welche anderen Domänen zugegriffen werden darf. Jede Domäne verwalten dabei die Teilinformationen, die besagen, von welchen Domänen aus der Zugriff auf die gerade betrachtete erlaubt ist.

Bewertung

Die regionorientierte Zugriffskontrolle stellt einen interessanten Ansatz dar, Ortsinformationen zur Festlegung von Zugriffsrechten auf lokale und entfernte Objekte zu verwenden. Dieser Ansatz ist jedoch auf die Kontextinformation „Lokation“ beschränkt und kann nicht auf beliebige Kontextinformationen übertragen werden, da die zugrunde liegende Lokationsstruktur des Netzes die Basis für die Konstruktion des Regiongraphen, dem Herzstück des Modells, bildet. Der Einsatz des regionorientierten Zugriffsmodells beschränkt sich auf geschlossene Systeme, da die Stationen eines Rechnernetzes bei der Konstruktion des Regiongraphen und damit bei der Festlegung der Zugriffsrechte bekannt sein müssen.

4.3.2 Zustandsabhängige Sicherheitsspezifikation

Ein Ansatz zur Spezifikation und Durchsetzung einer zustandsabhängigen Sicherheitspolitik wurde 1997 von Christian Eckert in [Eck97] vorgestellt. Der Begriff „zustandsabhängig“ bezieht sich dabei auf die aktuelle Position innerhalb einer beliebigen Handlungsabfolge. Zur Durchsetzung der zustandsabhängigen Sicherheitsspezifikation ist die Festlegung von erlaubten Handlungsfolgen für die Objekte des Systems notwendig. Die Handlungsfolgen werden Protokolle genannt, welche durch reguläre Ausdrücke spezifiziert und in endliche, deterministische Automaten umgesetzt werden.

Spezifikation durch reguläre Ausdrücke und endliche Automaten

Ein Informationssystem wird als Menge von Objekten betrachtet, die nach dem Prinzip des „Message Passing“ miteinander kommunizieren. Objekte sind dabei Instanzen von Klassen, in denen ihre strukturellen Eigenschaften und ihr Verhalten spezifiziert sind.

Die zentralen Konstrukte der Sprache sind:

- *Aktivator*: stößt durch Senden einer Nachricht an den Exekutor die Ausführung einer Operation an,
- *Exekutor*: ist der Empfänger der Nachricht und führt die Operation durch,
- *Teilnehmer (engl. participant)*: die Menge der Aktivatoren und Exekutoren

$$\begin{aligned} \text{participants} &\mapsto \text{participant; participants} \\ &\mapsto \text{participant} \\ \text{participant} &\mapsto \text{ident: ident}_{\text{classname}} \end{aligned}$$

- *Aktion (engl. action)*: eine erlaubte Folge von Operationen

action	↦	operation	(Basisoperation)
	↦	action; action	(Sequenz)
	↦	action | action	(Alternative)
	↦	action*	(beliebige Wiederholung)
	↦	(action)	

- *Schritt (engl. step)*: die Zusammenfassung eines Aktivators, eines Exekutors und einer angestoßenen Aktion

step	↦	$\text{ident}_{\text{activator}}$ $\text{ident}_{\text{executor}}$ action

- *Protokoll (engl*. protocol): eine erlaubte Folge von Schritten

protocols	↦	PROTOCOL ident; PARTICIPANTS participants; BEGIN protocol END;	
protocol	↦	step	
	↦	protocol; protocol	(Sequenz)
	↦	protocol | protocol	(Alternative)
	↦	protocol*	(beliebige Wiederholung)

Die Überprüfung eines Schritts auf „erlaubt“ oder „nicht-erlaubt“ ist äquivalent zur Überprüfung, ob ein Wort in einer Sprache enthalten ist oder nicht. Durch die Überführung der Protokolle in endliche deterministische Automaten, wird die Grundlage für die Durchsetzung der Spezifikation gebildet.

Realisierung in einem verteilten System

Während es in einem zentralen System möglich ist, den Automaten in einem zentralen Referenzmonitor zu realisieren, ist die Situation im Falle eines verteilten Systems komplizierter. Die in [Eck97] vorgeschlagene Realisierung verwendet dazu Capabilities. Als (verbrauchbare) Capability wird das Paar (A, q) betrachtet, wobei A den Automaten und q einen Zustand beschreibt. Damit ein Aktivator x eine Aktion op bei einem Exekutor o initiieren darf, muss dieser den Automaten A „besitzen“ und sich im Zustand q befinden.

Um in den Besitz eines Automaten zu gelangen, müssen mehrere Schritte durchgeführt werden. Wie in Abbildung 15 dargestellt, kann ein potentieller Aktivator *x1* einen Automaten anfordern (Schritt 2), nachdem dieser angeboten wurde (Schritt 1). Erhält er den Zuschlag, wird der Automat weitergereicht (Schritt 3). Anschließend ist es ihm möglich, die gewünschte Operation beim Exekutor *o1* zu initiieren (Schritt 4). Nimmt dieser an unter der Bedingung, dass sich der Automat im richtigen Zustand befindet, wird die Operation durchgeführt und der Automat wechselt in den Folgezustand (Schritt 5).

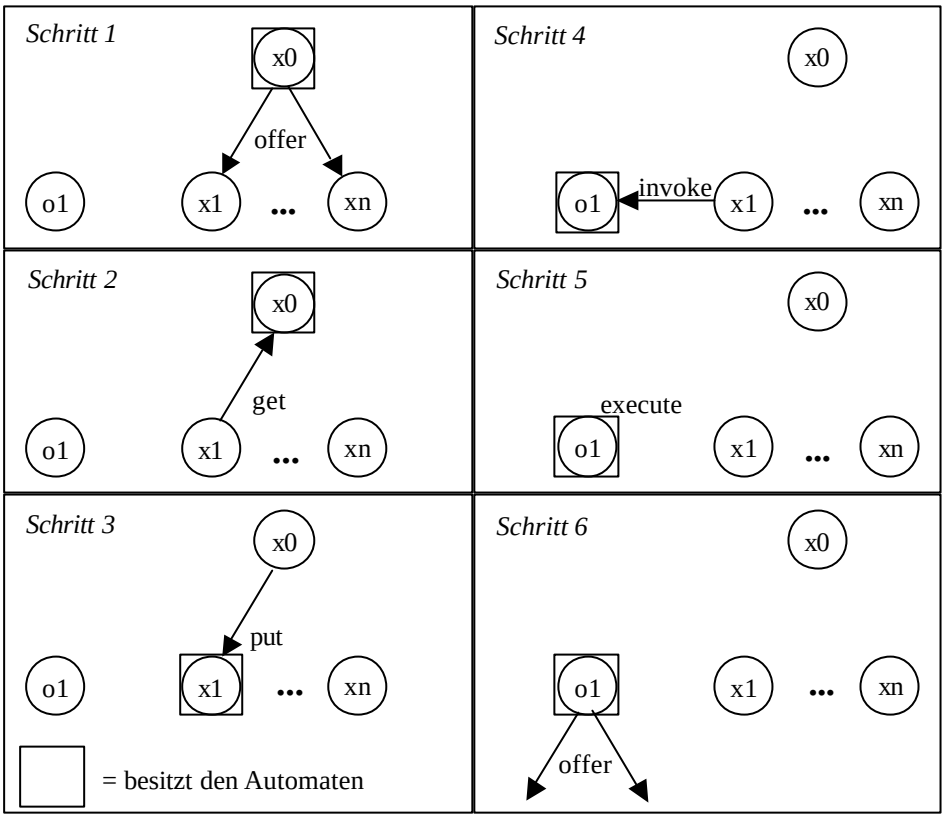

Abbildung 15 Ablauf eines Zustandsübergangs

Sicherheitsbetrachtung

Um Manipulationen an den Capabilities zu verhindern bzw. aufzudecken, werden die ausgetauschten Nachrichten, die den Automaten im aktuellen Zustand enthalten, signiert. So ist es möglich, die Authentizität des Senders sowie die Integrität der Nachricht zu prüfen. Die dazu benötigten öffentlichen Schlüssel aller Teilnehmer werden bei der Initialisierung des Gesamtsystems verteilt.

Es werden die „sign edge"-Variante und die „sign path"-Variante unterschieden. In der „sign edge"-Variante enthält die Capability nur den letzten Streckenabschnitt des durchlaufenen Pfads, die „sign path"-Variante jedoch den vollständigen Pfad. Die „sign edge"-Variante wurde entwickelt, um die Komplexität und den Bedarf an Speicherplatz für die Daten des Automaten gegenüber der „sign path"-Variante zu reduzieren und eine Implementierung zu ermöglichen.

Bewertung

In diesem Ansatz wird als Kontextinformation ausschließlich die „Historie" in Form von Handlungsabfolgen verwendet. Dieser Ansatz erlaubt durch die spezielle Spezifikation in Form eines Protokolls bzw. eines endlichen Automaten keine Erweiterung auf andere Arten von Kontextinformationen. Durch die Verwendung von Capabilities erstreckt sich die Anwendbarkeit des Systems nicht nur auf Handlungsabfolgen innerhalb einer einzelnen Anwendung, sondern kann flexibel eingesetzt

werden. Das Konzept scheint nach der Komplexitätsbetrachtung des Autors skalierbar zu sein. Leider existiert keine Implementierung, so dass der Beweis der Realisierbarkeit nicht erbracht wird.

4.3.3 Generalized Role Based Access Control (GRBAC)

Ein parallel zu dieser Arbeit entstandener Ansatz, der auf ein ähnliches Einsatzszenario abzielt, ist das Modell der „Generalized Role-Based Access Control" (GRBAC) [CAS01, MA01]. Dieses hat jedoch das spezielle Ziel, den Zugriff auf kontextabhängige Anwendungen abzusichern, im Gegensatz zu dieser Arbeit, die den Zugriff auf beliebige Anwendungen in ubiquitärer Umgebung einschränken möchte. Das GRBAC-Modell wurde im Rahmen des Projekts „Aware Home" am Georgia Institute of Technology in Atlanta, Georgia USA, von Matthew J. Moyer und Mustaque Ahamad entwickelt.

Im Projekt „Aware Home" soll das Haus der Zukunft entwickelt werden, das sich seiner Bewohner und deren Aktivitäten bewusst ist. Die Informationen über die Bewohner werden mittelt Sensoren, wie beispielsweise Kameras und Mikrophone, gesammelt und den Bewohnern und Anwendungen, die das tägliche Leben erleichtern sollen, zur Verfügung gestellt. Der Zugriff auf die gesammelten Informationen und Anwendungen kann dabei von innerhalb wie auch von außerhalb des Hauses erfolgen. Da die gesammelten Informationen zum größten Teil sensitiver Natur sind, da sie über das Verhalten und die Vorlieben der Bewohner Auskunft geben, müssen diese vor unautorisiertem Zugriff geschützt werden. Das vernetzte Haus und seine Bewohner müssen vor „virtuellen Attacken" geschützt werden [MA00].

GRBAC als Erweiterung von RBAC

Um dieses Problem zu Lösen, wurde das GRBAC-Modell entwickelt. Es setzt auf das bekannte und ausreichend erforschte Role-based Access Control (RBAC)-Modell auf. Nach der Argumentation der beiden Entwickler des GRBAC-Modells Moyer und Ahamad gibt es im konventionellen RBAC-Modell keine Möglichkeit, beispielsweise Zeit- oder Inhaltsabhängigkeiten zu unterstützen. Diese Mängel beheben sie durch das Hinzufügen von Umgebungs-Rollen (engl. environment roles) und Objekt-Rollen (engl. object roles) zu den vom RBAC-Modell abgeleiteten Subjekt-Rollen.

Subjekt-Rolle

Die Subjekt-Rolle im GRBAC-Modell entspricht der Subjekt-Rolle des RBAC-Modells. Sie repräsentiert einen Mitarbeiter einer Organisation. Die zur Klassifikation verwendeten Attribute der Subjekte können beispielsweise der Beruf, die Position in der Organisation oder die Sicherheitsstufe sein.

Objekt-Rolle

Eine Objekt-Rolle in GRBAC ist äquivalent der Subjekt-Rolle. Sie stellt eine Gruppierung oder Klassifikation von Objekten anhand sicherheitsrelevanter Attribute dar. Solche Attribute können beispielsweise die Dateigröße, das Erstellungsdatum, die Vertraulichkeitsstufe oder der Objekttyp sein.

Umgebungs-Rolle

Die Umgebungs-Rolle wird verwendet, um sicherheitsrelevante Informationen über die Umgebung zusammenzufassen. Sie bietet die gleichen Möglichkeiten einer Gruppierung oder Klassifikation von Kontextdaten wie Subjekt- und Objekt-Rollen für Subjekte bzw. Objekte. Der Unterschied ist, dass beliebig viele Umgebungs-Rollen in einer Sicherheitspolitik definiert werden können. Zum Zeitpunkt der Entscheidungsfindung müssen die aktuell aktiven Umgebungs-Rollen bestimmt und aus diesen diejenigen ermittelt werden, die für die betrachtete Transaktion relevant ist.

Transaktion

Eine GRBAC-Transaktion entspricht im Wesentlichen einer Transaktion in RBAC. Sie spezifiziert eine durchzuführende Aktion im System. Eine GRBAC-Transaktion ist ein Tupel der Form *<SRole, ORole, ERole, op>*, wobei *SRole* die aktive Rolle des agierenden Subjekts, *ORole* die Rolle des Zielobjekts, *ERole* die benötigte Umgebung und *op* die an dem Objekt durchzuführende Operation festlegt.

Eine GRBAC-Sicherheitspolitik besteht somit aus einer Menge von Politik-Regeln *<transaction, permission bit>*. Das *permission bit* legt dabei fest, ob die spezifizierte Transaktion erlaubt oder verboten ist.

Bewertung

Eine Bewertung von GRBAC schließt die Beurteilung mit ein, ob eine Erweiterung des RBAC-Konzepts in der Art und Weise, wie es in GRBAC vorgeschlagen wird, sinnvoll ist. Das RBAC-Modell wurde als Sicherheitsmodell für innerbetriebliche Bedürfnisse konzipiert und spiegelt die Organisationsstruktur wider. Damit sind die Rechte für die Zugriffe auf Objekte aufgabenorientiert vergeben. Da sich innerbetriebliche Organisationsstrukturen selten ändern, sind Änderungen in der Rechtestruktur selten. Personelle Änderungen (z.B. geänderte Aufgabenbereiche, geänderte Position oder Neueinstellungen) haben keine Auswirkungen auf die Zugriffsrechte, da diese an Rollen und nicht an Personen gebunden sind. Nachteile des RBAC-Modells sind zum einen die subjektbezogene Sicht und zum anderen die Einschränkungen durch die Konzentration auf eine einzige Dimension.

Diese Einschränkung auf eine Dimension wird durch die beiden neuen Rollen im GRBAC-Modell aufgehoben. Die Interpretation eines bestimmten Umgebungszustandes als Umgebungs-Rolle ist jedoch nicht einsichtig. Im klassischen Sinne wird eine Rolle als ein Gruppierungsmechanismus verstanden, welcher zur Vereinfachung der Rechteadministration dient. Eine Subjekt-Rolle ist nicht eine Klassifikation eines Subjekts, sondern eine Gruppierung von Transaktionen und daher nicht als inhärent für ein Subjekt zu betrachten. Es erfolgt eine explizite Zuweisung von Rollen an Subjekte. Im Gegensatz dazu ist eine Umgebungs-Rolle, wie sie in GRBAC verstanden wird, eine spezielle Konstellation von Umgebungsvariablen und deren Werte. Sie stellt somit eine sich ständig ändernde Größe dar, was dem Rollenprinzip widerspricht. Eine bestimmte Konstellation der Umgebungsvariablen kann nicht als Rolle interpretiert werden, sondern ausschließlich als einschränkende Bedingung für die Erteilung von Rechten, wie es in dieser Arbeit der Fall ist.

Eine Objekt-Rolle ist eine Zusammenfassung von Objekt-Attributen. Die Zuweisung einer Objekt-Rolle entspricht einer Klassifikation und könnte in GRBAC mit einer äquivalenten Umgebungs-Rolle abgedeckt werden.

4.3.4 Content-based Access Control

Das Content-based Access Control-Modell von Luigi Giuri und Pietro Iglio [GI97] stellt wie schon GRBAC eine Erweiterung des RBAC-Modells dar. Mithilfe der Kontextinformation „Inhalt des Zugriffszielobjekts“ bilden die Autoren das Konzept der „Views“ aus der Datenbank-Welt nach.

Die Realisierung erfolgt mit Hilfe von *restricted privileges* und *role-templates*. Ein restricted privilege ist ein Tripel (*am*, *o*, *exp*) bestehend aus dem Zugriffsmodus *am*, einer Objektmenge *o* und einem Ausdruck *exp*, welcher abhängig vom Inhalt des Objekts die Menge der Objekte weiter einschränkt. Das folgende Beispiel zeigt eine Einschränkung der Aktion „delete“ auf die Daten von Patienten, die als „entlassen“ gekennzeichnet sind:

(delete, PatientRecord, PatientRecord.State = „discharged’)

Role-templates erweitern das Konzept von Rollen durch die Kapselung von *restricted privileges*. Jedem Rollentemplate werden Parameter zugewiesen, die bei der Zuweisung an ein Subjekt gebunden werden. Das bedeutet, dass erst mit der Auswertung der aktuellen Parameterwerte bei der Zuweisung die eigentliche Rolle ermittelt und zugewiesen wird.

Bewertung

Dieses Konzept erweitert das RBAC-Modell nur um Objektbedingungen und ist daher nicht ausreichend. Allgemeine Kontextbedingungen können nicht formuliert werden, da ausschließlich die Attribute der Zielobjekte als Kontextinformationen verwendet werden.

4.3.5 History-Based Access Control

Das History-based Access Control-Modell von Edjlali, Acharya und Chaudhary wurde zum Schutz von und vor Mobilen Agenten entwickelt [EAC98]. Während die Standardmethoden zur sicherheitstechnischen Beurteilung von mobilem Code die Identität des Besitzers und die Lokation, von der der Code stammt, einbeziehen, werden in diesem Ansatz die vergangenen Aktionen des Agenten in die Sicherheitsbeurteilung miteinbezogen. Ein Beispiel einer möglichen Politik bei einer History-based Access Control ist: „ein Agent darf eine Datei nur öffnen, wenn er keine Socketverbindung offen hat, und er darf nur eine Socketverbindung öffnen, wenn keine Datei geöffnet ist“.

Die Autoren stellen an eine Realisierung ihres Modells folgende Anforderungen:

- Die Identität eines Programms muss berechenbar und eindeutig sein. Diese Forderung stellt eine Herausforderung dar, da nachladbarer Code das eigentliche Programm des mobilen Agenten verändert. Dieses Problem

wurde durch die so genannten *Deeds* gelöst. Alle evtl. benötigten Libraries werden statisch gelinkt und als eindeutige Identität ein Hashwert mit SHA-1 über den Code berechnet.

- Es muss einen effizienten Mechanismus zum Archivieren der Ereignisse sowie einen effizienten Abfragemechanismus geben.
- Eine persistente Speicherung der Politiken und der Ereignisse muss möglich sein.
- Da durch die mögliche Feingranularität eine große Menge von Regeln anfallen kann, muss es einen Gruppierungsmechanismus für die Politikregeln geben.

Während für die erste Forderung eine Lösung – die Deeds – in der Arbeit präsentiert wird, wird die Lösung der anderen Forderungen den individuellen Realisierungen des Konzepts überlassen.

Bewertung

Dieser Ansatz beschränkt sich ausschließlich auf die Kontextinformation „Historie". Die Realisierung baut auf den Java-Sicherheitsmechanismen auf und schränkt damit die Anwendbarkeit des Modells auf Java-Programme ein. Die Zuordnung einer Sicherheitspolitik zu einem Java-Programm erfolgt durch die Modifikation des Bytecodes. Als Event-Handler wird ein allgemeiner Mechanismus wie beispielsweise Java Beans vorgeschlagen. Um Ereignisse zu erzeugen, ist der Code der Programme entsprechend zu erweitern. Mit diesen Vorschlägen zur Realisierung ist eine allgemeine Einsetzbarkeit des Systems nicht gegeben. Das System ist nicht skalierbar, da für jedes individuelle Programm, das mit einer History-based Access Control versehen werden soll, eine individuelle Modifikation des entsprechenden Bytecodes vorgenommen werden muss.

4.4 Relevante Arbeiten im Bereich Politiksprachen

Zur Durchsetzung einer Sicherheitspolitik ist die Spezifikation der Politik in einer Sicherheitspolitiksprache notwendig. Unter einer Zugriffskontrollpolitiksprache – im Weiteren kurz Politiksprache genannt – wird ein Framework zur Spezifikation der Regeln einer Zugriffskontrollpolitik verstanden. Dieses sollte mächtig genug sein, die verschiedensten Zugriffskontrollmodelle zu unterstützen, aber auch einfach genug, um in einem Referenzmonitor implementiert werden zu können. Die Sprache sollte die Komposition von mehreren Politiken erlauben und auf Konflikte und Inkonsistenzen überprüfbar sein.

In den folgenden Abschnitten werden die wichtigsten Vertreter von Politiksprachen vorgestellt, welche über die Möglichkeit zur Spezifikation von Bedingungen verfügen. Diese steht bei der Betrachtung der Sprachen im Vordergrund, da die Fähigkeit komplexe Kontextbedingungen zu formulieren für eine mögliche Politiksprache zur Spezifikation einer kontextabhängigen Zugriffspolitik ausschlaggebend ist.

4.4.1 Adage

Das Authorization Toolkit for Distributed Applications and Groups (Adage) des Open Group Research Institutes in Cambridge hatte das Ziel, Administratoren bei der Festlegung von Zugriffsrechten in verteilten Systemen zu unterstützen. Dabei sollten die verschiedenen Sicherheitsmechanismen unterstützt werden. Um dies zu erreichen, wurde Adage auf folgenden Prinzipien aufgebaut:

- Benutzer-zentrierter Ansatz,
- Politikneutralität,
- modulare Architektur und
- rollenbasiertes Design.

Das Adage-Projekt umfasste neben der Definition der Authorization Language, die Entwicklung eines Editors und eines Trust-Modells [Adage99, BY97, HMS+96, SZ97].

Das Autorisierungsmodell basiert sowohl auf der Lampson-Matrix wie auch auf dem RBAC-Modell. Adage erweitert diese Modelle um komplexe Regeln zur Autorisierung. Eine Adage-Regel stellt eine Generalisierung einer ACL dar. Sie ist definiert als Tupel $<T, R, A, S, D>$, wobei T eine Menge von Teams, R eine zu erfüllende Relation, A die durchzuführende Aktion, S die Zielmenge und D eine Zeitbedingung ist. Die Relation R kann dabei verwendet werden, um Bedingungen wie Aufgabentrennung (engl. separation of duty) zu spezifizieren.

Die Authorization Language von Adage unterstützt verschiedene Arten von Bedingungen [BY97]:

- *Vertraulichkeits- und Integritätsbedingungen*: durch den Vergleich der Sicherheitsstufen von Aufrufer und Zielobjekt,
- *Trust-Bedingungen*: erfordert jedoch, dass der Aufrufer über ein entsprechendes Attribut, welches seine Vertrauensbeziehung zur aktuellen Zelle ausdrückt, besitzt,
- *Rollen-basierte Bedingungen*: aktuelle Zugehörigkeit des Aufrufers zu Rollen,
- *Zugehörigkeitsbedingungen*: Beziehungen zur Systemvergangenheit und zu Mengen sowie
- *Zeitbedingungen*.

Die Möglichkeiten, die die Politiksprache von Adage bietet, sind sehr umfangreich. Durch die den Subjekten zugeordneten Labels, die beliebige Informationen wie beispielsweise Sicherheitsstufen anheften können, bietet sie eine einfache Methode, die klassischen MAC-Modelle zu spezifizieren. Kontextinformationen wie Attributwerte, Zeit oder Vergangenheit können ausgedrückt werden. Einige der Bedingungen, können in der Authorization Language auf zwei unterschiedliche Arten spezifiziert werden: Authorization Language-konsistent und Tcl (Tool command language)-konsistent. Dieses Merkmal der Nicht-Eindeutigkeit der Authorization Language ist sehr fragwürdig und für den Benutzer verwirrend, doch die Autoren

argumentieren, dass Studien gezeigt hätten, dass Anwender oftmals unterschiedliche Bezeichnungen für das Gleiche verwenden würden und daher auch hier keine Schwierigkeiten damit hätten. Die Authorization Language spezifiziert keine Algebra für die Regeln, wodurch eine Komposition von Politiken nicht möglich ist. Die direkte Folge davon ist, dass in Adage keine Konfliktlösungsstrategie benötigt wird.

4.4.2 Ponder

Umfangreiche Arbeiten im Bereich Management-Politiken wurden am Department of Computing des Imperial College London unter Führung von Prof. Morris Sloman durchgeführt. Dabei entstand die Politiksprache „Ponder“ [SL99, DDL+00, DDL+01].

Ponder wurde speziell für den Einsatz im Bereich Netzmanagement konzipiert. Ausgehend von der Annahme, dass in einer Management-Domäne unterschiedliche Politiken zum Einsatz kommen, besitzt Ponder die Möglichkeit, Metapolitken zu definieren. Daher können Bedingungen auf zwei Ebenen spezifiziert werden: auf der einfachen Politik-Ebene und der Metapolitik-Ebene, wodurch Sicherheitsprinzipien wie Separation-of-Duty darstellbar sind.

Um Bedingungen in Ponder spezifizieren zu können, wird eine Untermenge der Object Constraint Language (OCL) eingesetzt [OCL97]. OCL wurde 1997 von IBM für UML 1.1, einem Industriestandard der OMG für objektorientiertes Software-Design [UML97], entwickelt mit dem Ziel, Bedingungen eindeutig spezifizieren zu können ohne formale Methoden verwenden zu müssen, welche für Laien nur schwer verständlich sind. OCL ist eine einfache, typisierte Ausdruckssprache, welche Bedingungen als Einschränkung von einem oder mehreren Werten des Objektmodells formuliert.

Ponder bietet die Möglichkeit, folgende Bedingungsklassen zu spezifizieren:

- Objekt- und Subjekt-Zustandsbedingungen: durch Vergleiche von Attributwerten von Aufrufer oder Zielobjekt,
- Ereignisbedingungen: Bedingungen basierend auf Attribute des aktuellen Ereignisses und
- Zeitbedingungen: Bedingungen basierend auf Zeit und Datum.

Ponder ist eine ausdrucksstarke, deklarative Politiksprache, welche die Spezifikation von Rollen-basierten wie auch allgemeinen Sicherheitsmodellen erlaubt. Da Ponder für das politikbasierte Netzmanagement entwickelt wurde, ist sie nicht nur für Autorisierungspolitiken (engl. Authority Policies) sondern auch für Obligation Policies ausgelegt, welche die Auslösung von Aktionen auf Objekte beim Eintreffen von bestimmten Ereignissen fordern. Damit bietet Ponder mehr Funktionalität als für diese Arbeit benötigt wird, aber auch ein höheres Maß an Komplexität. Das erschwert die Handhabung und geht auf Kosten der Benutzerfreundlichkeit. Die Möglichkeiten zur Spezifikation von Bedingungen, sind jedoch bei weitem nicht ausreichend.

4.4.3 LaSCO

Die Language for Security Constraints on Objects (LaSCO) wurde am Department of Computer Science der University of California entwickelt [Hoa00, HPL98, HPL01]. LaSCO stellt einen Versuch dar, durch einen graphischen Ansatz Sicherheitspolitiken benutzerfreundlicher spezifizieren zu können.

Politiken werden in LaSCO als gerichtete Graphen dargestellt, welche mit Ausdrücken beschriftet werden, die die eigentlichen Bedingungen ausdrücken, unter der die Politik anwendbar ist, und die Anwendungsdomäne festlegen. Abbildung 16 zeigt ein Beispiel einer LaSCO-Politik, welche besagt, dass ein Student eine Arbeit nur einreichen kann, solange der Lehrer die Musterlösung noch nicht in der Datenbank veröffentlicht hat.

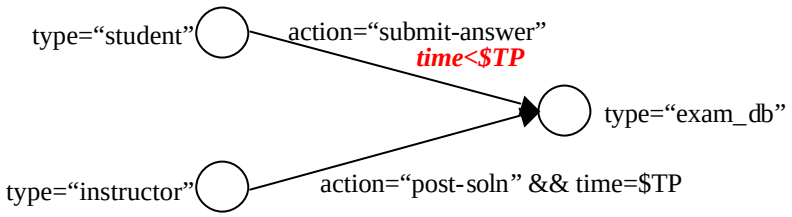

Abbildung 16 LaSCO-Politik

Dieses Beispiel zeigt, dass es in LaSCO möglich ist, Bedingungen auszudrücken, welche Bezug auf vergangene Ereignisse nehmen. Vergleiche von Attributwerten sind möglich, doch nur von direkt am betrachteten Ereignis beteiligten Komponenten. Nicht spezifiziert werden können in LaSCO Bedingungen, welche Bezug auf unterschiedliche Zustände eines Objekts nehmen oder Bedingungen, die das Nicht-Auftreten eines anderen Ereignisses voraussetzen. Die graphische Darstellung scheint auf den ersten Blick benutzerfreundlich zu sein; Studien wurden dazu von den Autoren jedoch nicht durchgeführt. Jedoch bedingt diese Darstellung eine Einschränkung in der Ausdrucksstärke der Politik.

4.4.4 SPSL

Die Security Policy Specification Language (SPSL) ist ein Versuch der Internet Engineering Task Force (IETF) eine Sicherheitspolitik für den Bereich der Netzkommunikation zu spezifizieren. SPSL wird eingesetzt, um Regeln für Firewalls oder die Protokolle IPSec und IKE (Internet Key Exchange) festzulegen.

SPSL ist eine allgemein gehaltene, ausdrucksstarke Politiksprache, in der Politiken zum Austausch von kryptografischen Schlüsseln bis zur Definition von Firewall-Regeln ausgedrückt werden können. Diese Komplexität aber macht sie schwer handhabbar; selbst eine Festlegung einer Untermenge der Politik ist dadurch schwierig. Die Festlegung der Regeln erfolgt über Textdateien. Die Syntax ist schlecht lesbar. Da sie für Politiken im Kommunikationsbereich ausgerichtet ist, ist sie für Politiken auf Anwendungsebene wenig geeignet.

4.4.5 RCL2000

Eine Politiksprache für RBAC, welche Bedingungen berücksichtigt, wurde am Department of Software Information System UNC Carlotte von Gail-Joon Ahn entwickelt und basiert auf den Arbeiten von Chen und Sandhu [CS95]. Die Role-based Constraints Specification Language 2000 (RCL2000) [Ahn99] basiert auf Prädikatenlogik erster Ordnung und ermöglicht speziell für das RBAC-Modell die Spezifikation von Bedingungen des Typs „Separation of Duty".

RCL2000 ist abgestimmt auf das RBAC-Modell und „Separation of Duty". Dies stellt eine enorme Einschränkung dar. Sie erlaubt auch keine Spezifikation von beliebigen Bedingungen. RCL2000 fand keine Verbreitung und die Weiterentwicklung wurde daher aufgegeben.

4.4.6 Akenti Policy Language

Akenti ist ein Projekt der Imaging and Distributed Collaboration Group des Lawrence Berkeley National Laboratory, Berkeley [Akenti]. Akenti ist ein Zugriffskontrollsystem für ein verteiltes Web-basiertes System und ermöglicht einen sicheren Fernzugriff auf signierte Daten anhand von Zertifikaten und Public Key-Verfahren. Die prototypische Implementierung von Akenti basiert auf einem SSL-gesicherten Apache Web-Server.

Um eine Zugriffspolitik im Akenti-Projekt zu spezifizieren, stehen drei Zertifikatstypen zur Verfügung: das Policy Certificate, Use Condition Certificate und Attribute Certificate [Tho01]. Das Policy Certificate legt fest, welche Certificate Authorities (CA's) als vertrauenswürdig eingestuft sind und anerkannte Zertifikate ausstellen dürfen. Das Attribute Certificate legt die Attribute für die beteiligten Entitäten fest. Das Use Condition Certificate enthält die Autorisierungsinformationen, die festlegen, welche Anwender welche Aktionen auf welchen Ressourcen durchführen dürfen. Dieses Zertifikat enthält einen booleschen Ausdruck, welcher bestimmt, welche Bedingungen der Anwender erfüllen muss, um Rechte zu erhalten. Für die Zukunft ist geplant, dass dieser boolesche Ausdruck auch auf Attribute der Aktion und des Objekts zugreifen kann sowie auf aktuelle Systemwerte.

Die Akenti Policy Language ist speziell auf das Akenti-Projekt ausgerichtet und kann nicht ohne weiteres auf andere Anwendungsfelder übertragen werden. Die aktuelle Version bietet bezüglich der Definition von Kontextbedingungen nur geringe Unterstützung und ist daher für eine kontextabhängige Zugriffspolitik nicht geeignet.

Kapitel 5

Modellierung von Kontext

Der erste Schritt bei der Entwicklung einer kontextabhängigen Anwendung ist die Festlegung der Kontextarten, welche für diese Anwendung relevant sind. Der nächste Schritt ist eine Methode zur Spezifikation von Kontextinformationen basierend auf einem entsprechenden Kontextmodell zu finden. Während im vorherigen Abschnitt 3.3.1 verschiedene Ansätze aus der Literatur für die Definition von Kontext im Bereich Ubiquitous Computing und Mobile Computing aufgezeigt wurden, wird in diesem Kapitel vorgestellt, wie der Begriff in dieser Arbeit verstanden wird. In Abschnitt 5.2 werden die Begriffe „Kontext“ und „Kontextinformation“ erläutert und eine umfassende Klassifikation von Kontextinformationen vorgenommen, welche in dieser Arbeit unterstützt werden sollen. Abschnitt 5.3 enthält die Beschreibung einiger besonderer Eigenschaften von Kontextinformationen. In Abschnitt 5.4 wird die dieser Arbeit zugrunde liegende Spezifikation von Kontextinformationen vorgestellt und daran angelehnt in Abschnitt 5.4 die Spezifikation von Kontextabfragen.

Um einige Begriffe, die bei der folgenden Spezifikation von Kontext und Kontextinformation sowie bei der Beschreibung der kontextabhängigen Zugriffskontrolle in Kapitel 6 verwendet werden, besser verstehen zu können, wird zunächst in Abschnitt 5.1 die dieser Arbeit zugrunde liegende Sichtweise eines ereignisbasierten Systemmodells erläutert.

5.1 Ereignisbasiertes Systemmodell

Für die Entwicklung einer formalen Spezifikation einer Zugriffspolitik ist es notwendig, ein abstraktes Systemmodell zugrunde zu legen. Dazu wird im Folgenden ein ereignisbasiertes Systemmodell eingeführt, das es erlaubt, sowohl Systemeigenschaften wie auch kausale und temporale Abhängigkeiten von Ereignissen in einem verteilten System auf einem hohen Abstraktionsniveau zu spezifizieren.

Ein verteiltes System wird als eine Menge von *Komponenten* betrachtet, welche über *Ereignisse* miteinander interagieren. Ereignisse treten dabei zu diskreten Zeitpunkten auf. Mehrere Ereignisse können parallel ablaufen. Asynchrone Aufrufe be-

stehen aus einem Ereignis, synchrone aus zwei *Ereignissen*, dem aufrufenden und dem terminierenden. Einem aufrufenden Ereignis können Übergabeparameter mitgegeben werden, das terminierende Ereignis kann Ausgabedaten zurückliefern.

Angestoßen durch Ereignisse, führt jede Komponente intern Aktionen durch. Jede Aktion bewirkt dabei eine Zustandsänderung der Komponente. Eine Komponente kann infolge der Ausführung einer Aktion ihrerseits beliebige Ereignisse erzeugen. Eine Komponente verfügt über eine endliche Menge von Attributen, die sich über die Zeit verändern können; Gleiches gilt für Ereignisse. Da die intern in einer Komponente ausgeführten Aktionen keine weitere Bedeutung bei der Modellierung besitzen, kann diese als eine „Black Box" betrachtet werden, mit einem Zustand, Eigenschaften und Schnittstellen, welche nach innen bzw. außen führen.

Im folgenden Abschnitt werden verschiedene Definitionen eingeführt und Festlegungen getroffen, welche für die darauffolgenden Spezifikationen und das Verständnis der Zugriffspolitiksprache nützlich sind. Die Notation ist an die Notation von Rapide [LV93, LKA+95, Ken96] und die Spezifikation von LaSCO [HPL98, HPL98b] angelehnt.

5.1.1 System, Komponenten, Ereignisse und Attribute

Einige Begriffe, die in den folgenden Abschnitten und Kapiteln verwendet werden, werden im Folgenden genauer definiert:

Definition 1: Komponente

Eine Komponente *c* ist eine Systemeinheit, die über einen eindeutigen Namen identifiziert werden kann. Sie wird charakterisiert durch eine Menge von Attributen $A = \{a_1, a_2, \ldots a_n\}$, deren Bindungen zu einem Zeitpunkt *t* den Zustand der Komponente zu diesem Zeitpunkt festlegen. Sei der Wertebereich der Attribute a_i gegeben durch $W(a_i)$, dann wird die Menge der Zustände CS_C der Komponente *c* bestimmt durch das kartesische Produkt:

$$CS_C = \mathbf{X}_{i=1}^{n} W(a_i)$$

Definition 2: Attribut und Attribut-Bindung

Ein Attribut besitzt eine eindeutige Identifikation und zu jedem beliebigen Zeitpunkt einen bestimmbaren Wert.

Eine Attribut-Bindung für ein Attribut a ist ein Tupel (α, μ) mit

- α Attribut-Name und
- $\mu \in W(a)$ der Wert des Attributs a.

Definition 3: Ereignis

Ein Ereignis *e* ist eine momentane Systemaktivität. Sie wird durch eine Komponente *s* (Produzent) initiiert und endet bei einer Komponente *o* (Konsument). Sie wird weiter charakterisiert durch eine Menge *A* von Attributen. Durch die beiden Attribute „Ereignistyp“ und „Zeit“ wird das Ereignis eindeutig identifizierbar. Es gilt:

$e = (s, o, A)$ mit $\quad src(e)=s$, $dest(e)=o$ und $attr(e)=A$, sowie

$time(e) = t$ mit $a_t=(\text{´time´}, t) \in A$ *und*

$type(e) = et$.

Definition 4: Systemzustand

Ein Systemzustand ist eine Momentaufnahme eines Systems zu einem Zeitpunkt *t* mit einer Menge *C* von Komponenten und einer Menge *E* von anstehenden Ereignissen:

$$S_t = (C_t, E_t)$$

Definition 5: System

Unter einem System wird die Gesamtheit der betrachteten Komponenten und Ereignisse über die Zeit verstanden. Zu jedem Zeitpunkt besitzt das System einen eindeutigen Systemzustand. Das Eintreten eines Ereignisses bewirkt einen Zustandsübergang.

Somit ist ein System $S = \{(C_t, E_t) \mid t \in N\}$ die Menge aller Systemzustände über die Zeit. Die Zeit wird dabei als diskrete, monoton steigende Größe betrachtet.

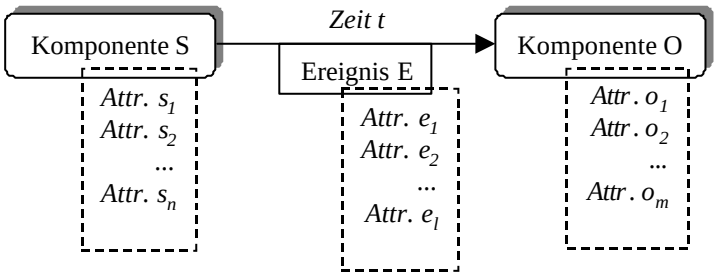

Abbildung 17 Ereignisbasiertes System

5.1.2 Abhängigkeiten zwischen Ereignissen

Ereignisse in einem verteilten System können sowohl unabhängig wie auch kausal oder temporal abhängig sein. Kausale und temporale Abhängigkeiten definieren auf der Menge der Ereignisse partiell geordnete Untermengen. Temporale und kausale Abhängigkeiten von Ereignissen setzen eine gemeinsame Uhr voraus (siehe Abschnitt 5.1.4).

Definition 6: Temporale Abhängigkeit

Ein Ereignis e´ ist *temporal abhängig* vom Ereignis e relativ zu einer Uhr U (im Zeichen $e \rightarrow_U e'$), wenn der Startzeitpunkt des Ereignisses e vor dem Startzeitpunkt des Ereignisses e´ liegt.

Definition 7: Kausale Abhängigkeit

Ein Ereignis e´ ist *kausal abhängig* von einem Ereignis e (im Zeichen $e \rightarrow e'$), wenn

1. die Ereignisse e und e´ von der gleichen Komponente c erzeugt wurden und e zeitlich vor e´ liegt (e´ temporal abhängig von e); Oder
2. eine Komponente c das Ereignis e annimmt und als Reaktion das Ereignis e´ erzeugt; Oder
3. es ein Ereignis e´´ gibt, so dass e´´ kausal von e und e´ kausal von e´´ abhängig ist (Transitivität).

Definition 8: Unabhängigkeit

Zwei Ereignisse e und e´ heißen unabhängig (im Zeichen $e \parallel e'$), wenn gilt:

1. Ereignis e ist nicht kausal abhängig von e´,
2. Ereignis e´ ist nicht kausal abhängig von e und
3. e ist ungleich e´.

$$e \parallel e' \Leftrightarrow \neg(e \rightarrow e') \wedge \neg(e' \rightarrow e) \wedge e \neq e'$$

Definition 9: Posets

Eine Menge von partiell geordneten Ereignissen wird Poset genannt. Die partielle Ordnung auf einer Menge von Ereignissen ist nicht-reflexiv, antisymmetrisch und transitiv.

5.1.3 Ereignis-Muster

Ereignis-Muster (engl. Event Pattern) sind reguläre Ausdrücke zur Spezifikation von Posets. Der Prozess, der einen Poset daraufhin prüft, ob dieser einem Ereignis-Muster entspricht, wird Pattern Matching genannt.

Ein Ereignis-Muster wird nach folgender Syntax gebildet:

pattern ::= basic_pattern
| ´(´pattern´)´
| empty | any
| pattern binary_pattern_op pattern
| pattern ´^´ ´(´ iterator_expression binary_pattern_op ´)´
| pholder_decl_list pattern
| pattern_macro

binary_pattern_op ::= ´®´ | ´®$_C$´ | ´||´ | or | and | not

iterator_expression ::= ´´ | ´+´ | expression*

pholder_decl_list ::= ´(´ pholder_decl { ´:´ pholder_decl } ´)´

*pholder_decl ::= ´?´ident { ´,´ ident } **in** expression*
*| ´!´ident **in** expression **by** operator*

Ein **Grundmuster** `basic pattern` entspricht dem Namen einer Aktion mit den assoziierten Parametern. **Zusammengesetzte Muster** ergeben sich durch die Kombination von Mustern. Die **Grundoperationen** `binary_pattern_op` sind:

- *Temporale Abhängigkeit „$P \rightarrow_U P'$“*
  alle Ereignisse, die das Muster P erfüllen, liegen zeitlich vor den Ereignissen, die P´ erfüllen,
- *Kausale Abhängigkeit „$P \rightarrow P'$“*
  alle Ereignisse, die das Muster P erfüllen, sind kausal abhängig von allen Ereignissen, die P´ erfüllen,
- *Unabhängigkeit „$P \,||\, P'$“:*
  alle Ereignisse, die das Muster P erfüllen, sind unabhängig von allen Ereignissen, die P´ erfüllen,
- *Disjunktion „P or P´“*
  es wird das Muster P oder das Muster P´erfüllt.
- *Konjunktion „P and P´“*
  Muster P und Muster P´sind erfüllt und
- *Negation „not P“*
  das Muster P darf nicht erfüllt sein.

Einzelne Muster können wiederholt werden. Dazu stehen die **Iterationsoperationen** ´*´ (beliebig oft), ´+´ (mindestens einmal) oder `n` (genau *n*-mal) zur Verfügung.

Für die Definition von Makros stehen zwei **Platzhalter** `place_holder` zur Verfügung: der existenzielle „?“ und der universelle „!“ Platzhalter. Diese beiden Platzhalter entsprechen den logischen Quantoren „∃“ und „∀“.

Makros können zur Abstraktion und Parametrisierung verwendet werden, sowie um neue Operationen zu definieren. Die Syntax lautet:

pattern_macro	*::=*	**pattern** *ident ´(´[macro_param_list]´)´* **is** *pattern;*
macro_param_list	*::=*	*macro_param { ´;´ macro_param }*
macro_param	*::=*	*macro_param*
	|	*type_param*
	|	**pattern** *identifier_list*

Es sind eine Reihe von Makros gegeben, die **Zeitoperationen**. Vordefinierte Zeitoperationen sind:

pattern ***during*** *(p : Pattern; t1, t2 : Time; c : Clock)*

Ein Poset *P* erfüllt das Muster *p*, wenn alle Ereignisse aus *P* nach dem Zeitpunkt *t1* starten und spätestens zum Zeitpunkt *t2* beendet sind. Die Zeitangaben beziehen sich dabei alle auf die lokale Uhr *U*.

pattern ***at*** *(p : Pattern; t : Time; c : Clock) is during (p; ?First, ?Last; c);*

Ein Poset *P* erfüllt das Muster *p*, wenn alle Ereignisse aus *P* genau zum Zeitpunkt *t* starten und terminieren.

pattern ***before*** *(p : Pattern; t : Time; c : Clock) is*
(?First, ?Last in Time) during (p; ?First, ?Last; c) where ?Last <= t;

Ein Poset *P* erfüllt das Muster *p* und alle Ereignisse aus *P* terminieren spätestens zum Zeitpunkt *t*.

pattern ***after*** *(p : Pattern; t : Time; c : Clock) is*
(?First, ?Last in Time) during (p; ?First, ?Last; c) where ?First >= t;

Ein Poset *P* erfüllt das Muster *p* und keine Ereignisse aus *P* beginnen vor dem Zeitpunkt *t*.

5.1.4 Zeitstempel und Zeit

Werden Ereignisse miteinander in Beziehung gesetzt, so erfolgt dies unter zu Hilfenahme des Zeitstempels, welcher jedem Ereignis zugeordnet ist. Das Problem dabei ist, dass vorausgesetzt wird, dass jedes Ereignis mit der gleichen Methode erfasst und der Zeitstempel unter den gleichen Bedingungen erzeugt wurde. Zum einen besitzt ein Ereignis – nicht wie in der idealisierten Sichtweise – eine zeitliche Ausdehnung und

zum anderen kann zwischen dem Auftreten des Ereignisses und der Erzeugung des Zeitstempels Zeit vergehen.

Der Zeitstempel kann vom Ereignis-Erzeuger erstellt werden oder aber von der beobachtenden Einheit, dem Observer. Des weiteren spielt es eine Rolle, um welche Art von Observer es sich handelt, einem passiven, welcher über das Eintreffen eines Ereignisses informiert wird, oder aber um einen aktiven, welcher die Ereignisse abfragt.

Kausale wie temporale Ereignisrelationen setzen eine gemeinsame Uhr zur Bestimmung der Zeit voraus, was bei Ereignissen einer Station gegeben ist. Bei einem verteilten System treten jedoch zwei Probleme bei der Bestimmung des Zeitpunkts des Auftretens eines Ereignisses auf: unterschiedliche Systemuhren sowie Verzögerungen bei der Übertragung.

Um Ereignisse in Relation setzen zu können, ist prinzipiell keine physikalische Realzeit notwendig, sondern lediglich eine Ordnung. Einer der ersten, der sich diesem Problem angenommen hat, war Lamport. Von ihm stammt der Ansatz der diskreten logischen Uhr und der logischen Zeit [Lam78]. Lamport ordnet jedem Prozess eine logische Uhr zu. Dazu wird jedem Ereignis die Zeit des Ereignis-Erzeugers mitgegeben, wodurch der Empfänger seine Zeit synchronisieren kann, in dem er seine Zeit auf die Sender-Zeit plus Eins setzt, da ein Empfangsereignis logisch erst nach dem zugehörigen Sendeereignis auftreten kann. Eine Erweiterung dieses Ansatzes stellt die *vector clock* von Fidge [Fid91] und Mattern [Mat89] dar. Grundlage dieses Verfahrens ist die Idee, jedem Prozess einen Vektor zuzuordnen, der die lokalen Zeiten aller Prozesse speichert, von denen er durch Nachrichten Kenntnis bekommen hat.

Da in einem offenen System nicht alle Prozesse im Voraus bekannt sind, die Anzahl sehr hoch und evtl. unbestimmt ist, muss doch auf die physikalische Realzeit zurückgegriffen werden, um in einem offenen System wie dem Internet Ereignisse in Relation zu setzen. Daher ist es nicht möglich, eines der genannten Verfahren zu etablieren. Auch gelten die Voraussetzungen für diese Verfahren nicht, da nicht jedem Ereignis eine Sende- und eine Empfangszeit durch einen jeweils lokalen Observer zugeordnet werden. Es ist somit nicht möglich, die logischen lokalen Uhren durch die logische Ordnung von Sende- und Empfangsereignisse zu synchronisieren. Die Art der Erstellung des Zeitstempels – also des Observers - ist ausschlaggebend. Besitzen Ereignisse nur einen Zeitstempel, kann nur auf die physikalische Zeit vertraut werden, die verwendet wird, um diesen Zeitstempel zu erstellen, um die Ereignisse in Relation zu setzen.

Die bekanntesten Verfahren zur Uhrensynchronisation sind: der Cristian-Algorithmus, der Berkeley-Algorithmus und das Network Time Protocol. Bei der Cristians-Methode [Cri89] wird ein zentraler Zeitserver eingesetzt, welcher sich auf die Universalzeit UTC synchronisiert hat. Die Klienten fragen die Zeit beim Zeitserver ab und setzen die eigene Zeit auf die gemeldete plus der halben, vom Klienten gemessenen Round-Trip-Zeit.

Die Berkeley-Methode [GZ89] geht – im Gegensatz zur Cristian-Methode - von einem aktiven Zeitserver aus. Dieser fragt bei den Klienten deren aktuelle Zeit ab. Die gemeldete Zeit, bereinigt um die statistisch gemittelte halbe Round-Trip-Zeit, wird mit

der Zeit des Servers verglichen und die Differenz an den Klienten gemeldet. Mit diesem Verfahren wird eine Genauigkeit von ±10ms im LAN erreicht.

Ein häufig in lokalen Netzen eingesetztes Verfahren ist das Network Time Protocol (NTP) [Mil94]. Der Zeitservice ist auf mehrere Server verteilt, welche in einer Baumstruktur organisiert sind. Der Primärserver synchronisiert sich mit der UTC. Zur Synchronisation stehen drei Modi zur Verfügung:

- Im Multicast-Modus sendet der Server den Zeitwert an alle Klienten, die zur Berechnung ihrer Zeit eine feste Delay-Zeit annehmen.
- Im Procedure-Call-Modus erfragen die Klienten die aktuelle Zeit von Server und im
- symmetrischen Modus erfolgt der Abgleich durch gegenseitiges Zuschicken der Zeitwerte.

NTP erreicht damit eine Genauigkeit mit weniger als 30ms Abweichung und bei Kompensation der lokalen Uhrendrift von weniger als 1ms.

Eine weitere, aus Kostengründen jedoch selten angewendete Methode ist der Abgleich über das Global Positioning System (GPS), welches ein Universal Time Coordinated-Signal liefert. Damit kann eine Genauigkeit von ±0,1-10ms erreicht werden.

5.2 Kontext und Kontextinformation

In den folgenden Abschnitten werden die Begriffe „Kontext“ und „Kontextinformation“, wie sie in dieser Arbeit verstanden und verwendet werden, erläutert. Obwohl es einige Überschneidungen mit den in Kapitel 3.3.1 vorgestellten Definitionen dieser Begriffe in der Literatur geben wird, ist es für das weitere Verständnis dieser Arbeit sinnvoll und hilfreich.

Auf eine eigene Definition des Begriffs „kontextabhängige Anwendung” wird verzichtet, da das Verständnis dieses Begriffs in dieser Arbeit weitestgehend mit den Sichtweisen der in Abschnitt 3.3.2 vorgestellten Definitionen in der Literatur übereinstimmt.

5.2.1 Kontext

Unter dem Begriff Kontext wird in dieser Arbeit die Gesamtheit möglicher Einflussfaktoren für Akteure, Objekte, Anwendungen oder Dienste betrachtet. Es wird zwischen zwei wesentlichen Gruppen unterschieden: den Menschen und der Umwelt bestehend aus Objekten (siehe Abbildung 18). Alle Beteiligten – Subjekte wie Objekte – verfügen über Eigenschaften und Beziehungen untereinander, welche den aktuellen Systemzustand festlegen. Diese Eigenschaften können durch Ereignisse verändert werden.

Informationen über den Kontext können auf zwei verschiedenen Wegen ermittelt werden: automatisch oder manuell. Die Ermittlung von Wissen durch den Menschen

erfolgt über den Dialog und der mechanischen Eingabe in ein System zur Weiterverarbeitung. Da diese Art von Wissensverarbeitung – wie sie bei Expertensystemen gerne eingesetzt wird - zu (zeit-)aufwendig ist, kann und wird sie im ubiquitären Bereich vernachlässigt. Es steht somit im Wesentlichen nur Wissen zur Verfügung, das über jegliche Arten von Sensoren ermittelt werden kann. Wissen über den Menschen, soziale oder emotionale Zustände, Beziehungen oder zwischenmenschlichen Dialoge wird nur soweit betrachtet, wie es von Sensoren (z.B. durch Kameras, Mikrofone) erfasst bzw. aus den erfassten Sensordaten abgeleitet werden kann.

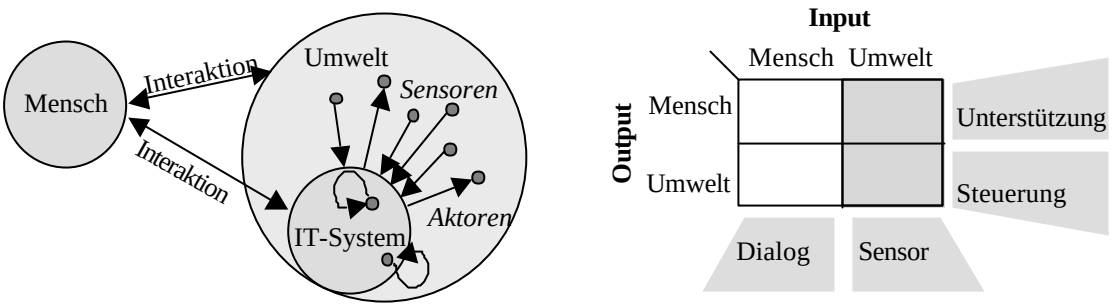

Abbildung 18 Mensch-Umwelt-Interaktion

Unter dem Begriff „Kontext“ wird in dieser Arbeit …

> *… die Gesamtheit aller (automatisch) zu erfassenden und berechenbaren Umgebungsinformationen, welche Einfluss auf das betrachtete Ziel haben könnten*

verstanden.

5.2.2 Kontextinformation

Während unter Kontext die Gesamtheit der Umgebungsdaten verstanden wird, wird der Begriff „Kontextinformation“ für eine Einheit des Kontexts verwendet. Dabei wird nicht unterschieden, ob es sich - im Sinne der Informationswissenschaft – um Daten, Informationen oder um Wissen handelt (siehe Abbildung 19). Sollte explizit der Unterschied hervorgehoben werden, so wird der Begriff „Kontextdaten“ verwendet, um zu verdeutlichen, dass es sich um „rohe“ unbearbeitete, direkt von Sensoren ermittelte Daten handelt.

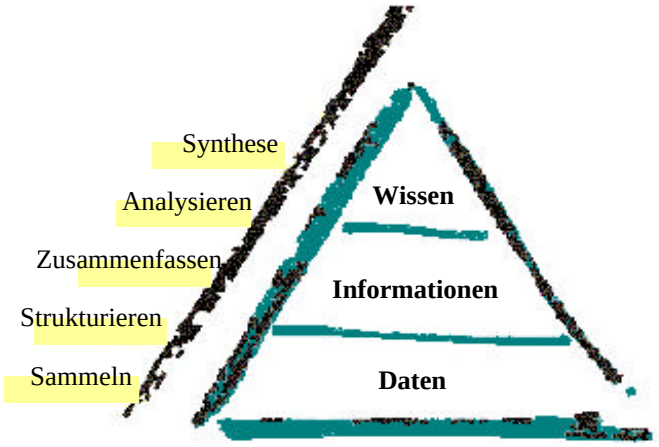

Abbildung 19 Daten, Wissen, Information (Quelle: [Kno01])

Unter Kontextinformationen werden somit in dieser Arbeit die verschiedensten Umgebungswerte verstanden, welche messbar, ermittelbar oder berechenbar sind, und Einfluss auf die Anwendung haben könnten. Umgebungswerte sind sowohl Objekt-Attribute wie auch Objekt-Beziehungen.

Zu den Kontextinformationen, die in dieser Arbeit berücksichtigt werden sollen, zählen:

- *Über Hardware-Sensoren messbare Werte*
  z.B. Anzahl der Personen, Identität der Person, Temperatur, Luftfeuchtigkeit, CO_2-Gehalt, Lichtverhältnisse
- *Zustand der Systemumgebung (Software-Sensoren)*
  z.B. CPU-Auslastung, Anzahl der Benutzer, Anzahl der geöffneten Socketverbindungen
- *Persönliche Präferenzen*
  z.B. persönliche Einstellungen für den Sm@rtAssistant [MHR01], P3P-Präferenzen
- *Zustand der Anwendungsumgebung*
  z.B. „in welchem Arbeitsschritt des Workflows befindet sich der Nutzer aktuell“, „für welches Projekt arbeitet der Benutzer gerade“
- *Zeitliche und kausale Beziehungen*
  z.B. „eine Verbindung ins Internet darf nur dann geöffnet werden, wenn keine Datei mit sensiblen Informationen zuvor geöffnet wurde“ oder „ein Gerät darf nur dann benutzt werden, wenn eine Reservierung dafür vorliegt“
- *Räumliche Beziehungen*
  z.B. „welche Personen befinden sich in der Nähe?“, „welche Geräte stehen in der näheren Umgebung zur Verfügung?“
- *Zeit und Ort.*

Nicht berücksichtigt werden in der aktuellen Version des Systems:

- *Soziale oder Emotionale Beziehungen*, die über spezielle Sensoren und interpretierende Programme geschlussfolgert werden, und
- *Situationswissen*, das über Inferenzmethoden aus anderen Kontextinformationen geschlossen wird.

Wie diese Aufzählung zeigt, gibt es unzählige Arten von Kontextinformationen. Um sie jedoch in einem IT-System bearbeiten zu können, ist eine Objektmodellierung notwendig. Die Grundlage dafür bildet eine Kategorisierung von Kontextinformationen und die Identifizierung der wichtigsten Merkmale.

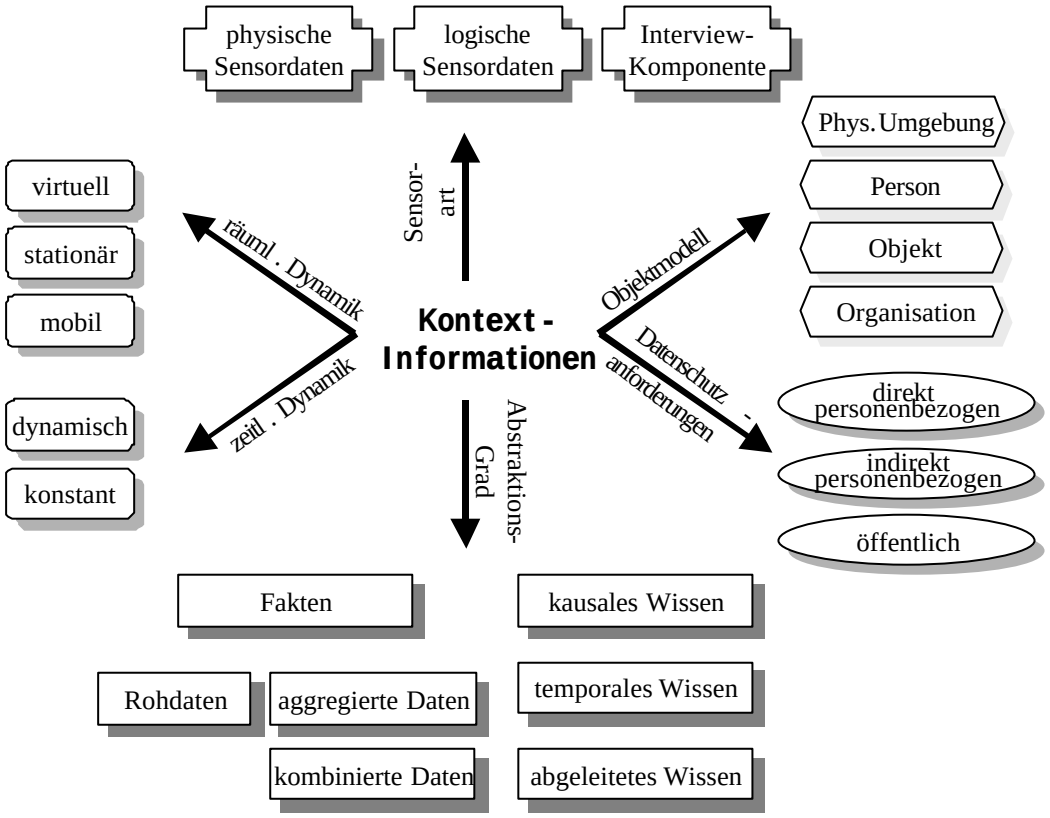

Abbildung 20 Kontextcharakteristika

In dieser Arbeit werden sechs wesentliche Charakteristika von Kontextinformationen unterschieden, welche Einfluss auf die Art der Datenakquisition, die Modellierung, die Bearbeitung und die Speicherung haben (vgl. Abbildung 20):

1. **Methode der Datenerhebung (Sensorart)**:

   Man kann zwischen physischen und logischen Sensoren sowie einer Interviewer-Komponente unterscheiden. Physische Sensoren sind beispielsweise Temperatur, Wind-, Feuchtigkeitsmesser, Chipkarten-Leser, RFID-Antennen oder GPS-Empfänger. Logische Sensoren sind Software-Komponenten, welche Zustände oder Werte in einem IT-System ermitteln. Die Werte

können Systemvariablen wie CPU-Auslastung oder Anzahl der geöffneten Netz-Verbindungen sein, aber auch Ergebnisse aus Datenbank-Abfragen wie beispielsweise die HW-Ausstattung eines Raums als Ergebnis einer Abfrage der Datenbasis eines Facility-Management-Systems. Mit einer Interviewer-Komponente werden über eine direkte Kommunikation mit den Benutzern Fakten ermittelt. Dazu zählen beispielsweise die Benutzer-Präferenzen und -Profile.

2. **Objektmodell**:

   Es können vier wesentliche Klassen von Bezugsobjekten unterschieden werden, welche Einfluss auf die Modellierung der Kontextinformation haben: Personen, Objektinhalte, organisatorische Strukturen oder Prozesse sowie physikalische Gegebenheiten. Unter physikalischen Gegebenheiten versteht man beispielsweise Zeit und Ort. Die Modellierung von Personen teilt sich in die Beschreibung der Person und deren Präferenzen. Organisatorische Strukturen schließen sowohl die Firmenstruktur (Aufbauorganisation) wie auch die Projekt- oder Ablauforganisation mit ein; mit Objektinhalten ist beispielsweise die interne Struktur einer XML-Datei gemeint.

3. **Abstraktionsgrad**:

   Wie schon erwähnt, wird in der Informationswissenschaft zwischen Daten, Informationen und Wissen unterschieden. Während für die Verwendung des Begriffs „Kontextinformation“ diese Unterscheidung nicht von Bedeutung ist, ist sie doch für die Datenmodellierung interessant, da auf den unterschiedlichen Abstraktionsebenen unterschiedliche Bearbeitungsmethoden zum Einsatz kommen, wie beispielsweise Datenaggregation, Datenkombination, verschiedene Inferenzmethoden oder Pattern-Matching bei kausalen und temporalen Abhängigkeiten.

4. **Grad des benötigten Datenschutzes**:

   Bei der Erhebung von Personen- und Umgebungsdaten sowie deren Speicherung, spielt der Datenschutz – aus juristischer Sicht in Deutschland mehr als beispielsweise in den USA – eine erhebliche Rolle [DSZ]. Gesetzliche Anforderungen verpflichten zu speziellen Speicherungs- und Sicherungsmaßnahmen, um den Missbrauch der Daten zu verhindern. Es können drei Klassen von Informationen bzgl. Datenschutz unterschieden werden: die direkten personenbezogenen Daten, die indirekten personenbezogenen Daten und die anonymen Daten. Dabei sind nach §3 BDSG (Bundesdatenschutzgesetz) personenbezogene Daten „... Einzelangaben über persönliche oder sachliche Verhältnisse einer bestimmten oder bestimmbaren Person“. Nach Art.2a der EU-Datenschutzrichtlinie wird als bestimmbar „… eine Person angesehen, die direkt oder indirekt identifiziert werden kann, insbesondere durch Zuordnung (...) zu einem oder mehreren spezifischen Elementen, die Ausdruck ihrer physischen, physiologischen, psychischen, wirtschaftlichen, kulturellen oder sozialen Identität sind.“. Im Gegensatz dazu sind anonyme

Daten solche, bei denen eine Zuordnung zu einer Person nicht oder nur mit unverhältnismäßig hohem Aufwand möglich ist. Die Einteilung der Informationen in die unterschiedlichen Klassen ist zum großen Teil durch Datenschutzgesetze vorgegeben, aber auch von der Datenschutzpolitik des Betriebes oder der betroffenen Person abhängig.

5. **Räumliche Dynamik**:

   Physische aber auch logische Sensoren können mobil oder stationär sein. Die Werte der durch diese Sensoren gewonnen Kontextinformationen sind daher vom Aufenthaltsort des Sensors zum Zeitpunkt der jeweiligen Datenerhebung abhängig.

6. **Zeitliche Dynamik**:

   Unter der zeitlichen Dynamik ist die Frequenz zu verstehen, in der sich die Werte der Kontextinformationen ändern. Physische und die Mehrzahl der logischen Sensoren ermitteln in kurzen Zeitabständen neue Werte. Obwohl die Werte mancher Kontextinformationen einer kontinuierlichen Veränderung unterliegen, erfolgt die Erfassung in der Regel in diskreten Zeitintervallen. Daten, welche über eine Interviewer-Komponente ermittelt werden, aber auch ein Teil der über logische Sensoren ermittelten Daten, verändern sich „nur“ in sehr großen zeitlichen Abständen, so dass von „statischen“ Kontextinformationen gesprochen werden kann. Darunter fallen beispielsweise Benutzerprofile und –Präferenzen, Informationen über stationäre Raumausstattungen oder Gebäudeinstallationen.

Diese sechs Charakteristika der Kontextinformationen haben direkten Einfluss auf die Modellierung der Kontextinformationen und deren Abbildung in Objektklassen und -hierarchien bei der Implementierung.

5.2.3 Kontextklassen

In dieser Arbeit werden sechs große Klassen von Kontextinformationen unterschieden: Ortsinformationen, Zeitinformationen, Sensorinformationen, Objekt- und Subjektinformationen, temporales und kausales Wissen sowie (einfaches) Situationswissen (siehe Abbildung 21).

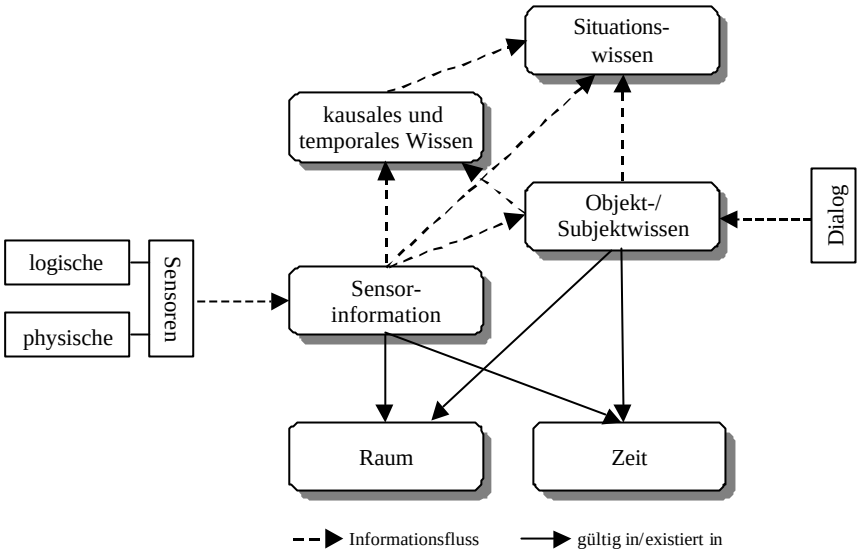

Abbildung 21 Kontext-Klassen

5.2.3.1 Ortsinformation

Eine notwendige Voraussetzung für die Festlegung und Interpretation von Kontextinformationen und deren Verwendung in einer kontextabhängigen Anwendung ist die Existenz eines Raum- oder Lokationsmodells. In der Literatur sind verschiedene Ansätze zur Raum-Modellierung zu finden. Einen guten Überblick und Zusammenfassung verschiedener Modelle sowie eine Klassifikation gibt Ulf Leonhardt in seiner Dissertation [Leo98]. Er unterscheidet zwischen zwei großen Gruppen: geometrische und symbolische Lokationsmodelle. Geometrische Modelle beschreiben Orte als Punkte, Gebiete oder Räume in einem Koordinatensystem. Symbolische Modelle beschreiben Orte durch abstrakte Symbole. Ein Vertreter des symbolischen Modells ist das „location domain model“ (dt. Bereichsmodell), das in der vorliegenden Arbeit verwendet wurde. Weitere interessante Ansätze für Raummodelle sind beispielsweise in [Nel98, LKR+99, HB00] zu finden.

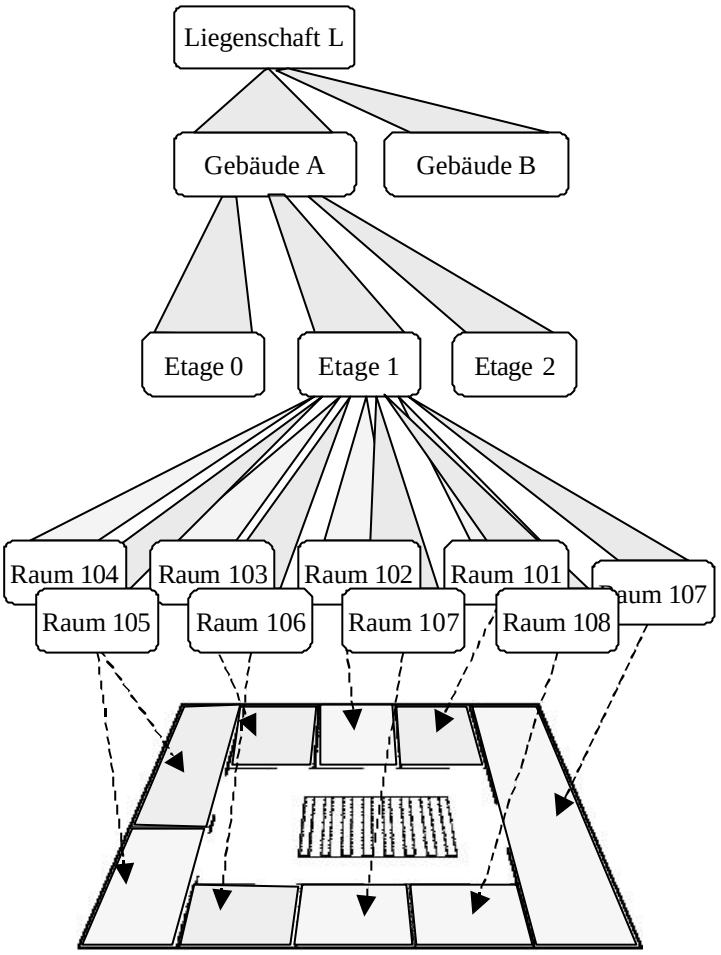

Abbildung 22 RC-Raummodell

Die Entscheidung für ein spezielles Lokationsmodell, das bei der Realisierung eines Kontext- oder Lokationssystems verwendet werden soll, hängt zum einen von den zur Verfügung stehenden Sensoren und deren Auflösungsgenauigkeit zur Bestimmung von Positionen ab, und zum anderen von den Anwendungen, welche diese Information verwenden möchten. Die untere Schranke für die Granularität der Lokationsinformation ist durch die Positionierungsgenauigkeit der verwendeten Sensoren gegeben.

Das Lokationsmodell dieser Arbeit (siehe Abbildung 22) ist auf das Raum-Computer-Projekt [RCP, MHR01] abgestimmt. Die zur Verfügung stehenden Lokationssensoren erlauben eine Positionierung von Objekten und Personen mit einer Genauigkeit von Räumen innerhalb eines vernetzten Gebäudes. Das Modell umfasst eine beliebige Anzahl von Gebäuden.

Zur Modellierung von Gebäuden eignet sich das Bereichsmodell. Das charakteristische Merkmal des Bereichsmodells ist eine partielle Ordnung auf den Zonen durch die Relation „enthält“ (vgl. Abbildung 22). Zonen des gleichen Abstraktionsniveaus überschneiden sich nicht. Ein zu lokalisierendes Objekt gehört immer eindeutig einer Zone einer Abstraktionsebene an und jeder der „Eltern“-Zonen. Damit ist eine nicht-reflexive partielle Ordnung auf der Zonenhierarchie gegeben.

5.2.3.2 Zeitinformation

Zeit ist – wie die Lokationsinformation auch - zum einen eine eigenständige Kontextinformation und zum anderen eine Qualifikation der Gültigkeit von Kontextinformationen. Bei der Modellierung von Zeit werden zwei Dimensionen unterschieden: absolute und relative Zeit, sowie Zeitpunkte und Zeitintervalle.

In der Literatur spielen temporale Aspekte vor allem auf den Gebieten temporales Schließen und den darauf aufbauenden Planungsverfahren eine wichtige Rolle. Kaum beachtet wurden dagegen temporale Aspekte bisher bei der Repräsentation von Messzeiten und der Modellierung des Zeitverhaltens von Sensoren. Doch gerade bei der Sensordatenverarbeitung spielt Zeit eine entscheidende Rolle. Verschiedene Zeitaspekte bei Sensoren und bei der Sensordatenverarbeitung sind:

- *Diskrete Abtastzeitpunkte:*

  Sensordaten sind Messgrößen diskreter Signale, denen eine diskrete Messzeit zugeordnet wird.

- *Abtastintervalle:*

  Sensordaten werden entweder in Abtastintervallen ermittelt oder bei Bedarf. Die Länge eines Abtastintervalls spielt dabei eine wesentliche Rolle für die Gültigkeit der Sensordaten.

- *Zeitcharakteristika von Sensoren:*

  Zwischen der tatsächlichen Datenerhebung und der Bereitstellung der Sensordaten vergeht in der Regel Zeit, da teilweise erste Berechnungen oder Auswertungen in den Sensoren stattfinden. Verschiedene Sensortypen verfügen daher über unterschiedliche Zeitcharakteristika.

Diese Zeitaspekte spielen beispielsweise eine Rolle bei der Abstimmung von Messwerten eines Zeitpunktes von verschiedenen, nicht synchronisierten Sensoren aufeinander (Sensor Fusion), der Auswertung verschiedener Messwerte zu Situationsinformationen oder der Ermittlung temporalen und kausalen Wissens.

Für die Repräsentation von Zeit kann zwischen quantitativen und qualitativen Ausdrucksformen unterschieden werden. Qualitative Ausdrucksformen verwenden keine Zahlenwerte für die Spezifikation von Restriktionen zwischen Variablen, sondern sind symbolischer Natur. Dies ist aber nicht immer ausreichend für das temporäre Schließen, da exakte quantitative Beschränkungen benötigt werden. Sowohl die quantitativen wie auch die qualitativen Ausdrucksformen basieren auf den Grundelementen Zeitpunkt und Intervall, absoluten und relativen Zeitangaben.

Absolute Zeitangaben sind dadurch gekennzeichnet, dass sie in der realen Zeit verankert sind. Während im alltäglichen Leben Zeitangeben in einer Datum-/Uhrzeit-Kombination gemacht werden, wird in IT-Systemen zur besseren Verarbeitung eine Zeitangabe in Form eines einzigen Zeitwerts, der mit einer Einheit versehen ist, gemacht. Da Zeit auch als Sensorinformation betrachtet werden kann, wird nicht von einem abstrakten Granularitätsintervall als kleinste darzustellende Zeiteinheit ausgegangen, sondern Zeit als physikalische Größe mit der Basiseinheit Sekunde oder einer daraus abgeleiteten Einheit modelliert. Damit wird die Einheit fester Bestandteil jeder Zeitangabe. Die verwendete Zeitskala hat ihren Nullpunkt in einem konkreten Zeit-

punkt, wodurch der Bezug zwischen dem Zeitwert und der Realzeit hergestellt wird. Realzeit kann mit Hilfe einer Uhr in einer bestimmten Granularität und einer bestimmten Genauigkeit gemessen werden. Ungenauigkeiten bei der Zeitmessung treten durch Abweichungen der Zeitskala von der tatsächlichen Zeitbasis und Fehler im Messprozess auf. **Relative Zeitangaben** beschreiben das Eintreffen von Ereignissen in Bezug auf ein anderes Ereignis.

Prinzipiell lässt sich Zeit als Zeitpunkt oder als Zeitintervall modellieren. **Zeitintervalle** erlauben die Repräsentation von Zeiträumen. Mit ihnen lassen sich Beginn, Dauer und Ende einer Aktion oder eines Zustands mit nur einem Datentyp beschreiben. Intervalloperationen ermöglichen Aussagen über die zeitliche Relation verschiedener Ereignisse wie beispielsweise deren Reihenfolge oder zeitliche Überschneidung. Intervalle können wie Zeitpunkte im absoluten oder einem relativen Zeitsystem angegeben werden. Ein **Zeitpunkt** kann als Intervall der Größe Null interpretiert werden. Welche Variante verwendet wird, hängt von der Forderung an die Genauigkeit und von der Qualität der Kontextinformationen ab. Ereignisse sind zeitlich punktuell, Aktionen, als eine Menge von Ereignissen verstanden, umfassen ein Zeitintervall und Zustände von Personen, Objekten oder Systemen haben eine unbestimmte zeitliche Ausdehnung.

Auf Zeitpunkten oder Zeitintervallen können verschiedene Operationen angewendet werden: Vergleichsoperationen und arithmetische Operationen. **Vergleichsoperationen** erlauben Aussagen über die Reihenfolge von Ereignissen und Aktionen. Vergleichsoperationen können sowohl auf Zeitpunkte wie auch auf Zeitintervalle angewendet werden. Vergleiche zwischen absoluten und relativen Zeitangaben sind jedoch nicht möglich. In [All83] wurden 13 Grundoperationen für die Festlegung von Vergleichsoperationen für Zeitintervalle und auch für Zeitpunkte als Zeitintervall der Größe Null identifiziert. Arithmetische Zeitoperationen dienen dazu, absolute Zeitangaben und relative Zeitangaben in Beziehung zu setzen, Granularitätsanpassungen vorzunehmen oder die Dauer von Aktionen, welche aus einer Menge von Ereignissen bestehen, zu berechnen

Eine ausführliche Analyse von verschiedenen Repräsentationsformen für Zeit ist in [Win93] zu finden. Andreas Winklhofer stellt darin fest, dass qualitative oder quantitative Repräsentation nicht ausreichend ist für temporales Schließen und entwickelt aus drei von ihnen (Intervallalgebra, temporales Constraint-Satisfaction-Problem und Mengen von Set-of-Possibility-Occurences) ein neues Modell, den Zeitkern. Das Ziel dieses Modells ist es, Beschränkungen der einzelnen Repräsentationsformen auszugleichen. Es erlaubt qualitative und quantitative Ausdrücke sowohl mit Zeitpunkten als auch mit Intervallen darzustellen. Das Zeitkern-Modell legt einen stärkeren Fokus auf arithmetische Operationen auf Zeitpunkten und Intervallen.

5.2.3.3 Sensorinformation

Sensorinformationen werden – wie der Name schon sagt – durch Sensoren ermittelt. Dabei unterscheidet man zwischen physischen und logischen Sensoren. Physische Sensoren ermitteln physikalische Faktoren, wie beispielsweise Temperatur, Windgeschwindigkeit, CO_2-Gehalt der Luft, Helligkeit oder auch Lokation. Logische oder Software-Sensoren ermitteln Faktoren wie beispielsweise CPU-Auslastung eines

Rechners, die Anzahl der offenen Verbindungen ins Internet oder Anzahl der angemeldeten Benutzer in einem System.

Der Einsatz von physischen Sensoren ist weit verbreitet, beispielsweise in der Robotik oder in der Automation bei der Prozesskontrolle. Grob unterschieden werden kann zwischen [SBG99]:

- optischen Sensoren (beispielsweise IR-, UV-Sensoren oder Kameras),
- Audio-Sensoren (beispielsweise Mikrophone),
- Bewegungssensoren (beispielsweise Infrarotsensoren),
- Positionssensoren (beispielsweise GPS oder Active Badge-Systeme),
- Bio-Sensoren (beispielsweise Puls- oder Blutmessgeräte) und
- einfachen Umgebungssensoren (beispielsweise Temperatur- oder Feuchtigkeitssensoren).

Für Sensorinformationen können unterschiedliche Qualitätsstufen unterschieden werden. Man spricht von einfachen oder Roh-Sensorinformationen, wenn die Daten direkt ohne Bearbeitung von Sensoren geliefert werden. Werden diese aggregiert, um beispielsweise die Daten an die benötigte Granularität von Raum oder Zeit anzupassen, spricht man von höherwertigen Sensordaten. Werden Sensordaten von mehreren, evtl. auch verschiedenartigen Sensoren zusammengefasst, um beispielsweise die Genauigkeit zu verbessern, spricht man von Sensor Fusion [BI97].

Es gibt nicht nur eine Methode für Sensor Fusion. Unter diesem Begriff verbergen sich eine große Menge von Problemen und verschiedenste mathematische Methoden. Die Probleme sind unter anderem die Extraktion relevanter Eigenschaften aus den Sensordaten, Mustererkennung oder inkompatible Datenformate. Um verschiedene Sensordaten zu verschmelzen, werden Inferenzmethoden, Schätzverfahren, Cluster-Methoden oder Verfahren des Soft Computing eingesetzt [BHM97]. Welches spezielle Sensor Fusion-Verfahren zur Anwendung kommt, hängt zum einen von der Art der verwendeten Sensoren und zum anderen von der Anwendung ab, welche die Daten verwendet.

5.2.3.4 Objekt- und Subjektinformationen

Der Begriff „Objekt“ wird sehr weit gefasst. Zu Objekten zählen beliebige Gegenstände einschließlich technischer Geräte aber auch virtuelle Objekte. Objekte lassen sich wie folgt charakterisieren:

- sie besitzen eine eindeutige Identifikation,
- sie existieren in Raum und Zeit,
- sie gehören einer eindeutigen Datenschutzgruppe an,
- sie besitzen beliebige objektspezifische Attribute sowie
- eine Menge von Beziehungen zu anderen Objekten oder Subjekten.

Optional können Objekte

- eine Vergangenheit (Historie) und
- einen Gültigkeitswert für ihre Attribute besitzen.

Die Modellierung von Subjekten - auch *User Modeling* genannt - ist ein sehr komplexes und umfangreiches Thema. Das zeigt sich auch daran, dass sich eigens die Arbeitsgemeinschaft „User Modeling Inc." [UMI] gebildet hat, die ein Journal veröffentlicht und verschiedene Konferenzen organisiert. Weiterführende Auseinandersetzungen mit dem Thema „User Modeling" finden sich bei A. Kobsa [Kob93, FK00], im Bereich Kontext bei Byun und Chevest [BC01], speziell für die Aspekte Sicherheit und Datenschutz bei J. Schreck [Sch00] sowie für benutzeradaptive Lehrdokumente bei C. Seeberg [See01].

Obwohl es aufgrund der Komplexität der Materie kein universelles Benutzermodell geben kann, haben sich doch einige Techniken zur Modellierung etabliert. Einige Kriterien zur Klassifikation stammen von Kobsa [Kob93]. Er unterscheidet zwischen

- Art der getroffenen Annahmen über den Benutzer (Ziele, Pläne, Präferenzen, Fähigkeiten, etc.),
- dem Grad der Integration des Benutzermodellerwerbs in den normalen Mensch-Maschine-Dialog (system- oder benutzergesteuerte),
- die verwendeten Techniken (primäre Erwerbsheuristiken oder Stereotyp-Ansatz),
- die Beobachtungsnähe der getroffenen Annahmen und
- der Sicherheit der getroffenen Annahmen.

Ein innerbetriebliches Anwendungsszenario, wie es dieser Arbeit zugrunde liegt (siehe Kapitel 2.2), ist das ideale Anwendungsgebiet für die Kombination des Stereotyp-Ansatz von Rich [Ric79] und Chin [Chi89] und der Definition von Präferenzen zu einem hierarchischen Ansatz, da die Organisationsstrukturen einfach auf Benutzergruppen abgebildet werden können. Der Stereotypansatz basiert auf drei Methoden: der Identifikation von Benutzeruntergruppen, der Identifikation von Schlüsselmerkmalen und die Repräsentation in hierarchischen Stereotypen.

Die Informationen über ein Subjekt können in drei Gruppen eingeteilt werden:

- *Angaben über die Person*

  Angaben zur Person können allgemeine Informationen wie Name, Adresse, Email-Adresse oder Geburtsdatum sein, Informationen über ihre Kenntnisse und Fähigkeiten, aber auch allgemeine Informationen über ihre Positionierung im Kontext (z.B. Rolle im Unternehmen, Aufenthaltsort, Status im Projekt). Das Merkmal dieser Angaben ist, dass sie durch eine Drittperson erhoben werden können.

- *Angaben über die Präferenzen einer Person*

  Unter Präferenzen werden persönliche Einstellungen verstanden, welche nur durch den Benutzer selbst erhoben werden können. Diese Angaben sind in

der Regel anwendungsabhängig. Sie umfassen unter anderem Ziele und Wünsche der Person. Beispiele dafür sind die Datenschutz-Einstellungen für das Web mit P3P [P3P01] oder die Benutzermodellierung in [See01]. In [See01] werden Informationen wie der Wissensstand, die Interessen oder beispielsweise die Sprache des Benutzers dazu verwendet, eine individuelle Aufbereitung von Lehrmaterialien zu ermöglichen.

- *Beobachtetes Wissen*

  Systeme wie forget-me-not [LF94] oder der remembrance agent [Rho97] beobachten das Verhalten der Personen und zeichnen es für eine spätere Analyse auf. Aus dem Verhalten in der Vergangenheit wird dann auf die Wünsche des Benutzers in der Zukunft geschlossen. Diese Art von Wissen über Systembenutzer wird in dieser Arbeit nicht weiter berücksichtigt.

Wie diese kurze Übersicht zeigt, ist der überwiegende Teil der benötigten Informationen über die Akteure anwendungsabhängig und nur ein kleiner Teil allgemeingültig. Aus diesem Grunde muss genauestens abgewogen werden, ob die anwendungsspezifischen Benutzerdaten in einem dezentralen Kontext-System zu verwalten sind, wie es beispielsweise in [Kob93] propagiert wird, oder zentral in der Anwendung selbst.

5.2.3.5 Temporales und kausales Wissen

Temporales und kausales Wissen basieren auf gespeicherten Ereignis-Informationen. Temporales Wissen betrachtet ausschließlich den zeitlichen Zusammenhang und besteht somit aus einer Menge von Ereignissen, die zeitlich geordnet sind. Kausales Wissen dagegen basiert auf der aktuellen Situation und den Beziehungen verschiedener Komponenten eines Systems zueinander. Beide können durch Ereignis-Muster spezifiziert werden (siehe Abschnitt 5.1).

Kausale und temporale Kontextinformationen sind beispielsweise früher durchgeführte Handlungen der betrachteten Benutzer. Sowohl bei temporaler wie auch kausaler Kontextinformation spielt die Vergangenheit eine zentrale Rolle. Während bei temporaler Kontextinformation beliebige zurückliegende Ereignisse betrachtet werden, beziehen sich kausale Kontextinformationen immer auf die Vergangenheit eines bestimmten Subjekts oder Objekts. Kausales Wissen stellt somit eine Untermenge des temporalen Wissens dar.

Protokolle

Eine spezielle Form von temporalem Wissen sind Protokolle. Protokolle beschreiben nicht-deterministische Ereignisfolgen aus temporal geordneten Ereignissen. Sie sind damit eine Untermenge der in Abschnitt 5.1.3 definierten Ereignis-Muster.

Die zentralen Konstrukte des in dieser Arbeit unterstützten Protokolls sind:

- *Subject*: initiiert ein Ereignis,
- *Object*: ist Ziel eines Ereignisses,
- *Event*: ein Ereignis besteht aus dem Subjekt, der Aktion und dem Zielobjekt

  event $\mapsto$ $ident_{subject}$ action $ident_{object}$

- *Protocol*: ein Protokoll besteht aus einer erlaubten Folge von Ereignissen

protocol	$\mapsto$	event	(Ereignis)
	$\mapsto$	protocol; protocol	(Sequenz)
	$\mapsto$	protocol | protocol	(Alternative)
	$\mapsto$	$protocol^*$	(beliebige Wiederholung)

5.2.3.6 Situationswissen

Unter Situationswissen wird Wissen verstanden, welches durch Interpretation von einfachen oder auch komplexen Kontextinformationen gewonnen wird. Es ist somit die Interpretation einer Kombination aus temporalem oder kausalem Wissen mit einfachem Objektwissen. Situationswissen betrachtet immer den aktuellen Zustand des Systems.

5.3 Eigenschaften von Kontextinformationen

Kontextinformationen besitzen bestimmte Eigenschaften, die bei der Modellierung und Implementierung berücksichtigt werden müssen. Dazu zählen der Bezug zu Zeit und Lokation sowie Gültigkeit, Unsicherheit und Unschärfe des an die Kontextvariable gebundenen Wertes.

5.3.1 Zeit und Lokation

Zeit und Lokation nehmen eine Sonderstellung unter den Kontextinformationen ein. Beide können als einfache Kontextinformationen betrachtet werden. Sie können voneinander abhängig sein, müssen es aber nicht. So ist eine exakte Zeitangabe immer nur in einem bestimmten Gebiet (Zeitzone) gültig, eine logische Lokation (z.B. „Raum A“) von der aktuellen Konfiguration des Raums. Relative Zeitangaben sind dagegen Lokations-unabhängig, wohingegen absolute Positionsangaben zeitunabhängig sind. Da diese Abhängigkeiten zwischen Raum und Zeit bei der Beschränkung auf ein Anwendungsgebiet sehr gering sind, werden sie in der Regel nicht weiter berücksichtigt.

Anders verhält es sich mit allen anderen Klassen von Kontextinformationen. Jede Kontextinformation ist direkt oder indirekt abhängig von Raum und Zeit. Ein gemessener Temperaturwert erhält erst durch die Kombination mit dem Zeitpunkt der

Messung und dem Ort des Temperatursensors seine Gültigkeit und seinen Aussagewert. Entsprechendes gilt für abgeleitete Informationen wie Situations-, temporales oder kausales Wissen.

5.3.2 Gültigkeit, Unsicherheit und Unschärfe

Durch die Abhängigkeit einer Kontextinformation von Zeit und Lokation, ist mit der stetigen Veränderung dieser beiden Größen jeder Kontextinformation eine Unsicherheit inhärent. Je größer die Entfernung des betrachteten Objekts zu seinem Messpunkt ist, je mehr Zeit vergeht, umso unsicherer wird der gemessene Wert.

Da einfache Kontextinformationen mit Sensoren gemessene Werte sind, sind diese mit (marginalen) Messfehlern behaftet. Aber auch logische Sensoren sind mit Messfehlern behaftet und liefern gerundete, diskrete Werte.

Durch diese Tatsachen ist jeder Kontextwert mit einem Unsicherheitsfaktor behaftet. Bewegt sich der Unsicherheitsfaktor unterhalb eines bestimmten (festzulegenden) Schwellwerts, so kann er bei der Modellierung des Systems außer Acht gelassen werden.

5.4 Spezifikation von Kontextinformationen

Die Modellierung der Kontextinformationen und die Spezifikation in einem allgemeinen Format sind von entscheidender Bedeutung für die Verarbeitung, die Speicherung sowie für das Wiederauffinden und Abfragen von Kontextinformationen. Nur mit Hilfe einer Spezifikationssprache ist es einer kontextabhängigen Anwendung möglich, die benötigten Kontextinformationen auf einer abstrakten Ebene zu beschreiben. Eine Beschreibungssprache für Kontextinformationen muss in ein Objektmodell abbildbar sein, eine Abbildung auf Ereignisanforderungen (engl. event subscription) des Ereignissystems ermöglichen und die Spezifikation einer Abfragesprache unterstützen.

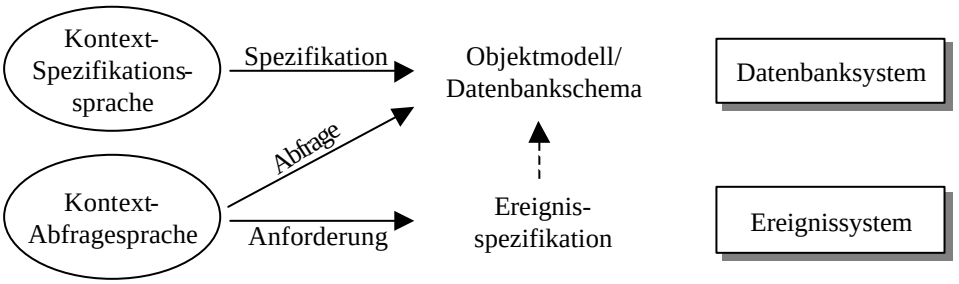

Abbildung 23 Kontextspezifikation und -abfrage

Der Vorteil eines Objektmodells liegt in der Trennung von Wissen, Verwendung und Akquisition der Daten. Durch die Verwendung unterschiedlicher Objektmodelle können gleiche Daten auf unterschiedliche Art und Weise für verschiedene Anwendungen aufbereitet werden. Das Datenmodell muss daher in der Lage sein, beliebige

Entitäten wie Personen, Objekte, Lokationen, Rechte oder Zertifikate zu beschreiben, aber auch Entitätsattribute und Relationen.

Um Interoperabilität mit anderen Systemen zu erreichen, aber auch um das Verständnis für die Semantik von Kontextinformationen zu erhöhen, sollten vorhandene Standards für die Spezifikation von Datenformaten verwendet werden. Nur durch die Verwendung von Vokabularen, die eine breite Akzeptanz finden, wird dieses Ziel erreicht. Das hat auch den Vorteil, dass durch die Wiederverwendbarkeit der Aufwand für die Datenerhebung reduziert wird.

Wie in den vorangehenden Kapiteln deutlich wurde, kann nicht *die* Kontextinformation existieren, da Kontextinformation von unterschiedlichster Natur sein kann. Bis auf einige Grundeigenschaften, sind Kontextinformationen, und vor allem die Beziehungen dazwischen, vom jeweiligen Anwendungsgebiet abhängig. Dies gilt besonders für höherwertige Informationen wie Situationswissen oder kausales und temporales Wissen, da dieses – da es sich um Interpretation von Daten handelt - in der Regel in einer speziellen Ausprägung spezifisch für eine Anwendungen ermittelt wird.

Als geeignete Mittel zum Erreichen der angestrebten Ziele für eine Spezifikationssprache für Kontextinformationen haben sich XML (eXtensible Mark-up Language) und RDF (Resource Description Framework) herausgestellt. Beide Ansätze, XML und RDF, erfüllen die Anforderungen an eine Spezifikationssprache für Kontextinformationen. Für beide existieren Editoren, Parser sowie spezielle Datenbanken zur Speicherung und Abfragesprachen für das Wiederauffinden. Obwohl XML bezüglich dieser Punkte einen klaren Vorteil gegenüber RDF hat, bietet RDF die besseren Möglichkeiten zur Beschreibung der Beziehungen zwischen den Kontextinformationen.

An Beispiel der Kontextinformationen „Lokation“, „Zeit”, „Objekt” und „Subjekt” werden verschiedene Ansätze zur Kodierung vorgestellt und der Einsatz von XML und RDF/XML exemplarisch demonstriert.

5.4.1 XML und RDF/XML als Spezifikationssprachen

XML [BPS+00] stellt sich immer mehr als die bevorzugte Kodiermethode für den Austausch von Daten heraus. Der Vorteil von XML ist die uneingeschränkte Erweiterbarkeit, die Möglichkeit der kompletten Trennung von Inhalt und Formatierung und die einfach zu erlernende Syntax. Die Syntax einer auf XML basierenden Sprache kann in einer so genannten Document Type Definition (DTD) [DTD98] festgelegt werden beziehungsweise deren Nachfolger XML Schema [TBM+01].

Die XML DTD stellt eine an die EBNF (Extended Backus Nauer Form) angelehnte reguläre Sprache zur Formulierung der Syntax einer XML-Sprache zur Verfügung. Diese textuelle Notation zur Spezifikation von XML-Textstrukturen wurde von SGML (Standard Generalized Markup Language) übernommen. Zur Verwendung in XML wurde lediglich der Sprachumfang reduziert und um einige nicht benötigte Konstrukte bereinigt. Daher sind die DTDs sehr stark dokumentenorientiert. Prinzipiell stellt die DTD jedoch eine eigenständige, vom XML-Dokument losgelöste Informationseinheit dar. Ihre Rolle entspricht der einer Klasse in der objektorientierten Programmierung. Sie beinhaltet die erlaubten Elemente sowie die hierarchische Gliederung der Elemente untereinander. DTDs verwenden eine eigene Syntax, welche

sich von der XML Syntax erheblich unterscheidet. Die Festlegung von Datentypen für bestimmte XML Elemente ist nicht möglich.

XML Schema ist eine neue Sprache, welche durch das World Wide Web Consortium (W3C) für die Spezifikation von Inhalt und Struktur von Dokumententypen in XML entwickelt wird. XML Schema wurde erst im Mai 2001 als so genannte W3C Recommendation verabschiedet [BM01, TBM+01]. XML Schema verfolgt den gleichen Zweck wie DTD, bietet aber mehr Flexibilität und Möglichkeiten zur Beschreibung von Inhalten. Es markiert den Übergang von präsentationsorientierten Strukturen hin zu Datenstrukturen.

Einige der Vorteile, die XML Schema gegenüber XML DTD bietet sind:

- *Erweiterte Datentypunterstützung:*

  DTDs erlauben für Elemente nur vier Inhaltsmodelle: child, elements, PCDATA, mixed content sowie das leere Inhaltsmodell EMPTY. Für die Attributdefinition stehen genau genommen nur Zeichenketten-artige Datentypen zur Verfügung. Die Modellierung benötigter Datentypen durch anwenderdefinierte Aufzählungstypen ist zeitaufwendig und fehlerträchtig.

- *Verbesserte Strukturierungsunterstützung*

  XML Schema bietet beispielsweise bessere Möglichkeiten, Auftrittshäufigkeiten zu spezifizieren.

- *Unterstützung von Wiederverwendbarkeit:*

  Während Elementstrukturen (zumindest) innerhalb der definierenden DTD beliebig wieder verwendet werden können, sind Attribute immer an das umgebende Element gebunden. Eine Nutzung in anderen als der definierenden DTD ist nicht vorgesehen.

- *Flexibles Typsystem:*

  Das in DTD angebotene Typsystem kann durch den Anwender nicht erweitert werden.

- *Unterstützung von Namensräumen:*

  In DTDs können keine Namensräume angegeben werden.

- *Erweiterte Referenzierungsmechanismen:*

  Die in DTDs angebotenen ID-IDREF(S)-Verknüpfungen sind ausschließlich Dokument-lokal möglich und gestatten keine Differenzierung hinsichtlich der Semantik des eindeutig identifizierten oder referenzierten Elements.

Mit XML Schema wurde eine Möglichkeit zur Formulierung von (eigenen) XML-Sprachen geschaffen. XML Schema bietet somit eine gute Möglichkeit Syntax zu beschreiben. Die Beschreibung der Semantik ist in XML Schema nur begrenzt möglich. Diese spielt aber beim Austausch und der anschließenden Auswertung von Daten über Domänengrenzen hinweg eine wichtige Rolle.

Eine auf XML basierende Anwendung zur Beschreibung von Semantik ist das *Resource Description Framework (RDF)* [RDF]. Mit RDF stellt das W3C eine Infra-

struktur zur Kodierung, dem Austausch und der Wiederverwendung von strukturierten Metadaten zur Verfügung. Eine Ressource wird hierbei als beliebiges Objekt definiert, das durch einen URI (Unified Resource Identifier) eindeutig identifiziert werden kann. Jede dieser Ressourcen lässt sich durch eine oder mehrere Eigenschaften beschreiben. RDF stellt lediglich ein Framework zum Austausch von Metadaten bereit, sowie ein Model und eine Syntax um Eigenschaften von Ressourcen zu spezifizieren.

Zur Festlegung der zu spezifizierenden Eigenschaften und deren Beziehungen zueinander stellt RDF den Mechanismus *RDF Schema* [BG01] zur Verfügung. Was XML Schema für XML ist, ist RDF Schema für RDF. Während XML Schema Beschränkungen bezüglich Inhalt und Struktur von XML Dokumenten festlegt, beschreibt RDF Schema Eigenschaften von RDF-Beschreibungen, deren Bedeutungen und Beziehungen.

Im Gegensatz zu XML Schema, das rein syntaktische Bedingungen für die Struktur eines XML Dokuments spezifizieren kann, stellt RDF Schema in begrenztem Umfang Informationen über die Interpretation einer Aussage in einem RDF Datenmodell zur Verfügung, mit anderen Worten, es wird eine sehr eingeschränkte Ontologie vordefiniert.

Trotz der unterschiedlichen Schwerpunkte von XML Schema und RDF Schema überschneidet sich ihre Funktionalität, was bei gleichzeitiger Anwendung beider Ansätze zu Problemen führen kann. Konflikte treten beispielsweise zwischen dem RDF Schema-Element „range“ und dem XML Schema-Element „type“, dem RDF Schema-Element „domain“ und der XML Schema Definition von „type“ und „element“ sowie dem RDF Schema-Elemente „comment“ und dem XML Schema-Element „annotations“, auf. Solange keine Parser existieren, welche die Konsistenz und die Verträglichkeit beider Beschreibungen überprüfen können, gibt es nur die Möglichkeiten, entweder nur einen dieser Ansätze zu verwenden oder aber eine eindeutige Trennungslinie für die Zuständigkeiten zu ziehen. Wie dies erfolgt, bleibt dem jeweiligen Anwender überlassen.

5.4.2 Lokation

Die Repräsentation von Lokationsinformationen kann in unterschiedlicher Form erfolgen. Die gängigsten Formate heute sind binär, Text- oder XML-basiert. Das Third Generation Partnership Project (3GPP), das eine Location Service Architecture für GSM und UMTS definierte, spezifizierte ein binäres Format, um Lokationsinformationen zu verschicken [3GP00]. GPS-Empfänger liefern die Daten nach NMEA 0183-Standard im ASCII-Format, ebenso der DNS-LOC [DVG+96] die Ortsinformationen von Rechnersystemen. Erste und sehr junge Ansätze, welche XML mit DTD zur Beschreibung von Positionsangaben verwenden, sind beispielsweise der Internet-Draft *Spatial Location Payload* (SLP)[KT01], die *Geospatial-eXtensible Markup Language* (G-XML) des Ministry of Economy, Trade and Industry Japan [GXML01 oder die *NaVigation Markup Language* (NVML) von Fujitsu [NVML99].

Der vielversprechendste Ansatz ist die vom OpenGIS Consortium (OCG) stammende *Geographical Markup Language* (GML) [CCD+01]. Die erste Version wurde im Mai 2000 verabschiedet, Version 2.0 im April 2001. Die Spezifikation von

GML lehnt sich stark an RDF an; So wurde das RDF-Modell Type-Property übernommen. GML ist jedoch nicht als Ontologie zu betrachten. Für die ersten Versionen von GML wurden eine DTD sowie ein RDF Schema spezifiziert. Ab Version 2.0 wurde die DTD durch ein XML Schema ersetzt.

GML ist eine XML-Anwendung für die Speicherung und den Austausch von geographischen Daten im Internet. Geodaten in GML besitzen einfache Geometrien und optional weitere beschreibende Eigenschaften. Geometrien raumbezogener Objekte werden durch SimpleFeatures repräsentiert, ebenfalls eine Spezifikation des OCG. Durch die Verwendung von SimpleFeature, können zwei-dimensionale Daten jedoch keine drei-dimensionalen Daten spezifiziert werden. SimpleFeature beschreibt ein geographisches Objekt als Punkt, Kurve, Polygon, Oberfläche oder eine Kollektion davon. Da GML kein räumliches Bezugssystem (engl. spatial reference system; SRS) vorgibt, ist für alle geographischen Angaben das SRS zu spezifizieren. GML definiert ausschließlich die Kodierung von geographischen Objekten und deren Beziehungen, jedoch nicht die visuelle Darstellung auf einem Bildschirm oder in einer Landkarte.

Alle genannten Ansätze, einschließlich GML, zur Kodierung von Lokationsdaten basieren auf verschiedenen geographischen Modellen. Symbolische Modelle werden nicht unterstützt. Aus diesem Grunde ist es notwendig, ein eigenes Spezifikationsschema für das Hierarchien-Modell zu entwerfen. Um jedoch den Weg für die Verbindung von symbolischen Modellen mit geographischen Modellen nicht zu versperren, muss auf Interoperabilität geachtet werden. Das ist beispielsweise dann notwendig, wenn über ein GPS-System geodätische Lokationsangaben ermittelt werden und diese in das entsprechende symbolische Modell transferiert werden sollen.

GML bietet die Möglichkeit, anwendungsspezifische Erweiterungen zu spezifizieren. Damit ist es möglich, ein symbolisches Modell als GML Erweiterung festzulegen. Als Beispiel eine Spezifikation eines Gebäudes und eines Raums, das neben GML das vCard-Vokabular (siehe auch Kapitel 5.4.5) verwendet, ist im Folgenden zu finden:

```
<Buiding ID ="TUD-W7">
    <description>university of technology</description>
    <NoFloors>4</NoFloors>
    <NoRooms>56</NoRooms>
    <vCard:ADR parseType="Resource">
        <vCard:Street> Wilhelminenstrasse 7 </vCard:Street>
        <vCard:Locality> Darmstadt </vCard:Locality>
        <vCard:Pcode> 64283</vCard:Pcode>
        <vCard:Country> Germany </vCard:Country>
    </vCard:ADR>
    <gml:extentOf>
        <gml:Polygon srsName="epsg:27354">
            <gml:outerBoundaryIs>
                <gml:LinearRing>
                    <gml:coordinates>
                            491888.999999459,5458045.99963358, ...
                    </gml:coordinates>
                </gml:LinearRing >
            </gml:outerBoundaryIs>
```

```
        </gml:Polygon>
    </gml:extentOf>
</Buiding>

<Room ID="R55">
    <description>library 4th floor</description>
    <Buiding>TUD-W7</Buiding>
    <Floor>4</Floor>
    <Usage>public</Usage>
    <gml:locationOf>
        <gml:Point srsName = „ … „>
            <gml:coordinates>
                55661.1454, …
            </gml:coordinates>
        </gml:Point>
    </gml:locationOf>
</Room>
```

GML stellt einige räumliche Attribute (engl. properties) bereit, wie beispielsweise „centerOf“, „positionOf“ oder „locationOf“ für punktuelle Raumangaben, „extendOf“ und „coverageOf“ für Polygone. Jedoch ist zu beachten, dass GML sowohl „location“ wie auch „position“ als Punktangabe spezifiziert. Der Unterschied zwischen den beiden Bezeichnungen – und damit die Semantik – wird in der Spezifikation nicht angegeben.

5.4.3 Zeit

Der internationale Standard zur Kodierung von Zeit ist ISO 8601 [ISO8601]. Auf diesen wird auch in verschiedenen Metadaten-Spezifikationssprachen, wie beispielsweise iCalendar [DS98] oder Dublin Core [DC, CDCE97, Hil01], für die Beschreibung von Zeit und Zeitintervallen zurückgegriffen.

In Dublin Core kann Zeit in W3C-DTF oder DCMI Period [Cox00] kodiert werden. Das W3C Date and Time Format (DTF) spezifiziert einige Zeit- und Datumsprofile nach ISO 8601 und versucht somit die Missverständnisse bei der Zeitspezifikation zu reduzieren [DTF97F].

W3C-DTF sieht sechs Stufen der Granularität vor:

- Jahr: YYYY (Bsp. 1997)
- Jahr und Monat: YYYY-MM (Bsp.1997-07)
- Vollständiges Datum: YYYY-MM-DD (Bsp.1997-07-16)
- Vollst. Datum plus Stunden und Minuten:
  YYYY-MM-DDThh:mmTZD (Bsp.1997-07-16T19:20+01:00)
- Vollst. Datum plus Stunden, Minuten und Sekunden:
  YYYY-MM-DDThh:mm:ssTZD (Bsp.1997-07-16T19:20:30+01:00)
- Vollst. Datum plus Stunden,Minuten, Sekunden und Millisekunden:
  YYYY-MM-DDThh:mm:ss.sTZD (Bsp.1997-07-16T19:20:30.45+01:00)

Die Angabe der Zeitzone erfolgt entweder explizit („Z“; UTC) oder durch Angabe der Zeitdifferenz „+hh:mm” zu UTC (Universal Time Coordinated).

DCMI Period erlaubt die Spezifikation von Zeitintervallen in Textformat. Das Intervall wird festgelegt durch einen Start- und einen Endpunkt und der Angabe der Zeitkodierung. Ein Beispiel eines Zeitintervalls in DCMI Period mit W3C-DTF-Kodierung:

```
<Period name="Perth International Arts Festival 2000">
    <start>2000-01-26</start>
    <end>2000-02-20</end>
</Period>
```

ICalendar [DS98] wurde als ein allgemeines Format für den Austausch von Kalender- und Planungsinformationen entwickelt. ICalendar verwendet wie Dublin Core das ISO 8601 Format zur Kodierung von Zeit:

```
<ical:DTSTART>
    <ical:DATE-TIME>
        <ical:TZID rdf:resource="#CET"/>
        <rdf:value>20010525T090000</rdf:value>
        <util:hour>09</util:hour>
        <util:minute>00</util:minute>
    </ical:DATE-TIME>
</ical:DTSTART>
```

5.4.4 Objekt

Zur Beschreibung einer bestimmten Klasse von Objekten, den Geräten, wurden vom W3C die Composite Capability/Preference Profiles (CC/PP) spezifiziert [KRW+01]. Wie der Name besagt, wurde CC/PP entwickelt, um die Leistungsfähigkeit von Geräten und Präferenzen von Benutzern zu spezifizieren, mit dem Ziel, die automatisierte Aushandlung von Inhalten im WWW durch Benutzeragenten zu verbessern. Zu den Leistungsmerkmalen der Geräte zählen beispielsweise Bildschirmauflösung, zur Verfügung stehender Speicher, Lautsprecher, oder vorhandene Bandbreite. Zu Benutzer-Präferenzen gehören Privacy-Einstellungen nach P3P [P3P01], die bevorzugte Sprache, Audio-Ausgabe oder Cookies.

Die RDF-Anwendung CC/PP ist sowohl für Software- wie auch Hardware-Komponenten konzipiert. Das zugrunde liegende Datenmodell ist eine Sammlung von Tabellen, wobei jede Tabelle eine Sammlung von RDF-Statements mit einfachen, atomaren Eigenschaften ist. Ein Profil besteht in der Regel aus einer Hardware-Spezifikation, einer Software-Spezifikation und einer Browser-Spezifikation. Zur Verdeutlichung sei hier das Beispiel einer Hardware-Spezifikation für einen „Nokia PDA” in CC/PP angegeben:

```
<rdf:Description about="HardwarePlatform">
    <prf:Defaults
        Vendor="Nokia"
        Model="2160"
```

```
        Type="PDA"
        ScreenSize="800x600x24"
        CPU="PPC"
        Keyboard="Yes"
        Memory="16mB"
        Bluetooth="YES"
        Speaker="Yes" />
    <prf:Modifications
        Memory="32mB" />
</rdf:Description>
```

Obwohl CC/PP darauf ausgerichtet ist, die Anpassung von Diensten und Benutzungsoberflächen im WWW an Endgeräte und Benutzer-Präferenzen zu erleichtern, ist es möglich, beliebige Objekte damit zu spezifizieren.

Einer der ältesten und inzwischen fest etablierten Ansätze zur Beschreibung von unterschiedlichsten Internet-Ressourcen ist Dublin Core (DC). Die Dublin Core Initiative wurde im Oktober 1994 auf der zweiten World Wide Web Konferenz ins Leben gerufen. Das Ziel war es, eine semantische Grundlage für Metadaten sowie für ein Konkordanz-Format zu schaffen, das die Zusammenführung von Ressourcen mit unterschiedlichen Dateiformaten und ein einheitliche Suche in solchen Pools ermöglicht. Museumsfachleute, Bibliothekare, Vertreter der öffentlichen Verwaltung, Vertreter kommerzieller Unternehmen, Informatiker und Netzwerkspezialisten erarbeiteten auf mehreren internationalen Metadaten-Workshops die semantische Definition eines Kernsatzes von 15 Elementen, das so genannte Dublin Core Element Set [CDCE97], das zur vorwiegend inhaltlichen und formalbibliographischen Beschreibung digitaler und digitalisierter Ressourcen angewandt werden kann, um diese einer verbesserten Indexierung und einem gezieltem Retrieval zugänglich zu machen. Die Bezeichnung „Kern“ weist schon darauf hin, dass Erweiterungen durch Hinzufügen neuer Elemente, entsprechend den Bedürfnissen bestimmter Anwendergruppen, möglich sind.

Dublin Core gibt keine Sprache zur Kodierung vor. Da DC hauptsächlich Anwendung bei der Beschreibung von HTML-Ressourcen findet, sind weit verbreitete Kodierungarten META- und LINK-Tags in HTML [Kun99]. Eine weitere, nahe liegende Kodierungsform für DC ist RDF [KS01], da die Entwicklung von RDF von DC und dem Warwick Framework [LLD96], einer Architektur, welche die Beschreibung von Ressourcen durch verschiedene Metadatensätze ermöglicht, stark beeinflusst wurde. Zur Verdeutlichung ein Beispiel für Dublin Core-Metadaten für Bücher in RDF/XML-Kodierung aus dem Sm@rtLibrary-Projekt [MHR01, SLP]:

```
<rdf:RDF xmlns:="http://purl.org/dublin-core#"
    xmlns:rdf="http://www.w3.org/TR/WD-rdf-syntax#"
    xmlns:rdfs="http://www.w3.org/TR/WD-rdf-schema#"
    xmlns:dct="http://purl.org/dublin-core/types#">
    <dct:Book>
        <Title> Sensors in Intelligent Buildings </title>
        <Creator> Gassmann O.</Creator>
        <Creator> Meixner H.</Creator>
        <Description> Microelectronics have become…</ Description >
        <Language>EN</Language>
```

```
        <Publisher> Wiley-VCH </Publisher>
        <Date>2001</Date>
        <Identifier scheme="ISBN">3-527-29557-7</Identifier>
        <Coverage>
            <rdf:value>shelves=12; shelf=2</rdf:value>
        </Coverage>
    </dct:Book>
</rdf:RDF>
```

Das Dublin Core Element *coverage* dient der Festlegung zeitlicher Gültigkeit oder der räumlichen Bestimmung (z.B. geographische Koordinaten) der Ressource. Dieses Element ist nicht vollständig spezifiziert, so dass erheblicher Spielraum für die Angabe der Lokation besteht. Es werden etablierte Standards (DCMI Point, DCMI Box, ISO 3199, TGN) für die Angabe von Raumkoordinaten empfohlen, aber nicht vorgeschrieben [CDCE97]. Dieses DC-Element ist daher als im Experimentierstadium zu betrachten und nur mit Vorsicht zu verwenden.

5.4.5 Subjekt

Ein Standard für allgemeine Benutzerdaten ist das Format einer elektronischen Visitenkarte, der vCard, eine Spezifikation des versit-Konsortiums. Die Spezifikation von vCard in XML/DTD ist in [DH98] zu finden, in RDF/XML in [Ian01]. Zur Verdeutlichung ein Beispiel einer vCard-Kodierung in RDF/XML:

```
…
    <vCard:FN>Hans Müller</vcard:FN>
    <vCard:N rdf:parseType="Resource">
        <vCard:Family>Müller</vcard:Family>
        <vCard:Given>Hans </vcard:Given>
        <vCard:Prefix>Dipl.-Inf.</vcard:Prefix>
    </vCard:N>
    <vCard:TITLE>
        <rdf:seq>
            <rdf:li> Principal Research Scientist </rdf:li>
            <rdf:li> Visiting Professor </rdf:li>
        </rdf:seq>
    </vCard:TITLE>
    <vCard:EMAIL>hans@mueller.de</vcard:EMAIL>
    <vCard:BDAY>1966-03-04</vcard:BDAY>
    <vCard:ADR parseType="Resource">
        <vCard:Street> Lingusterweg 7 </vCard:Street>
        <vCard:Locality> Darmstadt </vCard:Locality>
        <vCard:Pcode> 65432</vCard:Pcode>
        <vCard:Country> Germany </vCard:Country>
    </vCard:ADR>
    <vCard:ROLE>
        <rdf:bag>
            <rdf:li> Programmer </rdf:li>
            <rdf:li> Administrator </rdf:li>
        </rdf:bag>
```

```
    </vCard:ROLE>
…
```

5.5 Abfrage von Kontextinformationen

Jede kontextabhängige Anwendung muss in der Lage sein, die benötigte Kontextinformation beim Kontextsystem als Ereignis anzufordern oder abzufragen. Dazu wird eine geeignete Abfragesprache (engl. context query language) benötigt. Wird zur Spezifikation von Kontextinformation XML oder RDF/XML verwendet, so liegt die Verwendung einer XML oder RDF Query Language als Abfragesprache für Kontextinformationen nahe. Im Folgenden werden verschiedene Abfragesprachen vorgestellt und auf die Tauglichkeit als Context Query Language untersucht.

5.5.1 XML Abfragesprachen

Eine Untergruppe des W3C, die XML Query Working Group, ist damit beschäftigt, ein Datenmodell für XML Dokumente zu schaffen und eine Anzahl von notwendigen Operationen in der Query Algebra bereitzustellen. Diese Spezifikationen bilden die Grundlage für die Festlegung einer präzisen Semantik einer XML Abfragesprache.

Diese Working Group legte verschiedene Anforderungen an eine Abfragesprache fest [CFM+00]. So soll eine XML-Abfragesprache beispielsweise:

- auf eine intuitive Syntax aufbauen, die leicht zu lesen und zu schreiben ist,
- eine transparente Darstellung der Syntax in XML erlauben,
- deklarativ sein, um den Umgang zu vereinfachen,
- standardisierte Fehlerbedingungen festlegen, die während der Ausführung einer Anfrage auftreten können (z.B. Ausführungsfehler innerhalb von Ausdrücken),
- Operationen spezifizieren, die für alle möglichen Datentypen des Datenmodells gültig sind und
- Aggregationen von Daten erlauben.

Da die definierten Anforderungen sehr hoch gesteckt sind, existiert bis heute noch kein etablierter Standard für eine XML Query Language. Doch hat die Working Group durch die Entwicklung eines Datenmodells und einer im Entwurf vorhandenen Algebra die Grundlagen dafür geschaffen.

Im Folgenden werden die wichtigsten XML-Abfragesprachen kurz beschrieben, die noch um die Durchsetzung als etablierter Standard kämpfen. Weitere XML Abfragesprachen sind beispielsweise XSL, XML-GL, XQuery, XMAS und Quilt, welche aber nicht weiter betrachtet werden, da sie nur geringe Bedeutung besitzen oder wenig Verbreitung fanden.

Lorel

Lorel (Lightweight Object Repository Language) [AQM+97] wurde an der Stanford University als Abfragesprache für das Lore (Lightweight Object Repository) Database Management System [LORE] entwickelt. Das Ziel des Lore-Projekts war es, ein System zu schaffen, welches die effiziente Speicherung und Abfrage von semi-strukturierten Daten gestattet. Die Entwicklung von Lore und somit auch von Lorel wurde an die Merkmale von XML angepasst.

Anfragen in Lorel entsprechen der Form „*select ... from ... where ...*“. und zeichnen sich durch Benutzerfreundlichkeit aus. Lorel verfügt über starke Mechanismen für Pfadausdrücke und unterstützt Vergleichsbedingungen, universelle Prädikate und den Quantor „*for all*”. Als Aggregatsfunktionen werden Minimum, Maximum, Summe, Anzahl und Durchschnitt angeboten.

Lorel gehört zu den mächtigeren Abfragesprachen. Ursprünglich als SQL-Sprache entwickelt, wurde Lorel zu einem späteren Zeitpunkt um die XML-Funktionalität erweitert. Die Sprache ist deklarativ und ausdrucksstark. Ein Nachteil ist, dass sie Xpointer und XML Schema nicht unterstützt.

XML-QL

XML-QL [XMLQL98] wurde von AT&T entwickelt und ist frei verfügbar. XML-QL ist eng an SQL angelehnt. Anfragen genügen der Form „*where ... construct ...*“. Die *where*-Klausel bestimmt, was zu selektieren ist. Sie nutzt dabei Element Pattern, um die Daten bzw. die Elemente des XML Dokumentes zu vergleichen. Die *construct*-Klausel spezifiziert die Formatierung der Ausgabe. Eine Verschachtelung der Where-Construct-Klauseln ist möglich.

XML-Ql ist weniger mächtig als Lorel. So wird – im Gegensatz zu Lorel - kein universeller Quantor, keine Negation und keine Aggregationsfunktion unterstützt. Obwohl durch das Fehlen von komplexeren Konstruktionen die Abfragen unübersichtlich und sehr lang werden, hat sich XML-QL besser als Lorel durchgesetzt.

XQL

XQL [RLS98] ist eine dokumentenorientierte Abfragesprache. Es ist eine Erweiterung von XSL um boolesche Logik, Filter und Indexierung. Ursprünglich für das „Document Processing” entwickelt, war das Ziel, verschiedenartige Präsentationen eines Dokuments zu ermöglichen. Mit Hilfe von XQL ist es möglich, Elemente und Texte eines Dokuments zu selektieren und zu filtern. Die Sprache ist sehr einfach und kompakt und besitzt daher auch nicht die Leistungsstärke der anderen Abfragesprachen. XQL besitzt im Gegensatz zu den anderen Abfragesprachen kein eigenes Datenmodell, sondern baut auf die implizite Baumstruktur von XML auf. Joins sind nicht möglich. Es gibt keine extra Anweisungen für die Ausgabe und Verschachtelungen sind auch nicht möglich.

Neben XML-QL gehört XQL zu den bekanntesten Abfragesprachen. Grund dafür ist wohl auch, dass die Microsoft Corporation bei der Entwicklung beteiligt war. Es existieren einige Implementierungen, beispielsweise die XQL Engine vom Fraunhofer Institut IPSI [XQLE], XQEngine der Fatdog Software [XQEngine] oder Tamino von der Software AG [Tamino].

Vergleich

Wie ein Vergleich der Abfragesprachen bezüglich der vom W3C aufgestellten Anforderungen an Abfragesprachen zeigt, ist keine der Sprachen in der Lage, alle Anforderungen zu erfüllen [BC00]. XQL erlaubt keine Verschachtelung, XML-QL keine Negation oder Aggregation. Lorel erfüllt die meisten Anforderungen, doch ist es eine auf das System Lore abgestimmte Sprache.

	Lorel	***XML-QL***	***XQL***
Joins	Ja	Ja	Nein
Partielle Pfadangabe	Ja	Ja	Ja
Existentieller Quantor	Ja	Ja	Ja
Universeller Quantor	Ja	Nein	Ja
Negation	Ja	Nein	Ja
Aggregation	Ja	Nein	Teilweise
Verschachtelte Abfragen	Ja	Ja	Nein
Mengenoperationen	Ja	Teilweise	Ja
RDF Unterstützung	Nein	Nein	Nein

Tabelle 1 Vergleich von XML-Abfragesprachen

Um ein System auf eine XML-Abfragesprache aufbauen zu können, spielt neben der von der Sprache gebotenen Funktionalität auch die Verfügbarkeit von Implementierungen und unterstützte Datenbanksysteme eine wichtige Rolle.

5.5.2 RDF Abfragesprachen

RDF-Abfragesprachen sind bisher nur wenig untersucht worden, wodurch es nur wenig nennenswerte Resultate gibt. Es existieren einige Vorschläge, doch noch keine Standards.

Da RDF auf einem anderen Datenmodell als XML basiert, werden auch andere Anforderungen an eine RDF-Abfragesprache (engl. RDF Query Language) gestellt. In [DBS+98] wird gefordert:

- ein Repository zur Speicherung der RDF-Daten muss die Ausdrucksstärke des RDF-Datenmodells unterstützen. Da das Datenmodell von RDF sehr stark an objektorientierte und framebasierte Systeme angelegt ist, sollten entsprechende Konzepte, z.B. Klassenhierarchien und Vererbung, unterstützt werden.
- die Abfragesprache muss von der Kodierung von RDF abstrahieren,
- die Abfragesprache muss unterschiedliche Qualitätsstufen unterstützen, von einfachen Wertabfragen bis hin zu Datalog-ähnlichen Abfragen,
- Zusammenfassung von Klassen und Properties erlauben,
- Klassifikation von Ressourcen soll möglich sein und

- grundlegende Inferenzmöglichkeiten bereitstellen.

Es existieren verschiedene Ansätze für RDF-Abfragesprachen, die in zwei große Kategorien eingeteilt werden können: die SQL-ähnlichen Abfragesprachen und die deklarativen. Einige Vertreter dieser Klassen werden im Folgenden kurz vorgestellt.

RDF Query Specification

Einer der ersten Versuche einer RDF Query Language stammt von Sundaresan von IBM. Er entwickelte 1998 das auf Java basierende Tool „RDF for XML" [IBMAW]. Dieses Tool erlaubt RDF-Objekte zu erzeugen, die in RDF/XML kodiert werden und stellt eine Schnittstelle bereit, die es erlaubt, verschiedene Operationen auf diesen Objekten auszuführen. Zusammen mit Malhotra [SM98] legte Sundaresan die RDF Query Specification fest, die das Abfragen von verschiedenen Ressourcen ermöglicht.

Die Merkmale von RDF Query sind:

- SQL-ähnliche Syntax mit Select-from-Klauseln und Spezifikation von Bedingungen über ein Condition-Tag,
- Pfadausdrücke,
- Vereinigung und Schnittmenge,
- Gruppierung und Sortierung sowie
- die Unterstützung des existentiellen Quantors.

Die Sprache stellt alle Konstrukte zur Verfügung, um strukturierte Dokumente, deren Schema, Ressourcen und Attribute bekannt sind, abzufragen. Damit ähnelt sie jedoch sehr stark den XML-basierten Abfragesprachen und erfüllt nicht die Anforderungen, die an eine RDF-Abfragesprache gestellt werden. Dass dieses Manko sehr bald erkannt wurde, sieht man daran, dass dieses Tool wie auch die Sprache keine weitere Unterstützung erfuhr und die Entwicklung eingestellt wurde.

Query and Inference Service for RDF

Ein deklarativer Ansatz einer RDF-Abfragesprache wurde 1998 in [DBS+98] vorgestellt. Der Kern dieser Arbeit ist die Abbildung von RDF-Metadaten auf Frame-Logik. Die RDF-Tripel werden extrahiert und in eine F-Logik-Wissensbasis geladen. Damit ist die Basis für Inferenz und Query geschaffen. Die Abfragesprache basiert auf Frame-Logik. Eine Implementierung dieses Ansatzes ist SiLRI (Simple Logic Based RDF Interpreter), der neben der Abfrage auch Inferenz ermöglicht, wodurch das Herleiten von Wissen aus einfachen Kontextinformationen möglich ist.

Metalog

Metalog [MS98a, MS98b] ist ein vom W3C unterstützter Versuch, auf RDF-Metadaten Abfragen und Inferenz zu ermöglichen. Metalog versucht, von der Syntax der Anfragesprache zu abstrahieren. Da der zentrale Ansatz die Zuordnung der RDF-Tripel auf Logik-Prädikate ist, erfolgt die Abfrage auf Logik-Ebene, wodurch auch Relationen ausgedrückt werden können. Der Ansatz von Metalog ist ein erweitertes

Metalog-Schema, Variablen und zusätzliche Konnektoren (“and”, „or”, „not” und „implies”), wodurch es möglich ist, Inferenzregeln in RDF Schema auszudrücken.

Neben der Anforderung von Metalog, Inferenz zu ermöglichen, wurde die Benutzerfreundlichkeit in den Vordergrund gestellt. Metalog propagiert die Spezifikation von Anfragen in englischsprachiger Notation eingebettet in RDF Schema. Doch gibt es bis heute keinen konkreten und vollständigen Vorschlag einer Abfragesprache.

RDF Squish query language

Ein neuerer, SQL-ähnlicher Ansatz ist die RDF Squish query language von Miller [Mil01]. Dieser Ansatz arbeitet direkt auf dem Datenmodell von RDF und erlaubt daher ausschließlich Abfragen auf Strukturebene. Der Nachteil, welcher dadurch entsteht, ist, dass alle Daten als Tripel interpretiert werden und Abfragen auf Semantikebene nur in geringen Umfang möglich sind.

RQL

Die einzige RDF-Abfragesprache, die auch RDF Schema unterstützt, ist RQL [KCP+00]. RQL (RDF Query Language) ist sehr ausdrucksstark, aber auch sehr komplex. RQL wurde am Institute for Computer Science – FORTH Heraklion entwickelt. Eine prototypische Implementierung ist in RDF Suite zu finden [RDFS].

Die Syntax von OQL (Object Query Language) diente als Vorbild und wurde zum Teil übernommen. Ebenfalls unterstützt werden Pfadausdrücke. Das Datenmodell von RQL ist ein Graph, der das RDF Modell mit der Schema Spezifikation verbindet. RQL setzt auf Kern-Abfragen, auf Basisfilter und die Möglichkeit, weitere Abfragen durch Komposition und Iteration zu bilden.

RQL baut auf objekt-relationale Datenbankmanagementsysteme (DBMS) und deren Datenmodell auf. Die Anfragesprache orientiert sich daher an der Abfragesprache für diese DBMS. Die Abbildung der RDF-Graphenstruktur erfolgt mit Hilfe von vier Basistabellen, in denen Klassen, Attribute, sowie die darauf bestehenden Beziehungen (Subklasse und Subattribut) festgehalten werden.

Da RQL sehr ausdrucksstark und die Semantik eindeutig definiert ist [KCP99], kann RQL aktuell als der interessanteste und aussichtsreichste Ansatz einer RDF-Abfragesprache betrachtet werden.

5.6 Schlussfolgerung

Wie die Beispiele in Kapitel 5.4 für die Beschreibung von Personen, Profilen, Objekten oder Lokation zeigen, existieren sehr viele Möglichkeiten zur Spezifikation von Kontextdaten, die sich mehr oder weniger etabliert und durchgesetzt haben. Jede dieser Spezifikationen konzentriert sich in der Regel auf ein spezielles Gebiet, doch treten häufig Überschneidungen auf. Bei der Beschreibung von Kontextdaten ist es daher wichtig, eine der Anwendung angepasste Beschreibung für eine Klasse von Kontextdaten auszuwählen und festzulegen.

Da jede Entwicklung eines Kontextsystems von der Zielgruppe von Applikationen abhängt, können nur Vorschläge gemacht werden. Eine einheitliche Beschreibungssprache kann es auf Grund der Komplexität nicht geben. Als ein Ergebnis dieser Arbeit wird daher empfohlen, XML oder RDF/XML als Spezifikations- und Kodierungssprache zu verwenden, etablierte Vokabulare geschickt auszuwählen und die Überschneidungskonflikte zu lösen. RDF/XML hat gegenüber XML den Vorteil, dass nicht nur Syntax sondern ansatzweise auch Semantik spezifiziert werden kann. Doch muss beim aktuellen Stand beider Techniken bedacht werden, dass RDF/XML als jüngere der beiden Technologien aktuell noch weniger technische Unterstützung in Form von Editoren besitzt.

Trotz der noch geringen Unterstützung durch (kommerzielle) Tools wurden mit dieser Arbeit die ersten Schritte in Richtung RDF/XML als Kodierungssprache für Kontextdaten unternommen, diese Technik propagiert, evaluiert und eingesetzt. Durch den Einsatz etablierter Vokabulare können die kontextabhängigen Anwendungen optimal unterstützt und Synergieeffekte genutzt werden. Um diesen Schritt zu komplettieren wurde RQL als RDF-Abfragesprache gewählt. Dies ermöglicht nicht nur die Abfrage auf Strukturebene, wie es auch mit XML-Abfragesprachen möglich ist, sondern auch auf semantischer Ebene.

Kapitel 6

Modellierung einer kontextabhängigen Zugriffskontrolle

In Kapitel 5 wurden die Begriffe „Kontext“ und „Kontextinformation“ beschrieben, wie sie in dieser Arbeit verstanden und modelliert werden. Dies bildet die Grundlage für das Verständnis für die in diesem Kapitel vorgestellte Modellierung und Spezifikation der kontextabhängigen Zugriffskontrollpolitik. In Abschnitt 6.1 werden die Grundlagen der Zugriffspolitik beschrieben. Abschnitt 6.2 enthält die Beschreibung der entworfenen Zugriffspolitiksprache CDACL (Context-Dependent Access Control Language). In Abschnitt 6.3 werden Konfliktlösungsstrategien für die Politiksprache beschrieben und in Abschnitt 6.4 anhand einiger bekannter Sicherheitsmodelle die Ausdrucksstärke von CDACL gezeigt.

6.1 Sicherheitspolitik

Ausgehend von dem in Abschnitt 5.1 beschriebenen Systemmodell, kann eine Zugriffspolitik als eine Sammlung von Regeln angesehen werden, die bestimmen, in welchen Zuständen sich das System befinden kann und welche Ereignisse für welche Zustände unter welchen Bedingungen zulässig sind. Jede Politikregel muss dabei festlegen, welchen Ereignissen und Komponenten sie zugeordnet ist und welche Zustandsbedingungen erfüllt sein müssen, damit sie ausgeführt werden darf.

In den folgenden Abschnitten werden die einzelnen Elemente der kontextabhängigen Zugriffspolitik (Context-Dependent Access Control; CDAC) sowie eine Sprache zur Beschreibung von kontextabhängigen Zugriffsrechten – die Context-Dependent Access Control Language (CDACL) - vorgestellt.

6.1.1 Komponenten

Die Komponenten *K* der Zugriffspolitik entsprechen den Komponenten, wie sie unter 5.1.1 definiert wurden. Es kann sich dabei um beliebige Einheiten eines verteilten Systems handeln. Um zwischen Ausgangs- und Zielkomponente eines Zugriffsereignisses unterscheiden zu können, wird von Subjekten und Objekten gesprochen. Ein Subjekt eines Zugriffsereignisses ist der Erzeuger, das Objekt die Zielkomponente. Ein Subjekt handelt immer als Stellvertreter eines Benutzers. Die Menge *S* der Subjekte und die Menge *O* der Objekte sind nicht disjunkt. Das Objekt eines Ereignisses kann Subjekt eines weiteren Ereignisses sein.

$K = S \cup O; \quad S \cap O \neq \{\}$, mit

K: Menge aller Komponenten des betrachteten Systems

S: Menge aller Subjekte

O: Menge aller Objekte

6.1.2 Ereignisse

Mögliche Ereignisse e_i entsprechen den Schnittstellen der Objekte. Sie können somit als Attributwerte betrachtet werden. Um mit dem allgemeinen Sprachgebrauch konform zu bleiben, werden Ereignisse im Themenbereich der Zugriffskontrolle auch Aktionen genannt.

$E = \{e_1, \ldots, e_n\}$ Menge aller Ereignisse

$P(E)$ Potenzmenge der Ereignismenge ohne Ø (leere Menge)

6.1.3 Sets

Um Zugriffsregeln skalierbar und damit handhabbar machen zu können, ist die Einführung weiterer Abstraktionsstufen notwendig. Komponenten und Ereignisse können zu Gruppen zusammengefasst werden. Dabei unterscheidet man zwischen Klassen und Mengen von Komponenten. Als Oberbegriff von Klassen und Mengen wird der Begriff „Set“ verwendet.

- **Klassen** werden durch bestimmte Eigenschaften der Komponenten festgelegt.

  Bsp.: Klasse „mp3-Dateien“, Klasse „Sicherheitsstufe 2“.

- **Mengen** von Komponenten sind explizit und beliebig zusammengesetzte Gruppen. Sie können weitere Mengen oder Klassen, welche über Mengenoperationen verknüpft werden, enthalten. Eine spezielle Menge stellt „allC“ dar, welche alle Komponenten des betrachteten Systems enthält.

  Bsp: Menge „Abteilungsleiter der Forschungssabteilung“, Menge „alle SW-Entwickler, die nicht zur Forschungsabteilung gehören“.

$R = R_S \dot{E} R_E \dot{E} R_O$ Menge aller Sets, mit

$R_S = \{ r_{S1}, \ldots r_{Sk} \}$ *mit* $r_{Si} = \{ s_i \mid s_i \hat{I} S \}$

$R_E = \{ r_{E1}, \ldots r_{Sl} \}$ *mit* $r_{Ei} = \{ e_i \mid e_i \hat{I} E \}$

$R_O = \{ r_{O1}, \ldots r_{O,m} \}$ *mit* $r_{Oi} = \{ o_i \mid o_i \hat{I} O \}$

Rollen, wie sie im RBAC-Modell [FK92], im Telekooperationsmodell von Grimm [Gri94], R&A-Modell von Schier [Sch99] oder im Task-Based Privacy-Modell von Fischer-Hübner [Fis01] definiert sind, entsprechen Mengen von Komponenten, da es sich um explizit zusammengesetzte Gruppen von Subjekten handelt. Eine Möglichkeit zur Gruppierung von Objekten ist in [Lie01] zu finden.

6.1.4 Politikregeln

Eine Politik besteht aus einzelnen Politikregeln, welche festlegen, ob die Durchführung einer Aktion *a* auf einem Objekt *o* durch ein Subjekt *s* im vorliegenden Kontext *C* „erlaubt“, „verboten“ oder „nicht entscheidbar“ ist. Dies entspricht der Modalität der Regel.

Eine Politikregel kann als eine Generalisierung einer ACL aufgefasst werden. Sie besteht aus zwei Teilen:

- einem Scope-Prädikat und
- einem Bedingungs-Prädikat (engl. requirement-predicate).

Das Scope-Prädikat legt fest, auf welche Komponenten und welche Aktionen die Regel anwendbar ist. Das Bedingungs-Prädikat legt die eigentliche Regelentscheidung fest.

Im Gegensatz zu anderen Modellen, legt eine Regel in CDAC somit nicht für ein bestimmtes Subjekt-Aktion-Objekt-Trippel eine positive oder negative Autorisierung fest, sondern spezifiziert über ein Bedingungs-Prädikat die Zugriffsentscheidung. Im Gegensatz zu anderen Zugriffspolitiken, liegt der Schwerpunkt der Betrachtung der kontextabhängigen Zugriffskontrollpolitik auf der Bedingung und damit auf dem Bedingungs-Prädikat der Regel.

Definition 10: Zugriffspolitik

Eine Zugriffspolitik *R* umfasst eine Menge von Politikregeln: $R=\{r_1, r_2, \ldots, r_m\}$

Definition 11: Politikregel

Eine Politikregel *r* spezifiziert, unter welchen Kontext-Bedingungen C_r die im Scope D_r festgelegten Aktionen *a* an Objekten *o* von Subjekten *s* initiiert werden dürfen.

Eine Politikregel ist eine Funktion

$r: (D_r, C_r) \rightarrow \{$*„allowed“, „denied“, „undecided“*$\}$,

welche das Tupel (D_r, C_r) mit

D_r: dem Regel-Scope und

$C_r = \{c_1, c_2, \ldots c_n\}$: einer Menge von (Kontext-)Bedingungen

auf den Wertebereich WR = {„allowed“, „denied“, „undecided“} abbildet.

Man unterscheidet zwischen einfachen und kombinierten Regeln. Kombinierte Regeln können ihrerseits Regeln enthalten und über die logischen Operationen ´AND´, ´OR´ und ´NOT´ miteinander verknüpft werden. Die Verknüpfung der Werte erfolgt mittels der dreiwertigen Logik (siehe Tabelle 2).

6.1.4.1 Regel-Scope

Der Regel-Scope legt den Wirkungsbereich einer Regel fest.

Definition 12: Regel-Scope

Der Regel-Scope D_r ist eine Menge von Subjekt-Aktion-Objekt-Tripel (S_r, A_r, O_r), welche festlegt, auf welche Komponenten und Aktionen die Regel anwendbar ist.

$$D_r = \{(S_r, A_r, O_r) \mid S_r \in P(S), A_r \in P(A), O_r \in P(O) \}$$

6.1.4.2 Bedingungsteil

Der Bedingungsteil liefert die eigentliche Regelentscheidung. Er besteht aus booleschen Ausdrücken von Attributen beliebiger System-Komponenten oder Ereignis-Muster, wie sie in Kapitel 5.1.3 definiert wurden. Zur Spezifikation der booleschen Ausdrücke stehen folgende Operationen zur Verfügung:

Vergleichsoperationen:

´<´ kleiner

´≤´ kleiner gleich

´=´ gleich („equal“)

´≥´ größer gleich

´>´ größer

´!=´ ungleich

´∈´ Element von („in“)

´∉´ kein Element von

Spezielle Werte:

„true“

„false“

Logischen Operationen:

´&´ logisches AND

, ´|´ logisches OR

´¬´ logisches NOT

Ist ein boolescher Ausdruck aufgrund fehlender oder unzureichender (Kontext-)-Informationen nicht auswertbar, so wird der Wert „undecided“ zurückgeliefert.

Eine Regelbedingung kann aus mehreren booleschen Ausdrücken bestehen, welche über die booleschen Operationen AND, OR und NOT verknüpft werden. Die Verknüpfung der Werte erfolgt mittels der in Tabelle 2 dargestellten dreiwertigen Logik.

c1	*c2*	*c1* AND *c2*	*c1* OR *c2*	NOT *c1*
allowed	allowed	allowed	allowed	denied
allowed	denied	denied	allowed	denied
undecided	allowed	undecided	undecided	undecided
undecided	denied	undecided	undecided	undecided
denied	denied	denied	denied	allowed

Tabelle 2 Dreiwertige Logik für die Verknüpfung von Bedingungen und Regeln

Es werden vier große Klassen von Regelbedingungen unterschieden:

1. *Einfache Zeit- und Datumsbedingungen*

   Einfache Zeit- und Datumsbedingungen beziehen sich auf die lokale Zeit und Datum der Politik-Durchsetzungskomponente. Mögliche Bedingungsoperationen sind dabei die Vergleichsoperationen ´<´, ´<=´, ´>´, ´>=´, ´=´ und ´!=´. Zeitbedingungen, welche eine andere Uhr als Bezugspunkt verwenden, müssen als Kontext-Bedingungen formuliert werden. Kontext-Bedingungen werden, im Gegensatz zu Zeit- und Datumsbedingungen, nicht lokal von der Durchsetzungskomponente ausgewertet, sondern vom Kontext-System (siehe Abschnitt 8.2).

2. *Kontext-Bedingungen*

   Unter Kontextbedingungen werden alle Bedingungen zusammengefasst, die sich auf Kontextinformationen (siehe Abschnitt 5.2.2) beziehen.

3. *Scope-Bedingungen*

   Unter Scope-Bedingungen werden alle Bedingungen verstanden, welche sich auf die Komponenten Subjekt, Aktion und Objekt des aktuellen Ereignisses beziehen. Dazu gehören Gleichheitsbedingungen wie beispielsweise

   ```
   equals($object.owner, $subject.id)
   ```

   oder Mengenbedingungen wie beispielsweise

   ```
   in($subject, „developerList“) oder
   in($object, „mpFiles“).
   ```

Je nach Administrationsmodell für Komponenten-Klassen und –Mengen stellen die Scope-Bedingungen eine eigene Klasse dar oder sind eine Untermenge der Kontext-Bedingungen.

4. *Ereignis-Bedingungen*

   Unter Ereignis-Bedingungen werden alle Bedingungen verstanden, welche sich auf Ereignisse beziehen. Es werden drei Arten von Ereignisbedingungen unterschieden:

   - Kausale Abhängigkeiten und
   - Zeitliche Abhängigkeit wie in Abschnitt 5.1.2 definiert sowie
   - Protokolle wie in Abschnitt 5.2.3.5 spezifiziert.

6.1.5 Schutzzustand

Der Schutzzustand eines Systems mit kontextabhängiger Zugriffskontrolle kann als drei-dimensionale Matrix der Form

$$M: (S \dot{E} R_S) \times (O \dot{E} R_O) \times P(fCC) \rightarrow P(E)$$

modelliert werden. Die Zugriffsmatrix ist somit eine Funktion, welche Tripel aus Subjekten oder Subjekt-Sets, Objekten oder Objekt-Sets sowie Mengen von erfüllten Kontext-Bedingungen (fCC) auf Mengen von Ereignissen abbildet. Unter einer Kontext-Bedingung wird dabei eine Regelbedingung wie in Abschnitt 6.1.4.2 definiert verstanden.

6.2 Context-Dependent Access Control Language

Das Ziel dieser Arbeit war es nicht, eine neue Spezifikationssprache zu entwickeln, um ein Sicherheitsmodell zu beschreiben, wie es Brewer und Nash mit der Prädikatenlogik zur Beschreibung ihres Chinese Wall-Sicherheitsmodells machten oder Terry und Wiseman mit der Spezifikationssprache Z. Vielmehr sollte eine Methode zur Festlegung von Zugriffsrechten gefunden werden, die den in Kapitel 2.3 definierten Anforderungen genügt.

Während ein Sicherheitsmodell in einer formalen Weise die Sicherheitspolitik und ihre Arbeitsweise beschreibt, stellt eine Politik-Spezifikationssprache das formale Mittel dar, das zugrunde liegende abstrakte Sicherheitsmodell eindeutig und präzise zu beschreiben, so dass es einer Durchsetzungskomponente möglich ist, anhand der definierten Regeln eine Entscheidung zu treffen.

Bei der Definition einer Politiksprache sind folgende Festlegungen zu treffen:

1. Die Syntax der Sprache, die zur Beschreibung der Konzepte und deren Beziehungen eingesetzt wird. Eine Syntax kann dabei als formale Grammatik spezifiziert werden, in graphischer Form oder in einem geeigneten Austauschformat.

2. Die Semantik des Konzeptes, welche in der Regel mit einer formalen oder semi-formalen Methode eine Abbildung der Syntax auf die Objekte im Modell erreicht.
3. Eine Methode zur operationellen Durchsetzung.

In den nächsten Abschnitten werden die Syntax und die Semantik der entwickelten Context-Dependent Access Control Language (CDACL) genauer spezifiziert. Die Methoden zur operationellen Durchsetzung werden in Kapitel 8 beschrieben.

6.2.1 XML als Spezifikationssprachen

Als eine gängige Methode zur Spezifikation von Austauschdaten in einer für Menschen lesbaren Form, hat sich in den letzten Jahren XML (eXtensible Markup Language) [BPS+00] durchgesetzt. Angewendet auf den Bereich der Politiksprachen, bietet dieser Ansatz gegenüber den herkömmlichen formalen oder graphischen Methoden die folgenden Vorteile:

- Die Politik ist maschinen- und menschenlesbar,
- besitzt einen modularen Aufbau und ist daher einfache erweiterbar,
- kann schnell analysiert werden und erlaubt
- einen problemlosen Austausch über Domänengrenzen hinweg.

Aus diesem Grunde, wird in dieser Arbeit XML und XML Schema zur Spezifikation der Sicherheitspolitik verwendet (siehe auch Abschnitt 5.4.1).

Zwei Ansätze, welche XML zur Spezifikation einer Zugriffspolitik verwenden, sind die XML Access Control Language (XACL) [HK00, KH00] und deren noch in der Entwicklung befindende Nachfolgerin XML Access Control Markup Language (XACML) [Kud01] von IBM. Beide Sprachen sind ausschließlich für die Festlegung von Zugriffsrechte auf XML Dokumente und Teilinhalte bestimmt.

6.2.2 Sprachbeschreibung

Eine vollständige Beschreibung der Context-Dependent Access Control Language in XML ist in Anhang A zu finden. Im Folgenden werden nur die wichtigsten Sprachkonstrukte durch beispielhafte Sprach-Fragmente erläutert. Auslassungen innerhalb dieses Fragments werden mit „…“ angedeutet. Eine graphische Übersicht der wichtigsten Sprachelemente bietet Abbildung 24.

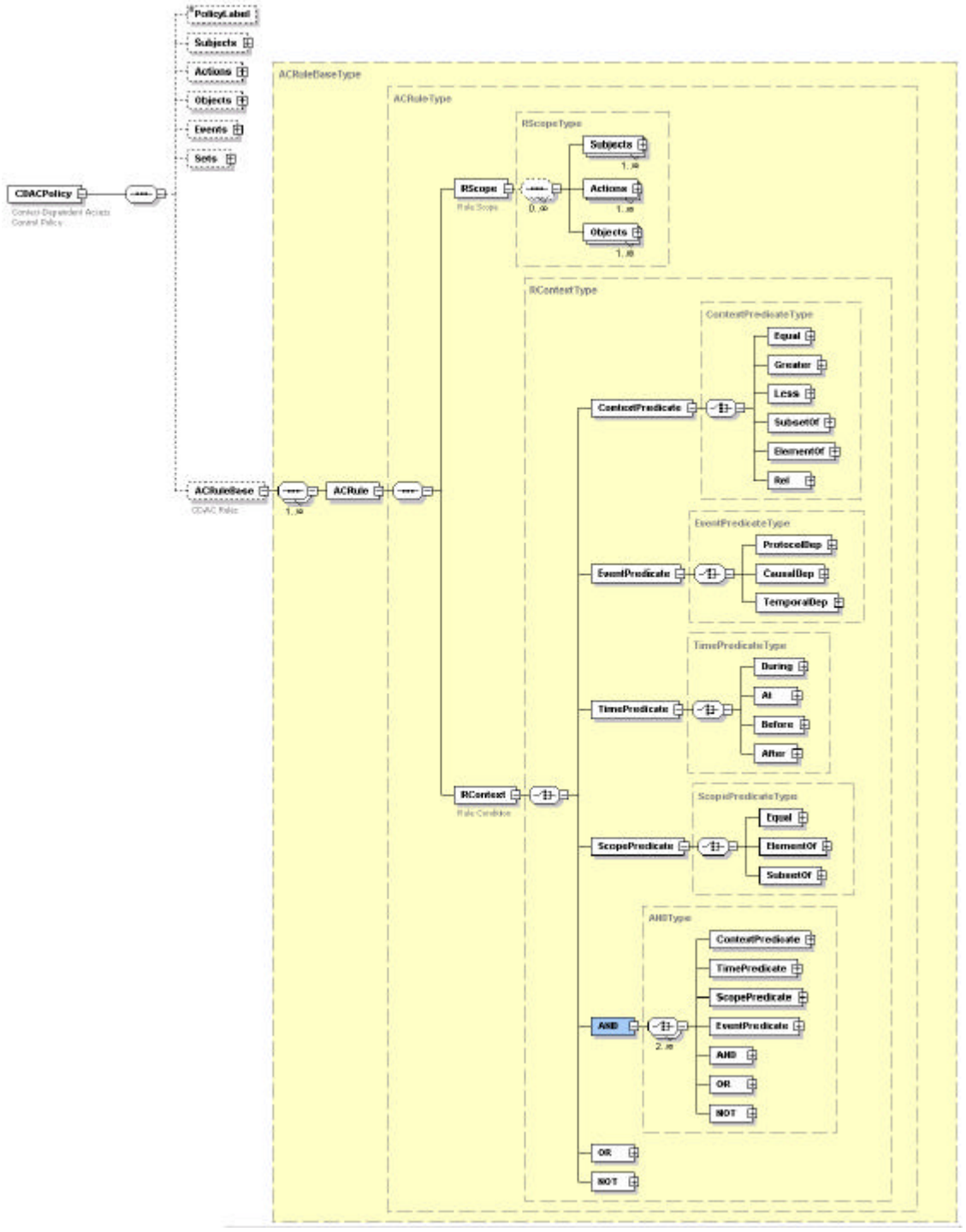

Abbildung 24 Graphische Darstellung der CDACL-Syntax

6.2.2.1 Politik

Das Root-Element jeder als XML-Datei gespeicherten CDAC-Politik bildet die Definition der CDAC-Zugriffspolitik (kurz „Politik“):

```
<CDACPolicy … ID="String" version="String">
```

Jede Politik wird eindeutig über eine Kennung *PolicyLabel* identifiziert. Die Angabe *version* spezifiziert die verwendete Politikversion.
Eine Politik setzt sich zusammen aus einer abzählbar endlichen Menge von Regeln. Optional können Subjekte, Aktionen, Objekten und Sets von Subjekten, Objekten und Aktionen festgelegt werden.

```
<CDACPolicy … ID="String" version="String">
   <PolicyLabel>Politikname</PolicyLabel>
   <Subjects>   …  </Subjects>
   <Actions>    …  </Actions>
   <Objects>    …  </Objects>
   <Sets>       …  </Sets>
   <ACRules>    …  </ACRules>
</CDACPolicy>
```

6.2.2.2 Subjekte, Aktionen und Objekte

Die Menge der Subjekte besteht aus einer Auflistung einzelner Subjekte. Ein Subjekt kann eindeutig über eine URI, einen Namen *uname*, eine eindeutige Kennung *uid* oder eine Referenz auf ein zuvor spezifiziertes Subjekt identifiziert werden. Um ein spezifiziertes Subjekt referenzieren zu können, muss dieses Subjekt durch eine für diese XML-Datei eindeutige Identifikation *ID* identifiziert werden. Wird ein Subjekt nicht referenziert, kann diese optionale Identifikation entfallen. Nicht unterstützt wird in der aktuellen Version von CDACL die Identifikation eines Subjekts über ein Zertifikat.

```
<Subjects>
    <Subject ID=”_hans-meier”><href>http://…</href></Subject>
    <Subject><uname>hmueller</uname></Subject>
    <Subject><uid>007</uid></Subject>
    <Subject ref="_hans-meier"/>
</Subjects>
```

Objekte werden äquivalent zu den Subjekten über eine URI oder eine Referenz auf ein spezifiziertes Objekt identifiziert.

```
<Objects>
    <Object ID=”_O-ID01”><href>http://…</href></Object>
    <Object ref=”_O-ID01”/>
</Objects>
```

Aktionen werden durch einen Namen und Aufrufargumente festgelegt. Wird eine eindeutige Identifikation *ID* spezifiziert, so kann über diese die Aktion referenziert werden.

```
<Actions>
    <Action ID=”_A-ID01”>
        <Name>execute</Name>
        <Args>Arg1</Args>
        <Args>Arg2</Args>
    </Action>
    <Action ref="_A-ID01"/>
</Actions>
```

Die explizite Spezifikation von Objekten, Aktionen und Objekten in einer Politik dient ausschließlich einer vereinfachten Schreibweise, da damit die Möglichkeit eröffnet wird, in der Regelbasis oder bei der Definition von Sets auf de entsprechenden Elemente zu verweisen. Eine Mehrfachspezifikation wird damit vermieden, ist andererseits aber auch nicht verboten.

6.2.2.3 Sets

Sets ermöglichen die temporäre Gruppierung von Elementen. Die Bildung von Hierarchien ist durch die Spezifikation von Untergruppen möglich. Jede Untergruppe ist eindeutig und implizit einer übergeordneten Gruppe zugeordnet. Die Zuordnung von Elementen zu Gruppen erfolgt bei der Spezifikation der Sets. Ist ein Element mehreren Sets zugeordnet, so ist dieses bei jeder einzelnen Gruppe zu spezifizieren.

```
<Sets>
    <SubjectSet ID="_employee">
        <SubSet ID="_departmentA">
            <Subject ref="_ursula-koenig"/>
            <SubSet ID=”_project_A”>
                <Subject ref="_elke-winter"/>
            </SubSet>
            <SubSet ID=”_project_B”/>
                <Subject><uname>hmueller</uname></Subject>
            </SubSet>
        </SubSet>
        <SubSet ID="_heaf-of-department">
            <Subject ref="_friedolin-hoffmann"/>
            <Subject ref="_rainer-lutze"/>
        </SubSet>
        <SubSet ID="_secretary">
            <Subject ref="_ursula-koenig"/>
        </SubSet>
    </SubjectSet>
    <SubjectSet ID="_employees">
        <gname>employees</gname>
```

```
        </SubjectSet>
        <ActionSet ID="_public_api">
            <Action ref="_execute"/>
        </ActionSet>
        <ObjectSet ID="_private">
            <Object><href>http://….calendar.xml</href></Object>
            <Include>http://…./malu</Include>
            <Exclude>http://…./malu/private/</Exclude>
        </ObjectSet>
    </Sets>
```

Obiges Beispiel zeigt, dass „Ursula Koenig" sowohl der Gruppe *_secretary* der Sekretärinnen, der Projektgruppe *_project_B* und der übergeordneten Gruppe *_employee* der Angestellten angehört.

Subjektgruppen, welche durch die Benutzer-, Klassen- und Mengenverwaltung vordefiniert sind, sind immer gültig und können bei der Spezifikation einer CDAC-Politik verwendet werden. Werden neue Gruppen spezifiziert, so ist darauf zu achten, dass bei der Benennung keine Konflikte mit den in der Verwaltung vordefinierten Gruppen auftreten (siehe Kapitel 8.3).

Objektgruppen können sich aus Einzelobjekten zusammensetzen oder durch ein Verzeichnis spezifiziert werden. Verzeichnisse können dabei explizit dazu genommen (*"Include"*) oder ausgeschlossen (*"Exclude"*) werden. Letzteres ist nur für Unterverzeichnisse von inkludierten Verzeichnissen sinnvoll, da implizit alles ausgeschlossen ist, was nicht explizit als aufgenommen spezifiziert wird. So wird in obigem Beispiel in die Menge der Objekte das Verzeichnis „…/malu" inklusive aller darunter liegenden Verzeichnisse aufgenommen, anschließend das Unterverzeichnis „…/malu/private" explizit wieder ausgeschlossen.

6.2.2.4 Regel

Eine Zugriffsregel besteht aus genau einem Scope und einem Kontextteil.

```
    <ACRule ID="rule_001">
        <RScope>
            …
        </RScope>
        <RContext>
            ….
        </RContext>
    </ACRule>
```

Scope

Der Scope besteht aus genau einer Subjekt-Menge, genau einer Aktions-Menge und genau einer Objekt-Menge. Jede Menge kann leer sein, einzelne Elemente oder Element-Sets enthalten. Jede Menge ohne Angabe von Elementen oder Sets bedeutet implizit „alle" Vertreter dieses Elementtyps (im Beispiel die Menge der Aktionen).

```
<RScope>
    <Subjects>
        <SubjectSet ref="_heaf-of-department"/>
        <SubjectSet ref="_secretary"/>
        <Subject ref="="_ursula-koenig"/>
    </Subjects>
    <Actions></Actions>
    <Objects>
        <Object><href>http://.../intranet/</href></Object>
    </Objects>
</RScope>
```

Kontext

Das Kontextprädikat bestimmt die eigentliche Regelentscheidung „*Zugriff erlaubt*" ("allowed"), „*Zugriff verweigert*" ("denied") oder „*nicht entscheidbar*". Die Entscheidung setzt sich aus den booleschen Funktionen „AND", „OR" und „NOT", den Zeit-, Scope-, Kontext- und Event-Prädikaten sowie den beiden expliziten Werten „ALLOWED" und „DENIED" zusammen.

Die booleschen Funktionen „AND" und „OR" erwarten mindestens zwei Argumente, die Funktion „NOT" genau ein Argument. Als Argumente werden boolesche Funktionen oder Prädikate erwartet.

```
<RContext>
    <AND>
        <TimePredicate>
            <During>
                <tStart>6:00</tStart>
                <tEnd>20:00</tEnd>
            </During>
        </TimePredicate>
        <OR>
            <ContextPredicate>
                <Greater>
                    <Var Unit="celsius">__temperatur</Var>
                    <Const Value="24"/>
                </Greater>
            </ContextPredicate>
            <ContextPredicate>
                <Less>
                    <Var  Unit="celsius"
                          Certainty="0.7">__temperatur
                    </Var>
                    <Const Value="20"/>
                </Less>
            </ContextPredicate>
        </OR>
    </AND>
```

```
</RContext>
```

Das Kontext-Prädikat ermöglicht die Definition einer Kontext-Bedingung. Eine Kontext-Bedingung überprüft Kontextinformationen, die nur vom Kontextserver ermittelt werden können. Als vordefinierte Funktionen stehen „Equal“, „Greater“, „Less“, „SubsetOf“, „ElementOf“ und „Rel“ zur Verfügung. Die Funktion „Rel“ (für *Rel*ation) erlaubt das Abfragen von speziellem Situationswissen.

Als Argumente einer Kontextbedingung können sowohl Konstanten wie auch Kontextvariablen eingesetzt werden. Jede Kontextvariable verfügt über eine Standard-Einheit. Wird die im Vergleich angegebene Kontextvariable ohne Einheit angegeben, so wird implizit die Standard-Einheit verwendet. Durch die explizite Angabe einer Einheit kann die Standardeinstellung verändert werden, doch ist zu beachten, dass die angegebene Einheit dem Kontextsystem bekannt ist und als Abfrageparameter zur Verfügung gestellt wird.

Für jede Kontextvariable kann ein Mindest-Sicherheitswert („Certainty“) spezifiziert werden. Dieser Wert legt fest, welchen Sicherheitswert (Umkehrung des Unsicherheitsfaktors) der Kontextwert besitzen muss, damit er für die Berechnung akzeptiert wird. Liegt der zur Laufzeit ermittelte Sicherheitswert der Kontextvariablen unterhalb dieser Schranke, wird der Kontextwert als „zu unsicher“ eingestuft und als nicht auswertbar betrachtet.

Das Scope-Prädikat erlaubt die Formulierung von Bedingungen, welche sich auf den Scope der Regel beziehen. Da sich der Regel-Scope aus Elementen und Mengen zusammensetzt, stehen die Funktionen „Equal“, „ElementOf“ und „SubsetOf“ zur Verfügung. Weitere Funktionen wie „ist nicht Element von“, oder „ist Super-Set“ können aus den übrigen Scope-Prädikaten und den booleschen Funktionen zusammengesetzt werden.

Als Zeit-Prädikate stehen „During“, „At“, „Before“ und „After“ zur Verfügung. Ob es sich dabei um absolute oder relative Angaben handelt, entscheiden die Argumente.

```
<OR>
    <TimePredicate>
        <During>
            <tStart>6:00</tStart>
            <tEnd>20:00</tEnd>
        </During>
    </TimePredicate>
    <TimePredicate>
        <During>
            <tStart>23:00</tStart>
            <Const Value="2" Unit="hour"/>
        </During>
    </TimePredicate>
</OR>
```

Das Event-Prädikat erlaubt die Formulierung von Bedingungen bezüglich Events. Als Funktionen werden unterstützt: „CausalDep“ (kausale Abhängigkeit), „Temporal-Dep“ (temporale Abhängigkeit) und „Protocol“.

```
<CausalDep>
    <Event>
        <ESubject>ce.subject</ESubject>
        <EObject>ce.object</EObject>
        <EAction>create</EAction>
    </Event>
    <TimeCond>
        <During>
            <tStart>ce.time</tStart>
            <Const Value="2" Unit="hour"/>
        </During>
    </TimeCond>
</CausalDep>
```

Das obige Beispiel zeigt die kausale Bedingung, dass der Initiator des aktuellen Ereignisses dieses Objekt während der letzten 2 Stunden erzeugt hat. Auf das aktuelle Ereignis kann über den vordefinierten Bezeichner „ce“ (current event) zugegriffen werden.

```
<NOT>
    <TemporalDep>
        <Event>
            <Esubject/>
            <Eobject/>
            <EAction>openSocket</EAction>
        </Event>
        <TimeCond>
            <During>
                <tStart>ce.time</tStart>
                <Const Value="1" Unit="hour"/>
            </During>
        </TimeCond>
    </TemporalDep>
</NOT>
```

Im Gegensatz zur kausalen Bedingung, bei der mindestens eine Überschneidung bei Objekt und Subjekt des spezifizierten Ereignisses mit Objekt und Subjekt des aktuellen Ereignisses vorhanden sein muss, können bei einer temporalen Ereignisbedingung beliebige Ereignisse spezifiziert werden. Wird kein Subjekt oder Objekt spezifiziert, so wird implizit ein „beliebiges“ Subjekt bzw. Objekt angenommen. Das Beispiel zeigt die temporale Bedingung, dass während der letzten Stunde niemand eine Socketverbindung geöffnet hat.

```
<ProtocolDep>
    <Initial>
        <ESubject>ce.subject</ESubject>
        <EObject>ce.object</EObject>
        <EAction>create</EAction>
    </Initial>
    <Protocol>
        <ESeq>
            <Event>
                <ESubject>ce.subject</ESubject>
                <EObject>ce.object</EObject>
                <EAction>write</EAction>
            </Event>
            <EIt>
                <ESubject> </ESubject>
                <EObject>ce.object</EObject>
                <EAction>read</EAction>
            </Eit>
        </ESeq>
    </Protocol>
</ProtocolDep>
```

Unter einem Protokoll wird eine Menge von Ereignissen mit einem optional definierten Anfangszustand verstanden, wie es in Abschnitt 5.2.3.5 vorgestellt wurde. Im obigen Beispiel eines Protokolls ist die Bedingung definiert, dass ausgehend von einem initialen Ereignis „create“, das auf diesem Objekt genau einmal vom Erzeuger des Objekts geschrieben werden muss und anschließend nur noch beliebig oft gelesen werden darf. Eine Zeitbeschränkung ist in diesem Beispiel nicht festgelegt.

Da das Protokoll durch reguläre Ausdrücke spezifiziert ist, kann es in einen endlichen deterministischen Automaten mit Anfangszustand umgesetzt werden [Eck97]. Jedes neue Ereignis wird darauf geprüft, ob der Automat sich in einem Zustand befindet, in dem dieses Ereignis akzeptiert wird.

6.2.3 Namensschemata

Ein wichtiger Punkt bei der Spezifikation der Zugriffsspolitik in CDACL ist die Eindeutigkeit der verwendeten Namen für Subjekte, Aktionen, Objekte und Kontextinformationen.

Subjekte werden eindeutig durch ihre Benutzeridentifikation (User Identifier; UID) oder durch die Angabe von *<Benutzernamen>@<Domain-Name>* identifiziert. Die Angabe der Domäne kann entfallen, falls die Zugehörigkeit eindeutig ist (z.B. wenn nur eine Domäne existiert). Gruppen und Klassen von Benutzern besitzen einen eindeutigen, dem System bekannten Namen. Gleiches gilt für Aktionen und Aktionsgruppen.

Zur Identifikation von Objekten wird der Uniform Resource Identifier (URI) [Bar94] bzw. den Uniform Resource Locator (URL) als Teilspezifikation des URI eingesetzt. Damit ist eine eindeutige Objektidentifikation folgendermaßen aufgebaut:

<Zugangsprotokoll>://<Hostadresse>[:<Portnummer>]/<Pfadangabe>

Als Zugangsprotokoll werden in der Spezifikation http, ftp, wais, news, telnet, gopher, mailto und file unterstützt. Als Hostadresse kann die IP-Adresse oder der Domänen-Name angegeben werden. Als Pfad wird der volle Pfadname einschließlich eventueller Dateinamen angegeben.

Die eindeutige Identifikation von Kontextinformationen gestaltet sich komplexer, da diese von unterschiedlichster Natur sein können. Nachfolgend werden die wichtigsten Vertreter von Kontextinformationen – Lokation, Zeit, Sensorinformationen sowie Personen und Geräte vorgestellt.

Die Interpretation von Lokationsinformation hängt von dem verwendeten Lokationsmodell ab (siehe Kapitel 5.2.3.1). Die Spezifikation kann jedoch so gestaltet sein, dass sie die wichtigsten Modelle einschließt. Ein solches Namensschema wurde von Ulf Leonhardt in [Leo98] vorgeschlagen. Dieses Schema erfüllt die Anforderungen an Skalierbarkeit, Intuitivität und Allgemeinheit. Um dieses Schema in Verbindung mit anderen Spezifikationen von Kontextinformationen in dieser Arbeit verwenden zu können, wurde die Syntax geringfügig abgeändert.

Eine Lokation ist eindeutig durch die Angabe eines Bereichs und einer Position spezifiziert. Der Bereich kann durch die Angabe geometrischer Daten oder durch ein Lokationskonzept vorgegeben werden. Die Positionsangabe bildet den Bezugspunkt für die Gebietsangabe. Sie kann durch die Angabe eines lokalisierten Objekts, einer symbolischen oder einer geographischen Ortsangabe erfolgen.

```
location            ::= <area> '%' <position>
area                ::= <location concept> | <geometric definition>
position            ::= <located object> | <fixed position>
fixed position      ::= <geometric position> | <symbolic position>
geometric position  ::= <reference system> ':' <coordinates>
symbolic position   ::= (<symbolic position> '/' <label>)
                        | <well known position>
```

Ein Beispiel einer Lokationsangabe, basierend auf dem Raumkonzept, welches in Abschnitt 5.2.3.1 vorgestellt wurde, das einen Raum 105 in Gebäude 7 der TU Darmstadt benennt:

Room%TU Darmstadt/Gebäude 7/Raum 105

Eine Datums- und Zeitangabe erfolgt entweder durch die Angabe eines vordefinierten Intervalls, welches eindeutig über einen Bezeichner identifiziert wird, durch die Angabe eines Datums- oder Zeitpunktes, eines Zeitintervalls oder einer Zeitspanne. Die Angabe einer Zeitspanne kann in Sekunden, Minuten, Stunden, etc. erfolgen, die Angabe eines Intervalls durch Start- und Endzeitpunkt. Ein absoluter Zeitpunkt wird in ISO 8601-Notation [ISO8601] spezifiziert.

```
time            ::= <time_ident> | <time_period>
                    | <time_interval>| <time_point>
time_interval   ::= '['<iso_8601> ',' <iso_8601>']'
time_period     ::= <iso_8601>
```

Unvollständige Zeitangaben, wie beispielsweise der Form „YYYY-MM-DD hh“, spezifizieren ein Zeitintervall. Eine Datums- oder Zeitangabe ohne weitere Angabe eines Bezugssystems bezieht sich implizit auf die Greenwich Mean Time (GMT). Wird eine andere Referenzzeit verwendet, so ist dies als Parameter und nicht als Teil des Namens zu spezifizieren.

Zur Spezifikation von beliebigen Kontextdaten (ausgenommen Lokation und Zeit) wurde folgendes Modell an Anlehnung an die Notation für Lokation entwickelt. Ausgehend von der Annahme, dass ein Wert einer Kontextinformation eindeutig durch den Klassennamen *context_ident* der Kontextinformation und den Angaben von Lokation und Zeitpunkt identifiziert werden kann, wird eine Kontextinformation absolut spezifiziert durch:

```
context         ::= <context_ident> ['?' <position> '::' <time>]
                    | <context_ident> '?' <position> ['::' <time>]
                    | <context_ident> '?' [<position>] '::' <time>]
```

Die Angabe der Lokation und der Zeit erfolgt in der zuvor beschriebenen Notation. Beide Angaben sind optional. Erfolgt keine Angabe, so wird implizit die aktuelle Lokation des Aufrufers bzw. die aktuelle Zeit der Auswertung der Anfrage eingesetzt. Die Überlegung, de hinter dieser Festlegung steckt, ist, dass es sich bei der verwendeten Kontextinformation um aktuelle Daten der näheren Umgebung des Benutzers handelt.

Neben der absoluten Angabe einer Kontextinformation, kann eine Spezifikation relativ zu Personen oder Gegenständen vorgenommen werden, da eine Kontextinformation in der Regel auch als Attribut dieser Personen oder Gegenstände interpretiert werden kann. Eine relative Kontextinformation umfasst die Angabe der Kontextklasse, der Zeit (optional), der Beziehungsrelation (optional) und die Angabe einer Person oder eines Gegenstandes. Als Beziehungsrelationen stehen aktuell die Lokationsrelationen, wie sie bei der Spezifikation einer Lokation möglich sind (<area>-Angabe), zur Verfügung.

```
context   ::= <context_ident> '?'[<pos_rel>::]<object_ident> ['::'<time>]
            | <context_ident>'?'[<pos_rel>::]<subject_ident> ['::'<time>]
```

Mit diesen Festlegungen können beispielsweise folgende Kontextinformationen formuliert werden:

- *Temperature?TU Darmstadt/Gebäude 7/Raum 105*

  Temperatur im Raum 105 von Gebäude 7 der TU Darmstadt zum aktuellen Zeitpunkt

- *Location?room:hans@mueller-gmbh.de*

  Aktueller Aufenthaltsort des Benutzers „hans“ der Domäne „mueller-gmbh.de“ zum aktuellen Zeitpunkt.

- *Location?room:hans@mueller-gmbh.de::2001-11-11 11:11+0100*

  Aufenthaltsort des Benutzers „hans“ der Domäne „mueller-gmbh.de“ am 11.11.2001 um 11:11 Uhr Mitteleuropäischer Zeit (CET).

- *Location?1m:131.32.27.223*

  Gebiet mit dem Radius 1m um das Gerät mit der IP-Adresse 131.32.27.223.

- *AuthRoles?hans*

  Dem Benutzer „hans“ aktuell zugeordnete Rollen.

- *NearByPersons?hans*

  Alle Personen in der Nähe von Benutzer „hans”.

- *Lend?hans*

  Alles, was Benutzer „hans“ ausgeliehen hat .

6.3 Konfliktlösungsstrategien

Die vorgestellte Zugriffspolitiksprache ermöglicht die Definition sowohl von positiven wie auch von negativen Bedingungen. Dies ist eine Eigenschaft, die als sehr wichtig betrachtet wird und beispielsweise für den C2-Level des TCSEC-Standards [DoD85] vorgeschrieben ist. Der Nachteil, der dadurch entsteht ist, dass Konflikte zwischen mehreren Politikregeln bzw. Politiken auftreten können.

Im Bereich Sicherheitspolitik werden verschiedene Konfliktarten unterschieden. Eine sehr detaillierte Klassifikation von Konflikten für Management-Politiken ist in [MS94] zu finden. Da eine Zugriffs- oder Autorisierungspolitik eine Untermenge der Management-Politiken darstellt, treten je nach zugrunde liegendem Sicherheitsmodell nur einige der in [MS94] genannten Konfliktklassen auf. Da die vorgestellte Politiksprache ausschließlich die Autorisierung für eine Aktion bzw. einen Zugriff und keine durchzuführenden Aktionen festlegt, wie es in Obligation-Policies der Fall ist, können nur so genannte Modalitätskonflikte (Positiv-Negativ-Konflikte) durch sich widersprechende Politikregeln auftreten.

Positiv-Negativ-Konflikte treten genau dann auf, wenn zwei oder mehr Regeln auf die aktuelle Zugriffsanforderung anwendbar sind und die Ergebnisse der Regeln sich widersprechen. Da in herkömmlichen Autorisierungspolitiken das Ergebnis in der Regel nicht vom Kontext abhängt, sondern von der Art der Regel – positive oder negative Autorisierungsregel – werden in der Literatur zur Lösung eines Positiv-Negativ-Konflikts im Wesentlichen zwei Strategien vorgeschlagen: der zweistufige Prioritätsansatz und der dreistufige Prioritätsansatz. Im zweistufigen Prioritätsansatz dominiert die höher priorisierte positive Regel die nieder-priore negative Regel [MST90]. Sind in einem System sowohl DAC wie auch MAC vertreten, so legt der dreistufige Prioritätsansatz [DoD85] folgende Reihenfolge fest: die höchst-priore

negative MAC-Regel dominiert die mittel-priore positive DAC-Regel, welche ihrerseits die niedrig-priore negative Default-Regel dominiert.

Einen möglichen Ansatz zur Vermeidung von Positiv-Negativ-Konflikte wird in [LS97] beschrieben. Dabei wird bei der Spezifikation einer neuen Regel überprüft, ob sich deren Scope mit dem Scope einer schon spezifizierten Regel überschneidet. Da herkömmliche Autorisierungsregeln explizit und eindeutig als positiv oder negativ spezifiziert sind, wird bei einer Scope-Überschneidung die neue Regel nur dann zugelassen, wenn es sich um eine Spezialisierung der im Konflikt liegenden Regel handelt. In Fällen von Modalitätskonflikten, bei denen die Scopes der Regeln nicht über Spezialisierung in Beziehung stehen, wird der Regel-Ersteller benachrichtigt, der die Konflikte explizit lösen muss. Semantische Konflikte sind anwendungsspezifisch und können weder automatisch erkannt noch aufgelöst werden. Diese Konflikte können über Metaregeln zur Laufzeit gelöst werden.

Neuere Sicherheitspolitiken, in denen die Regeln nicht schon im Wesen als positive oder negative Autorisierung spezifiziert werden, erfordern weitere, teilweise sehr aufwendige Strategien zur Lösung von Konflikten. In der Literatur finden sich dafür folgende Ansätze:

- *Reihenfolgenansatze:*

  Der einfachste Ansatz ist der Reihenfolgenansatz, der die Entscheidung der ersten Regel der Konfliktmenge als Ergebnis übernimmt. Alle weiteren Regeln der Konfliktmenge werden nicht berücksichtigt. Die Konfliktmenge besteht aus allen Regeln, welche auf das aktuelle Ereignis anwendbar sind. Die Regelentscheidung hängt beim Reihenfolge-Ansatz somit von der Strategie zur Bestimmung der Konfliktmenge und der daraus resultierenden Reihenfolge in der Regelliste ab. Diese wird in der Regel selten nach irgendwelchen Gesichtspunkten geordnet sein, sondern von der Speicherung der Regeln und damit der Reihenfolge der Auswertung abhängen.

- *Prioritätsansatz:*

  Einer der einfacheren Ansätze ist die Erweiterung des einfachen, oben beschriebenen Prioritätsansatzes zu explizit zugewiesenen Prioritäten, welche festlegen, welche Regeln über welche dominieren. Zur Bestimmung der Priorität einer Regel werden dabei Informationen wie der Status des Regel-Erstellers, das Erstellungsdatum oder der Detaillierungsgrad des Regel-Scopes verwendet [LS97, Lup98, LS99].

- *Metaregelansatz:*

  Ein weiterer Ansatz zur Lösung von Konflikten ist die Einführung von Metaregeln, die zur Laufzeit anhand von Bedingungen bestimmen, welche der Konfliktregeln zur Anwendung kommt [Hos93, Kün98]. Diese Methode wird jedoch aufgrund ihrer Komplexität relativ selten angewendet, um Positiv-Negativ-Konflikte zu lösen. Metaregeln werden meist eingesetzt, wenn Konflikte zwischen unterschiedlichen Politiken, welche bei der Überschreitung von Domänen-Grenzen auftreten, zu lösen sind oder Konflikte bezüglich des Management- oder Sicherheitsziels, welche auf Applikationsebene auftreten. Ein Beispiel dafür ist das Ponder-Projekt, das Metapolitiken

dazu verwendet, anwendungsspezifische Restriktionen, wie Separation-of-Duty, zu erreichen [DDL+01]. Die Metapolitiken in Ponder werden für Gruppen von Regeln festgelegt und können so die parallele Ausführung von Konflikt-Aktionen verhindern. In [Kün98] werden Metapolitiken dazu verwendet, Konflikte, die bei der Überschreitung von Politikdomänen auftreten, aufzulösen. Die Strategien, die dabei zur Konfliktlösung angewendet werden, entsprechen denjenigen aus [Hos92].

Da in der vorgestellten Politiksprache CDACL nur Positiv-Negativ-Konflikte auftreten können, ist ein Prioritätenansatz der geeignetste Weg. Der Metaregelansatz ist zu aufwendig und der Reihenfolgeansatz zu willkürlich von der Implementierung abhängig. Der Prioritätenansatz dagegen erlaubt eine benutzergesteuerte Prioritätsordnung sowie eine effiziente Implementierung.

Ein auf den ersten Blick sehr guter allgemeiner Ansatz zur Bestimmung von Prioritäten, der oftmals eingesetzt wird, stellt die Spezialisierung dar. Dieser ist jedoch durch die spezielle Scope-Festlegungen in CDACL nur sehr schwierig und aufwendig zu realisieren. Der Spezialisierungsansatz besagt: liegt der Scope einer Regel innerhalb des Scopes der anderen Regel, so dominiert die spezialisiertere Regel. Spezialisierter heißt, der Scope der spezialisierteren Regel bildet eine Untermenge des Scopes der anderen Regel. Durch die Möglichkeit in CDACL, den Regel-Scope über Mengenoperationen aus einzelnen Einheiten zu bilden, ist es ein sehr aufwendiges Problem festzustellen, welcher Scope in welchem enthalten sein könnte. Schon das in Abbildung 25 gezeigte „einfache“ Teilproblem für die Subjekt-Menge eines Scopes macht die Schwierigkeit und Komplexität deutlich.

Das Subjekt X ist ein Mitglied der Subjekt-Sets B und A2. Der Scope der betrachteten Regel enthält als Subjekt-Menge die Schnittmenge A der Sets A1 und A2, die andere Regel das Set B. Die Frage, die zu beantworten ist, lautet: gilt die Beziehung A $\subseteq$ B, B $\subseteq$ A oder keine der beiden. Diese Frage ist sowohl für die Subjektmenge, wie auch für die Aktionen- und Objektmenge zu entscheiden. Aus diesem Grunde, wird die Spezialisierung zur Festlegung der Regelpriorität in dieser Arbeit nicht eingesetzt.

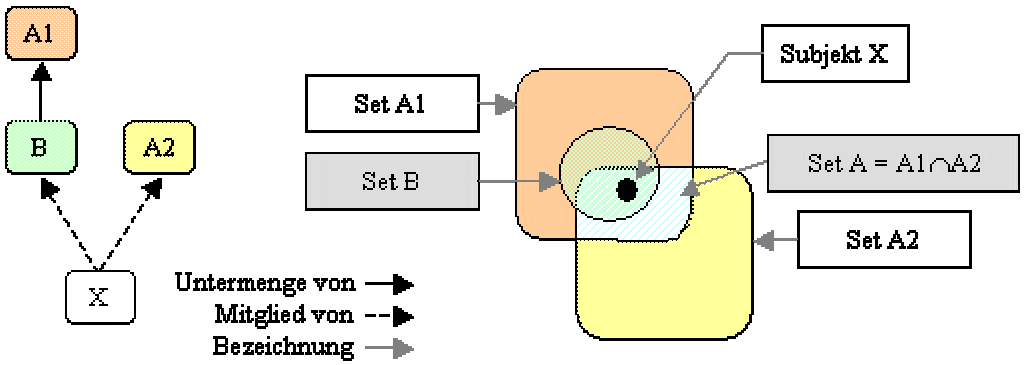

Abbildung 25 Spezialisierungsproblem

Algorithmen zur Konfliktauflösung, welche das Erstellungsdatum der Regel als Priorität einsetzen, erscheinen wenig sinnvoll, da es bei einer benutzerdefinierbaren Politik damit möglich wäre, dass ein beliebiger Benutzer eine Regel eines Sicherheits-

administrators außer Kraft setzen könnte. Ein Verfahren, welches den Status oder den Rang des Regel-Erstellers verwendet, um die Regelpriorität festzulegen, erscheint dagegen sinnvoll. Diese Variante kann mit geringem Aufwand über eine explizite Prioritäten-Lösung, wie sie im Folgenden vorgestellt wird, implementiert werden.

In einer kontextabhängigen Zugriffspolitik stehen semantische Konflikte, welche zur Laufzeit durch bestimmte Kontextkonstellationen auftreten im Vordergrund. Diese Konflikte können in den seltensten Fällen bei der Politikerstellung erkannt werden, da die Menge der Kontextzustände beliebig groß werden kann. Eine Überprüfung, ob bestimmte Zustände zu einem Konflikt führen, ist daher sehr aufwendig. In dieser Arbeit werden zwei Wege zur Vermeidung bzw. zur Lösung von Konflikten eingeschlagen: eine implizite und eine explizite Lösungsstrategie.

Implizite Lösungsstrategie

Der erste Weg ist die Einführung von Hierarchien von Regelbedingungen. Durch die Kombination von Regelbedingungen (Kontextbedingung; KB) über die booleschen Operationen „AND“, „OR“ und „NOT“ wird implizit eine „Konfliktlösungsstrategie“ bzw. eine Metapolitik implementiert (siehe Abbildung 26). Dabei werden Inkonsistenzen jedoch nicht aufgedeckt, sondern nur verdeckt. Es liegt also in der Obhut des Regel-Erstellers, seine Bedingungshierarchien so zu gestalten, dass diese sich nicht widersprechen.

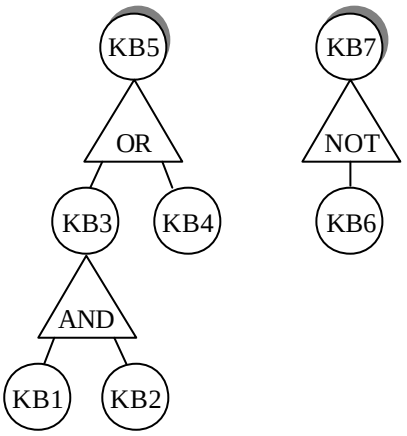

Abbildung 26 Bedingungshierarchien

Da es aber in der Realität so sein wird, dass Politikregeln nicht von einer einzelnen Person erstellt werden, werden mehrere Regeln und mehrere Politiken parallel existieren. Auch wird ist es nicht immer sinnvoll sein, alle Regeln durch Bedingungshierarchien zu verbinden. Die daraus entstehenden Konflikte können nur auf einer weiteren Ebene in Form einer „Metapolitik” gelöst werden.

Explizite Lösungsstrategie

Um Konflikte lösen zu können, die bei parallelen Regeln oder Politiken für den gleichen Scope-Bereich evtl. auftreten, wird jeder Regel eine Priorität zugeordnet. Die Priorität einer Regel entspricht der Priorität des Regel-Erstellers. Je höher die Rechte des Erstellers, umso höher die Priorität der Regel. Somit ist es beispielsweise nicht

möglich, dass ein einfacher Mitarbeiter Regeln eines Sicherheitsadministrators durch das Festlegen eigener Regeln außer Kraft setzt. Diese Standardeinstellung kann jedoch durch den Ersteller geändert werden: jeder Regel-Ersteller kann die Prioritäten seiner eigenen Regeln nach unten korrigieren und diesen so eine geringere Priorität als die standardarmäßig zugewiesene zuweisen.

Tritt ein Positiv-Negativ-Konflikt von Regeln gleicher Priorität auf, so wird, je nach Konfiguration des Systems, die positive oder negative Entscheidung favorisiert. Diese globale Strategie – eine übergeordnete, dem System inhärente Metapolitik - legt auch fest, ob eine Regelentscheidung „*undecided*“, die beispielsweise durch fehlende Kontextinformationen auftreten kann, als „*denied*“ oder „*allowed*“ interpretiert werden soll.

Die Realisierung der expliziten Lösungsstrategie erfolgt in der Implementierung des Datenmanagements und der Entscheidungskomponente des Zugriffkontrollsystems (siehe Abschnitt 8.4).

6.4 Sicherheitspolitiken in CDACL

Das zentrale Ziel dieser Arbeit war es, beliebige Kontextinformationen in die Entscheidungsfindung der Zugriffskontrolle einbeziehen zu können. Daher war das oberste Ziel bei der Entwicklung der Politiksprache CDACL die Fähigkeit, komplexe (Kontext-)Bedingungen spezifizieren zu können. Es sollte aber auch möglich sein, die gängigsten Sicherheitsmodelle in CDACL spezifizieren zu können. Diese Fähigkeit bildet die Grundlage dafür, dass evtl. an den Domänengrenzen auftretende Konflikte durch unterschiedliche Sicherheitsmodelle gelöst werden können, sofern diese in der gleichen Sprache spezifiziert sind, die das System versteht.

Anhand einiger ausgewählter Beispiele wird im Folgenden gezeigt, wie sich andere Politiken in CDACL spezifizieren lassen.

6.4.1 DAC

Eine benutzerbestimmbare Sicherheitspolitik (Discretionary Access Control Policy; DAC Policy) erlaubt den Benutzern, die Rechte an den Objekten, welche sie erzeugt haben, individuell zu vergeben und zurück zu nehmen. Jeder Benutzer hat an seinen Objekten alle Rechte.

Es existieren viele Vertreter einer DAC-Politik, doch basieren alle auf zwei wesentlichen Bedingungen: die erste erlaubt dem Besitzer eines Objekts, alle möglichen Aktionen darauf auszuführen, die zweite Rechte und damit eine Politik für seine Objekte festzulegen. Auf das aktuelle Ereignis wird mit „ce“ (current event) zugegriffen, auf die aktuelle Politik mit „this“.

```
<CDACPolicy>
   <PolicyLabel>DAC</PolicyLabel>
   <ACRuleBase>
      <ACRule Priority="5">
```

```
            <RScope/>
            <RContext>
                <OR>
                    <ScopePredicate>
                        <Equal>
                            <Var>ce.subject</Var>
                            <Var>ce.object.owner</Var>
                        </Equal>
                    </ScopePredicate>
                    <ScopePredicate>
                        <Equal>
                            <Var>this.owner</Var>
                            <Var>ce.object.owner</Var>
                        </Equal>
                    </ScopePredicate>
                </OR>
            </RContext>
        </ACRule>
    </ACRuleBase>
</CDACPolicy>
```

6.4.2 RBAC

Im Gegensatz zu herkömmlichen Sicherheitsmodellen wie beispielsweise DAC, welche das Subjekt in den Vordergrund stellen, steht bei rollenbasierten Sicherheitsmodellen die zu bearbeitende Aufgabe im Mittelpunkt. Die Entscheidung über die Zulässigkeit eines Zugriffs wird beispielsweise anhand der Rolle(n) des Benutzers getroffen, die dieser innerhalb der Organisation belegt.

Die Zugriffsrechte sind an die definierten Rollen, nicht an die Benutzer gekoppelt. Somit ist nach der Festlegung der Rechte für bestimmte Rollen, die Zuordnung von Benutzern zu den Rollen ausschlaggebend für die eigentliche Erteilung von Rechten. Jedem Benutzer ist eine Menge von Rollen zugeordnet, für die er dann autorisiert ist. Mit der Aktivierung einer Rolle, erlangt er deren Rechte.

Die Rechteerteilung in RBAC umfasst somit drei Schritte:

1. Die Spezifikation der Rollen und die Zuordnung von Benutzern zu den Rollen.
2. Die Aktivierung/Deaktivierung von Rollen für Benutzer.
3. Die Rechteüberprüfung anhand der aktiven Rollen eines Benutzers.

Die Spezifikation von Rollen und die Zuordnung von Benutzern zu Rollen sowie die Vergabe von Zugriffsrechten an Rollen erfolgen wie in Abschnitt 6.2.2.3 beschrieben.

Die Aktivierung einer Rolle für einen Benutzer ist dann erlaubt, wenn diese Rolle zu der Menge der autorisierten Rollen zählt und keine Rolle aktiviert ist, in deren Konfliktmenge sich die zu aktivierende Rolle befindet. Diese letzte Forderung garantiert die Einhaltung der Trennung von Aufgaben und Pflichten (Separation-of-Duty),

ein Prinzip, dass in RBAC unterstützt wird. Als Bedingungen in CDACL spezifiziert lauten beide Forderungen:

```
<CDACPolicy>
   <PolicyLabel>RBAC</PolicyLabel>
   <ACRuleBase>
      <ACRule Priority="5">
         <RScope>
            <Subjects/>
            <Actions><Action>activate</Action></Actions>
            <Objects>Object><oid>activateRoles</oid></Object>
            </Objects>
         </RScope>
         <RContext>
            <AND>
               <ScopePredicate>
                  <ElementOf>
                     <Var>this.action.arg[1]</Var>
                     <Var>this.subject.authRoles</Var>
                  </ElementOf>
               </ScopePredicate>
               <NOT>
                  <ScopePredicate>
                     <ElementOf>
                        <Var>this.action.arg[1]</Var>
                        <Var>this.subject.conflictRoles</Var>
                     </ElementOf>
                  </ScopePredicate>
               </NOT>
            </AND>
         </RContext>
      </ACRule>
   </ACRuleBase>
</CDACPolicy>
```

6.4.3 Regionorientierte Zugriffskontrolle

Die regionorientierte Zugriffskontrolle stellt nach Aussage des Autors Held eine spezielle Form des rollenbasierten Zugriffsmodells dar. Rollen werden jedoch nicht als Abbildung der Organisationsstruktur verstanden, sondern von Regionen. Der wesentliche Unterschied zu RBAC besteht darin, dass Objekte fest einer einzelnen Region zugeordnet sind, welche untereinander disjunkt sind. Wie in RBAC erhält ein Benutzer Rechte durch die Zuordnung zu einer Region.

6.4.4 Zustandsabhängige Sicherheitsspezifikation

Eine Spezifikation einer zustandsabhängigen Sicherheitspolitik nach [Eck97] erfolgt mittels regulärer Ausdrücke. Das Kernstück ist das Protokoll, das eine Folge von Schritten mit Sequenzen, Alternativen und Iterationen beschreibt. Schritte bestehen aus Aktivatoren, Exekutoren und Aktionen und entsprechen somit den in dieser Arbeit als Ereignisse bezeichneten Tripel. Protokolle stellen eine Untermenge der in Abschnitt 5.1.3 beschriebenen Ereignis-Muster dar, welche über Ereignis-Prädikate (siehe Abschnitt 6.2.2.4) und Protokolle (siehe 6.2.2.4) beschrieben werden können.

Kapitel 7

Kontextinfrastruktur

Seit dem Aufkommen des Context-aware Computing wurden unzählige prototypische Anwendungen entwickelt, die den Vorteil und die Nützlichkeit der Kontextabhängigkeit zeigen. Eine weite Verbreitung haben sie jedoch noch immer nicht erreicht. Dies liegt zum Teil daran, dass die Entwicklung solcher Anwendungen sehr schwierig, aufwendig und folglich teuer ist. Viele Prototypen beziehen ihre Kontextinformationen direkt von speziell für diese Anwendung entwickelten bzw. bereitgestellten Sensoren. Anderen Anwendungen ist dann die Verwendung dieser Kontextinformationen nicht ohne weiteres möglich. Die Kommunikation mit den Sensoren, der Zugriff auf die Daten und deren Verarbeiten müssen von jeder Anwendung neu implementiert werden.

Einige Ansätze, die dieses Dilemma ansatzweise lösen, wurden in Abschnitt 3.3.3 vorgestellt. Schilits PARCTAB sowie der Generic Context Server von Salber und Abowd stellen eine allgemeine Architektur dar, das Stick-e Document Framework von Brown sowie das Context Toolkit von Dey, Salber und Abowd – ebenfalls ein Framework - unterstützen die Entwicklung einer bestimmten Klasse von kontextabhängigen Anwendungen. Der Situated Computing Service von Hull und die MUSE Infrastruktur vom UCLA Department of Computer Science sind ein erster Schritt bei der Entwicklung einer Middleware. Jede dieser Arbeiten hat sich jedoch auf ein bestimmtes Problem oder ein bestimmtes Anwendungsszenario spezialisiert.

Die vorliegende Arbeit wurde von vielen Ideen in den genannten Arbeiten inspiriert. Der Grundgedanke, der dieser Arbeit jedoch zugrunde liegt ist, Kontextinformationen über eine Diensteinfrastruktur, welche das zugrunde liegende Raum-Konzept widerspiegelt, beliebigen Anwendungen zur Verfügung zu stellen.

In diesem Kapitel werden das Konzept und die einzelnen Komponenten der Kontextarchitektur beschrieben. In Abschnitt 7.1 wird ein Überblick über die Gesamtarchitektur gegeben. Eine detaillierte Beschreibung der beiden Hauptkomponenten Lokations- und Kontextdienst folgt in Abschnitt 7.2 und 7.3.

7.1 Aufbau der Kontextinfrastruktur

Betrachtet man den Datenfluss von Kontextinformationen (vgl. Abbildung 27), so können folgende Stationen und damit Kernfunktionen der Kontextinfrastruktur unterschieden werden:

- Sensoren ermitteln Werte, die so genannten Kontextdaten,
- schicken diese an den Kontextserver,
- welcher die Daten beobachtet, filtert, speichert und
- kontextabhängigen Anwendungen zur Verfügung stellt.
- Diese können die gewünschte Kontextinformation aktiv abholen oder
- sich eine Benachrichtigung schicken lassen, wenn ein bestimmtes Ereignis oder eine bestimmte Situation (kombiniertes Ereignis) eintritt.

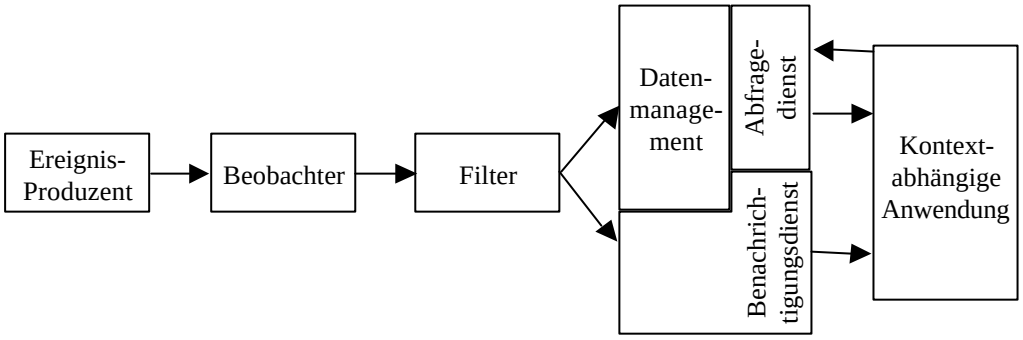

Abbildung 27 Datenfluss von Kontextinformationen

Damit können drei wesentliche funktionale Einheiten identifiziert werden, die sich in der Architektur widerspiegeln (Abbildung 28):

- einen *Ereignis-Dienst* mit Beobachtern (engl. Observer), Filterfunktionen und Benachrichtigungsdienst (engl. Notification Service), welcher die Kommunikationsschnittstelle sowohl zwischen Sensoren und Kontextdienst wie auch zwischen kontextabhängigen Anwendungen und Kontextdiensten bildet,
- eine *Datenhaltungs- und –managementkomponente* für Kontextinformationen und
- einen *Abfragedienst*.

Ereignisbasierte Infrastrukturen und Benachrichtigungssysteme wurden in den letzten Jahren intensiv untersucht und entwickelt. Es stehen einige kommerzielle Produkte wie auch Forschungsprototypen zur Verfügung. Ziel dieser Arbeit war es nicht, eine neue ereignisbasierte Infrastruktur zu entwickeln, sondern es wurde untersucht, in wie weit verfügbare Systeme für den Einsatz im Umfeld einer Kontext-Infrastruktur geeignet sind. Die wichtigsten zu untersuchenden Kriterien sind dabei die

Art der unterstützten Ereignistypen, die Ausdrucksstärke der Filterfunktionen sowie die Skalierbarkeit des Ereignissystems.

Kontextabhängige Anwendungen können mittels eines Benachrichtigungsdienstes über das Eintreffen spezifizierter Ereignisse informiert werden. Einige der kontextabhängigen Anwendungen – dazu gehören auch mobile Anwendungen, die nicht ständig über eine Netzverbindung verfügen – sind nicht daran interessiert, in bestimmten Abständen über Ereignisse benachrichtigt zu werden, sondern benötigen die Daten in unregelmäßigen Zeitabständen bei Bedarf. Ein Kontextsystem muss daher eine entsprechende Schnittstelle zur Abfrage von Kontextdaten sowie eine Abfragesprache (engl. Query Language) bereitstellen.

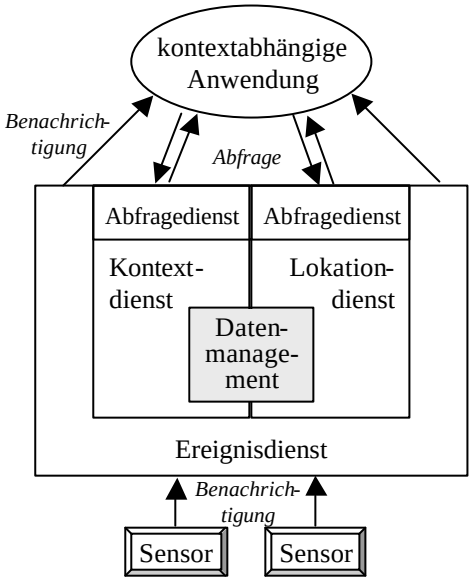

Abbildung 28 Kontextinfrastruktur

Wie in Kapitel 5 ausgeführt wurde, kann Lokation als eine spezielle Form von Kontextinformation betrachtet werden, von hoher Komplexität sein oder aber als Bezugs- und Identifikationsmerkmal für andere Kontextinformationen dienen. Viele kontextabhängige Anwendungen verwenden ausschließlich Lokation als Kontextinformation. Aus diesen Gründen wurde in dieser Arbeit der Lokationsdienst vom übrigen Kontextdienst[1] getrennt, um so die Administration zu erleichtern und den Lokationsdienst als „eingeschränkten" Kontextdienst nutzen zu können, wenn nur dieser benötigt wird. Es besteht jedoch eine sehr enge Bindung zwischen Lokationsdienst und Kontextdienst. Sowohl Objektdaten, Daten des Lokationsmodells und Ereignisdaten müssen verwaltet und geeignet gespeichert werden. Da sowohl der Kontextdienst wie auch der Lokationsdienst teilweise auf den gleichen Daten arbeiten, wird die Datenhaltungskomponente gemeinsam genutzt (siehe Abbildung 28).

[1] Der Lokationsdienst ist Teil des Gesamt-Kontextdienstes. Um jedoch besser unterscheiden zu können, wird der Begriff „Kontextdienst" im weiteren Text für den Teil des Gesamt-Kontextdienstes ohne den Lokationsdienst verwendet.

Die einzelnen Elemente der beiden Hauptkomponenten Lokations- und Kontextdienst der Kontextinfrastruktur werden in den folgenden Abschnitten genauer beschrieben.

7.2 Lokationsserver und Lokationsdienst

Informationen über den Aufenthaltsort einer Person, eines Gerätes oder eines Objekts werden als direkte Kontextinformation von einer steigenden Anzahl von ortsabhängigen[2] Anwendungen verwendet. Es ist unökonomisch und daher nicht sinnvoll, dass jede ortsabhängige Anwendung selbst den Aufenthaltsort bestimmt. Ein universeller Lokationsdienst kann die Entwicklung eines ortsabhängigen Dienstes vereinfachen und über den Aufenthaltsort der Anwendung hinaus räumliche Informationen liefern.

Ortsinformation hat aber nicht nur als direkte Kontextinformation Bedeutung, sondern wird auch indirekt für die Bereitstellung anderer Kontextinformationen benötigt. Ein Lokationsdienst muss daher nicht nur Informationen für ortsabhängige Anwendungen liefern, sondern auch eine enge Schnittstelle zum Kontextdienst anbieten. Ein Lokationsdienst sollte unabhängig von den Anwendungen sein, welche den Dienst in Anspruch nehmen, und die Hard- und Software-Details der Lokationssensoren gegenüber den Anwendungen verbergen.

Ein Lokationsserver wurde von Want [WH92] beschrieben als ein System bestehend aus vier Schichten: der Netzwerkkontrolle, der Repräsentation, der Datenbearbeitung und der Darstellungsschnittstelle. Die unterste Schicht bildet die Schnittstelle zu den Lokationssensoren, die die Kontextdaten liefern, die zweite Schicht komplettiert die Datenstruktur der Kontextdaten mit einem Zeitstempel, die dritte Schicht verarbeitet die Daten und die vierte Schicht ist für die Darstellung zuständig. Eine ähnliche Festlegung ist in [LK99] zu finden. Leonhardi und Kubach beschreiben darin einen Lokationsserver als ein „System, das Positionsinformationen sammelt, integriert, lagert und an interessierte Klienten weitergibt". Nach Leonhardt [Leo98] ist ein Lokationsdienst ein Dienst, welcher Informationen über die physischen Positionen von Objekten und Subjekten zur Verfügung stellt.

Die Notwendigkeit eines universalen Lokationsdienstes für ortsabhängige Anwendungen ist in der Literatur unbestritten. Frühe Arbeiten im Bereich des Ubiquitous Computing haben sich schon dieser Thematik angenommen: 1993 stellten Mike Spreitzer und Marvin Theimer in [ST93] eine einfache Architektur zur Bereitstellung von Ortsinformationen vor, mit besonderem Blick auf die Wahrung von Persönlichkeitsrechten (engl. Privacy). Diese Architektur sieht Ortsinformationen ausschließlich für Personen vor, welche durch User Agents repräsentiert werden. Dieser Dienst wurde speziell für eine Anwendung zur Lokation von Personen innerhalb eines Gebäudes konzipiert. Er ist weder skalierbar, erweiterbar noch

[2] Ortsabhängige Anwendungen sind eine spezielle Form von kontextabhängigen Anwendungen, welche jedoch ausschließlich die Kontextinformation „Lokation" verwenden. Wird von kontextabhängigen Anwendungen gesprochen, so sind ortsabhängige Anwendungen darin enthalten.

allgemein einsetzbar, doch die Überlegungen zum Schutz privater Daten wurden in einigen späteren Arbeiten aufgegriffen und in ähnlicher Weise gelöst.

Im Jahre 1998 beschreibt Giles John Nelson in seiner Dissertation [Nel98] verschiedene Lokationssensoren für den Innenbereich und stellt ein Architektur zur Unterstützung kontextabhängiger Anwendungen vor, wobei unter Kontext in [Nel98] ausschließlich Ortsinformationen verstanden werden. Der wichtige Punkt des Datenschutzes und der Wahrung der Privatsphäre wird in Nelsons Arbeit gänzlich außer Acht gelassen.

Kurz danach veröffentlichte Ulf Leonhard in seiner Dissertation [Leo98] allgemeine Überlegungen zum Thema Lokation. Er stellte verschiedene Orts- und Raummodelle vor, beschrieb die Anforderungen und betonte die Notwendigkeit einer Sicherheitsarchitektur zur Wahrung der Privatsphäre. Eine Implementierung der vorgestellten Überlegungen existiert jedoch nicht.

Andrew Martin Robert Ward beschreibt in seiner Dissertation [War98] einen neuentwickelten Lokationssensor sowie eine Architektur zur Bereitstellung der ermittelten Ortsinformationen für interessierte Anwendungen. Da das Hauptaugenmerk dieser Arbeit auf der Entwicklung eines Lokationssensors lag, wurde die Architektur einfach gehalten. Sicherheits- und Datenschutzaspekte wurden nicht berücksichtigt.

Neuere und sehr umfassende Arbeiten auf dem Gebiet eines universalen Lokationsdienstes sind im Projekt NEXUS [Nexus] der Universität Stuttgart zu finden [VSF+00, HKL+99, LK99]. Das Projekt NEXUS hat das Ziel, eine Plattform für ortsabhängige Anwendungen zu realisieren, mit dem Projektschwerpunkt auf der Unterstützung mehrerer unterschiedlicher Raummodelle. Bisher veröffentlichte Arbeiten berücksichtigen keine Sicherheits- und Datenschutzüberlegungen. Da das NEXUS-Projekt bis Ende 2002 läuft, ist noch kein abschließender Bericht vorhanden. Da diese Arbeiten sehr umfassend und vielversprechend sind, ist zu hoffen, dass die Ergebnisse auch in Form einer Implementierung allgemein verfügbar gemacht werden.

Alle zuvor genannten Arbeiten haben wichtige Grundlagen auf dem Gebiet eines universalen Lokationsdienstes für den Bereich Ubiquitous Computing geschaffen. Leider ist keine Implementierung öffentlich verfügbar. Der im Rahmen dieser Arbeit entwickelte Lokationsdienst, welcher in einigen Konzepten der zuvor beschriebenen Arbeiten ähnelt, wird in den folgenden Abschnitten genauer beschrieben.

7.2.1 Funktionale Architektur

Der im Rahmen dieser Arbeit entwickelte Lokationsdienst [RMB+99, MRH00] hat drei wesentliche Designziele, welche sich von den zuvor vorgestellten Architekturen unterscheiden:

- Abstimmung der Systemarchitektur an die darunter liegenden RaumComputer-Architektur (siehe Kapitel 5.2.3.1) um Skalierbarkeit und Effizienz zu erreichen,
- eine enge Anbindung an den Kontextserver ohne die Eigenständigkeit zu verlieren und

- die Gewährleistung von Datenschutz (engl. Data Protection) und Schutz der Privatsphäre (engl. Privacy).

Die Hauptaufgaben eines Lokationsdienstes sind

- die Verwaltung von Raummodellen und
- die Bereitstellung einer einheitlichen Schnittstelle für die Abfrage von Lokationsinformationen.

Abbildung 29 zeigt die wichtigsten funktionalen Systemkomponenten des entwickelten Systems, die zur Erfüllung dieser Aufgaben benötigt werden.

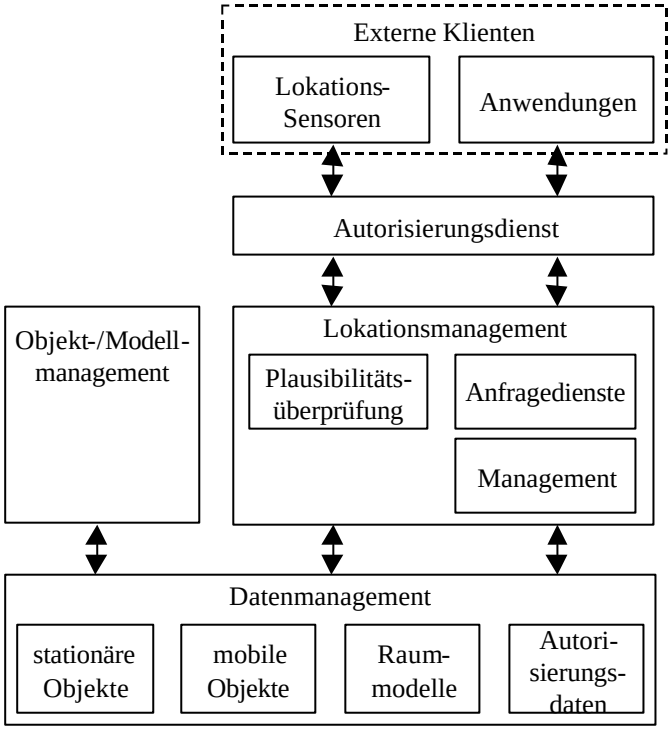

Abbildung 29 Funktionale Komponenten des Lokationsdienstes

Externe Klienten

Zu den externen Klienten, welche mit dem Lokationsserver kommunizieren, zählen zum einen die ortsabhängigen Anwendungen und zum anderen die Lokationssensoren, welche die Lokationsdaten liefern. Zu den möglichen ortsabhängigen Anwendungen zählten auch die kontextabhängige Zugriffskontrolle und der Kontextdienst. Beide Dienste sind gegenüber beliebigen ortsabhängigen Anwendungen, welche nur bedingt auf sensible Daten zugreifen dürfen, mit erweiterten Berechtigungen ausgestattet. Beide Anwendungen benötigen diese Lokationsinformationen für die Weiterverarbeitung.

Autorisierungsdienst

Um die Rechte verschiedener externer Klienten an den verschiedenen Lokationsdaten überprüfen zu können, wird ein Autorisierungsdienst benötigt. Dieser

gewährt oder verweigert den Zugriff auf sensible Lokationsdaten von Personen oder personalisierte Geräte.

Lokationsmanagement

Das Lokationsmanagement ist für die Zuordnung von Objekten zu Lokationen oder Positionen verantwortlich. Er greift dabei auf die gespeicherten Objekt- und Raummodell-Daten zurück. Das Lokationsmanagement stellt eine Schnittstelle für Lokationssensoren bereit, welche die Daten entgegen nimmt und auf Plausibilität überprüft, bevor sie an das Datenmanagement weitergegeben werden. Anfragen von externen Klienten werden über eine Abfrage-Schnittstelle entgegengenommen, ausgewertet und beantwortet.

Objekt- und Modellmanagement

Das Objektmangement erlaubt es, statische und mobile Objekte hinzuzufügen oder zu entfernen sowie Objektattribute inklusive Sicherheitsattribute zu verändern. Externe Lokationssensoren sind - je nach Sensorart - Teil der verwalteten statischen oder mobilen Objekte. Das Modellmanagement ermöglicht die Modifikation der Raummodelle, um diese an räumliche Veränderungen anzupassen.

Datenmanagement

Das Datenmanagement verwaltet die Daten über die Objekte, die Raummodelle und die Zugriffsrechte. Es ist zuständig für die Verteilung der Daten an andere Stationen, das Caching und die Speicherung. Die Verteilung und das Caching der Daten richtet sich nach Kriterien der Datennotwendigkeit, der Skalierbarkeit des Systems und des Datenschutzes. Neben der notwendigen kurzzeitigen Speicherung der aktuellen Lokationsdaten ist eine langfristige Aufzeichnung nur bedingt erlaubt. Eine Datenschutzpolitik legt fest, wie und wie lange Daten gespeichert werden dürfen.

7.2.2 Systemarchitektur

Bei der Umsetzung der funktionalen Anforderungen an den Lokationsdienst sind – neben den Sicherheits- und Datenschutzaspekten – die vorhandenen Hard- und Software-Gegebenheiten zu berücksichtigen. Da in dieser Arbeit die RaumComputer-Architektur als Ausgangsplattform gewählt wurde, bedeutet dies im Wesentlichen:

- ein vernetztes System von RaumComputern mit geringer Ressourcen-Kapazität,
- Vernetzung über Ethernet sowie einen
- zentraler RaumComputer-Server mit nicht beschränkender Ressourcen-Kapazität.

Ein RaumComputer verwaltet einen bestimmten Bereich, in der Regel einen Raum. Alle Lokationssensoren eines Raums sind mit dem lokalen RaumComputer vernetzt.

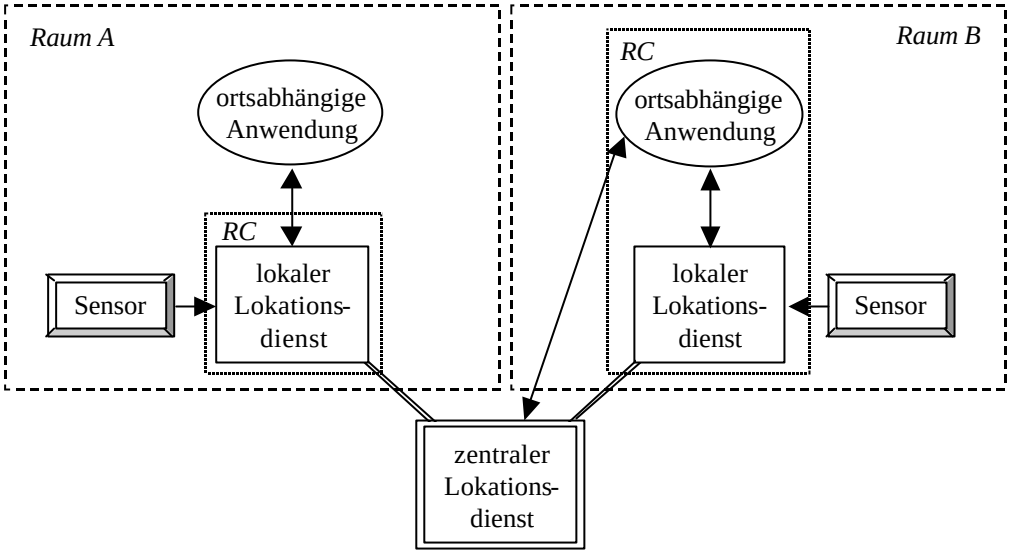

Abbildung 30 Systemübersicht des Lokationsdienst

Jedem RaumComputer (RC) ist ein lokaler Lokationsdienst (LD) zugeordnet (siehe Abbildung 30). Stationäre Lokationssensoren, welche sich im Verwaltungsbereich eines RCs befinden, melden ihre Daten an den lokalen Lokationsdienst. Dieser verwaltet die Lokationsinformationen für stationäre und mobile Objekte, welche sich in seinem Zuständigkeitsbereich befinden. Im lokalen LD werden ausschließlich die aktuellen Lokationsdaten vorgehalten, wodurch ein lokaler LD nur über das Wissen über den aktuellen Stand seines Zuständigkeitsbereichs verfügt. Mobile Sensoren, welche nicht eindeutig einem lokalen LD zugeordnet sind, melden ihre Daten einem beliebigen LD.

Im Gegensatz zu den lokalen Lokationsdiensten verfügt der zentralen Lokationsdienst über das vollständige Objekt-, Modell- und Datenmanagement. Hier werden alle Informationen der lokalen LDs gesammelt und gespeichert.

7.2.3 Lokationssensoren und Lokationssysteme

Zur Bestimmung von Orts- oder Positionsinformationen existieren verschiedene Sensoren und Systeme, teilweise speziell für diese Aufgabe entwickelt, teilweise als Add-on anderer Systeme realisiert. Die Lokationssensoren und -systeme können nach folgenden Kriterien unterschieden werden:

- verwendete Technologie (Funk, Infrarot, Ultraschall, etc.) und Methode (Triangulation, Bewegungserkennung, Signalstärke, zellenbasiert, etc.) verwendet wird,
- das mögliche Einsatzgebiet (innerhalb von Gebäuden, im Freien, beides),
- ob die Lokationsinformation im Klienten oder in der Lokationsinfrastruktur ermittelt wird und
- ob die Information als absolute oder relative Lokationsinformation bestimmt wird.

Ein spezielles System ausschließlich für den Einsatz im Außenbereich ist das Global Positioning System (GPS) [Dan01, GPS02a, GPS02b] bzw. das Differential Global Positioning System (DGPS), das die Position eines Empfängers mit einer Genauigkeit von weniger als 25m bzw. 1m in der Ebene bestimmt. Das GPS-System gehört zur Klasse der Lokationssysteme, in der die eigentliche Ortsbestimmung im Klienten stattfindet und die Infrastruktur die Position nicht kennt.

Im Gegensatz zu GPS ermöglicht das Wireless Andrew-System der Carnegie Mellon University [Hil99] eine Messung sowohl in Gebäuden wie auch (in begrenztem Umfang) außerhalb, da es Wireless LAN zur Positionsbestimmung nutzt. Der Klient misst die Feldstärke verschiedener Access Points und berechnet daraus die Position mit einer Genauigkeit von ±5m [SSS00].

Im Locust-Projekt [Locust, SKA97] am M.I.T. Media Lab wurde, basierend auf Infrarot, ein System zur Lokationsbestimmung für Personen entwickelt. Ein IR-Sender (der Locust), wird an der Decke befestigt und sendet per Infrarot eine ID aus, welche von mobilen Sensoren empfangen wird. Anhand der ID kann der Klient die Position der Person auf Zellengröße von maximal 6m im Durchmesser - entspricht meist einem Raum - bestimmen.

Ein weiteres System dieser Klasse ist GSM. Die Genauigkeit der Ortung bei GSM hängt stark von der Größe der GSM-Zellen ab. In einem Gebiet mit überlappenden Zellen, können Teilnehmer mit einer Genauigkeit von ±50m geortet werden. Mit zusätzlichen Informationen, wie beispielsweise Verkehrsdaten, kann dieses Ergebnis noch verbessert werden. Ein Beispiel dafür ist das vom BMBF geförderte TeleTravel-Projekt [TTS], das die Ziele hat, den Verkehr und die Mobilität besser zu erfassen, differenzierte Analysen des Verkehrsverhaltens durchzuführen und eine Verbesserung der Informationen für zielorientierte Verkehrsplanung zu erreichen.

Zur Klasse der Systeme, bei denen die Ortsbestimmung in der Infrastruktur erfolgt, gehören beispielsweise das ActiveBadge-System, das ParcTab-System, das ActiveBat-System oder das MediaCup-Projekt. Das ActiveBadge-System [WH92, WHF+92] wurde 1991 vom Olivetti Research Lab entwickelt. Es wird überwiegend zur Lokalisation von Personen verwendet. Die Positioniergenauigkeit hängt von der Dichte des Empfängernetzes und den physikalischen Eigenschaften von Infrarot ab. Das ParcTab-System von Xerox [WSA+95] basiert – wie das ActiveBadge-System – auf Infrarot. Das ActiveBat-System wurde 1998/1999 von AT&T entwickelt, da sich das ActiveBadge-System, das bis zu diesem Zeitpunkt von AT&T verwendet wurde, als unzureichend für die Anforderungen einer drei-dimensionalen Positionierung des Projekts herausstellte. Das ActiveBat-System benutzt Ultraschall zur Positionsbestimmung und liefert 3D-Informationen mit einer Genauigkeit von ±14cm. Im MediaCup-Projekt [GBK99, MediaCup] des Telecooperation Office (TecO) der Universität Karlsruhe wurde ein IrDA-Transceiver an der Decke befestigt, welcher Signale, die von einer speziell entwickelten Tasse mit IrDA-Transmitter, der MediaCup, ausgesendet werden, empfängt und so deren Anwesenheit registriert. Da zur Realisierung des MediaCup-Projekts Standard-Komponenten verwendet wurden, beträgt der von dem IrDA-Transceiver ausgeleuchtete Bereich und damit die Genauigkeit der Lokation einer MediaCup ca. 1,5m im Durchmesser.

Eine weitere Klasse von Lokationssystemen, welche bisher kaum Beachtung fand, verwendet Kameras und Bilderkennungssoftware zur Lokalisation von Objekten oder Personen. Dazu zählen beispielsweise das Visual Tag-System am M.I.T. Media Lab [SMR+96] oder das Findentity-System der Firma Thax Software GmbH [Findentity]. Im Findentity-System werden Objekte mit einem OpticMarker versehen, welcher über eine WebCam und entsprechender Software erkannt wird. Der Nachteil dieser Verfahren ist der benötigte Sichtkontakt des Sensors zum zu lokalisierenden Objekt.

Die Technologien zur Akquisition von Lokationsdaten können über verschiedene Qualitätmerkmale bewertet werden:

- die Qualität der gelieferten Daten,
- die Skalierbarkeit des Systems,
- der Einfluss der Umgebung,
- die Handhabbarkeit und
- die Kosten.

Unter Qualität von Lokationsdaten ist beispielsweise die Genauigkeit, die Anzahl der Dimensionen oder die Ausrichtung des Objekts im Raum zu verstehen. Skalierbarkeit erfordert eine Anpassung der Akquisition an die benötigte Qualität der Daten. Für die Qualität der Daten ist oftmals auch der Einfluss der Umgebung ausschlaggebend. Für verschiedene Technologien, wie beispielsweise Infrarot, ist Sichtkontakt notwendig. Aber auch der Einfluss der verwendeten Technologie auf die Umgebung sollte berücksichtigt werden. So ist eine dauerhafte Bestrahlung mit Radiowellen nicht erwünscht, genauso wenig wie eine vollständige Überwachung durch Videosysteme bei visuellen Lokationssystemen. Bei der Einführung eines Lokationssystems spielt auch die Akzeptanz durch den Benutzer bzw. den „Überwachten" eine entscheidende Rolle. Nur Systeme, welche unauffällig agieren, keine oder nur wenig Aktionen eines Überwachten erfordern und diesen in seiner Bewegungsfähigkeit nicht einschränken, werden Akzeptanz finden. Für die Kosten-Nutzen-Analyse spielen die Einzelkosten der verschiedenen Komponenten des Lokationssystems eine wichtige Rolle.

Für diese Arbeit wurde ein Positioniersystem für Personen und Objekte benötigt, das innerhalb von Gebäuden funktioniert, geringe Kosten verursacht und dessen Sensoren geringe Ausmaße besitzen, so dass sie an beliebige Objekte geheftet werden können [MHR01]. Aus diesem Grund konnte keines der beschriebenen Systeme verwendet werden, da sowohl die Kosten wie auch die Größe der Sensoren nicht den Anforderungen genügten. Nicht gefordert war eine genaue Lokalisierung von Personen wie sie beispielsweise durch das ActiveBat-System geliefert wird. Eine Positionierung mit der Genauigkeit von Räumen ist für dieses Projekt ausreichend. Ausgehend von diesen Anforderungen wurde in dieser Arbeit zur Realisierung die Radio Frequency-Identification (RFID) Technologie eingesetzt.

Die Radio Frequency (RF)-Technologie (siehe Abbildung 31) wird seit vielen Jahren zur Identifikation und zum Verfolgen von Objekten eingesetzt [Fin99]. Ein Radio Frequency Identification (RFID)-System besteht immer aus zwei Komponenten: den Transpondern, die an das zu identifizierende Objekt angebracht werden, und dem Lese-/Schreib-Gerät.

Der Transponder ist der eigentliche Datenträger. Er besteht aus einem integrierten Mikrochip und einem Koppelelement, der Antenne. Transponder existieren in den unterschiedlichsten Bauformen für unterschiedliche Anwendungszwecke: als Münzen, als Glasgehäuse für die Identifikation von Tieren, in Plastikgehäuse (beispielsweise im Schlüssel als Teil der elektronischen Wegfahrsperre), mit Ferittkern (für Metalloberflächen), in einer Uhr als Skipass, als Schlüsselanhänger, in Form von kontaktlosen Chipkarten (ID-1) oder Smartlabels.

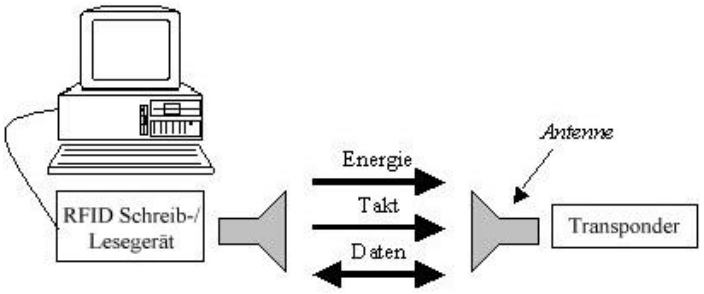

Abbildung 31 RFID-System

Unter Smartlabels versteht man papierdünne Transponder, die auf eine ca. 0,1mm dünne Plastikfolie aufgebracht sind. Diese werden dann oftmals mit einer Papierschicht laminiert und mit einem Kleber beschichtet. Smartlabels sind – im Gegensatz zu den meisten anderen Transpondern – passiv und besitzen keine eigene Energieversorgung (Batterie), so dass sie außerhalb der Reichweite eines Lese-/Schreib-Geräts inaktiv sind. Erst durch die kontaktlose Übertragung von Energie im Ansprechbereich des Lese-/Schreib-Geräts wird der Transponder aktiv. Neben der benötigten Energie werden auch Takt und Daten durch die Antenne zum Transponder übertragen.

Ein RFID-Leser beinhaltet typischerweise ein Hochfrequenzmodul (Sender und Empfänger), eine Kontrolleinheit sowie ein Koppelelement (Antenne) zum Transponder oder Tag. In der Regel sind die Lese-/Schreib-Gerät mit einer zusätzlichen Schnittstelle für die Verbindung zu einem anderen System (PC, Handheld, Automatensteuerung, etc.) ausgestattet, um die gelesenen Daten weiterzuleiten oder Daten zum Schreiben zu empfangen.

7.2.4 Datenakquisition

Die Charakteristika der Lokationssensoren, wie sie in Abschnitt 7.2.3 beschrieben wurden, nehmen in unterschiedlichem Umfang Einfluss auf die Datenakquisition. Während die verwendete Technologie und das Einsatzgebiet nur in geringem Maße die Datenakquisition bestimmen, spielen der Ort der Lokationsbestimmung und die Art der Lokalisationsdaten eine wesentliche Rolle bezügliche des Datenschutzes, der Datenhaltung und –bereitstellung.

Die Bestimmung des Aufenthaltsorts im Endgerät hat den Vorteil, dass keiner außer dem Klienten selbst den Aufenthaltsort kennt, da die Infrastruktur diese nicht ableiten kann. Diese Art der Lokationsbestimmung ist aus Sicht des Datenschutzes optimal, da die Datenhaltung verteilt erfolgt. Die Probleme bzw. Nachteile dabei sind,

dass außer den Anwendungen auf dem Klienten keiner unmittelbar auf die Lokationsdaten zugreifen kann, es schwierig ist, räumliche Beziehungen zwischen den lokalisierten Objekten herzustellen und der Klient über die Daten des zugrunde liegenden Raummodells verfügen muss, um die Position bestimmen zu können.

Wird der Aufenthaltsort einer Person oder eines Objekts in der Infrastruktur ermittelt, spricht man auch von Tracking. Tracking wirft Skalierungsprobleme sowie Sicherheits- und Datenschutzprobleme auf. Durch die zentrale Verarbeitung der Lokationsdaten ist es jedoch einfacher, Beziehungen zwischen Objekten und Personen abzuleiten. Der Zugriff auf Lokationsdaten anderer (mobiler) Objekte kann schneller erfolgen, da die direkte Nachfrage bei diesen Objekten entfällt.

Die Information des Aufenthaltsorts – auch Positions- oder Lokationsdatum genannt – muss, um in Anwendungen weiter verwendet werden zu können, neben der Angabe der Position auch die Angabe des zu positionierenden Objekts, den Zeitpunkt der Positionsermittlung und das verwendete Raummodell beinhalten. Liegt dem Gesamtsystem nur ein Raummodell zugrunde, so kann diese Angabe entfallen, da sie implizit festgelegt ist.

Die Angabe der Position erfolgt in geometrischen Raummodellen als Punkt in einem N-dimensionalen Koordinatensystem, beispielsweise durch Angabe der Längen- und Breitengrade. In symbolischen Raummodellen dagegen werden Positionen über Werte eines festgelegten, endlichen Wertebereichs bestimmt.

Das in dieser Arbeit realisierte RFID-System zur Lokalisation von Objekten und Personen besteht aus einem oder auch mehreren RFID-Lesern als Lokationssensoren (LS) pro Raum. Diese sind an den diesen Raum verwaltenden RaumComputer (RC) angeschlossen (Abbildung 32).

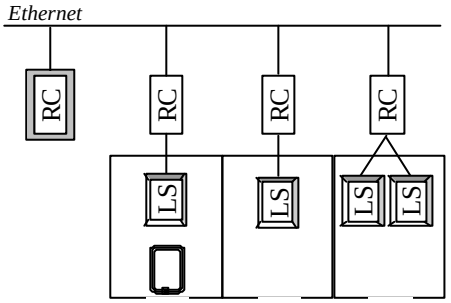

Abbildung 32 Lokationsarchitektur

Die Software zur Datenakquisition besteht aus drei Schichten (siehe Abbildung 33):

- der Schnittstelle zu den verschiedenen Lokationssensoren,
- einer Datenabstraktionsebene, welche die Rohdaten von den Lokationssensoren (LS) aufarbeitet und an den Lokationsdienst (LD) in einem Sensor-unabhängigen Format übergibt, und
- der Konsistenzprüfung.

Die Sensor-Schnittstelle (Treiber) und die Datenabstraktion sind durch den so genannten Generator realisiert. Der Generator ist ein Stück Software, welches ein oder mehrere Sensoren verwaltet, die Daten vom Sensor abfragt oder entgegennimmt, diese aufarbeitet und an das Lokationssystem weiterreicht. Er ist daher vom verwendeten Sensor abhängig implementiert.

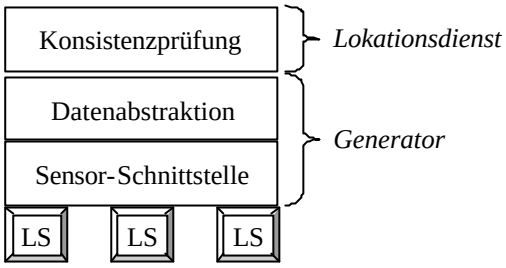

Abbildung 33 Softwarekomponenten der Datenakquisitation

Die Konsistenzprüfung erfolgt im Lokationsdienst, da nur hier genügend Daten zur Verfügung stehen. Werden bei der Datenakquisition RFID-Sensoren in Form von Schleusen im Eingangsbereich eines Raumes verwendet, so muss dies bei der Verarbeitung der Lokationsdaten berücksichtigt werden. Aufgrund der technologischen Beschränkungen durch RFID umfasst der überwachte Bereich nicht den ganzen Raum, sondern nur den Türbereich. Damit bedeutet den „Aufenthaltsort" eines Objekts oder einer Person zu ermitteln, eine Schlussfolgerung aus dem „Passieren" einer Schleuse zu ziehen. Ob dieses Passieren von außen nach innen („betreten") oder von innen nach außen („verlassen") bedeutet, kann nur durch logische Schlussfolgerungen durch das System ermittelt werden, da RFID als Technologie keine Möglichkeit bietet, die Richtung zu ermitteln.

Ein spezieller RFID-Generator ermittelt in kurzen Zeitintervallen die IDs aller RF-Tags, welche sich im überwachten Bereich befinden. Die gelesenen Tag-IDs werden mit der Identifikation des Lesegeräts und einem Zeitstempel an den lokalen Lokationsdienst gemeldet (siehe Abbildung 34). Wird ein Tag in mehreren aufeinander folgenden Zeitintervallen erkannt, so wird dies vom Generator, welcher die Daten des Sensors aufnimmt und vorverarbeitet, als „Verweilen in der Überwachungszone" interpretiert. Liegen jedoch mehrere Zeitintervalle dazwischen, in denen das RF-Tag nicht gelesen wurde, wird es zweimal gemeldet und als „Passieren" bewertet.

Die eigentliche Lokationsbestimmung erfolgt über die eindeutige Zuordnung von Tag-IDs zu Objekten bzw. Personen, die festgelegte Zuordnung von Sensor zu Raum und dem zugrunde liegenden Raummodell. Wie auch das ActiveBadge-System liefert das implementiert RFID-System somit Lokationsinformationen in Form von „Objekt X ist in Raum Y enthalten".

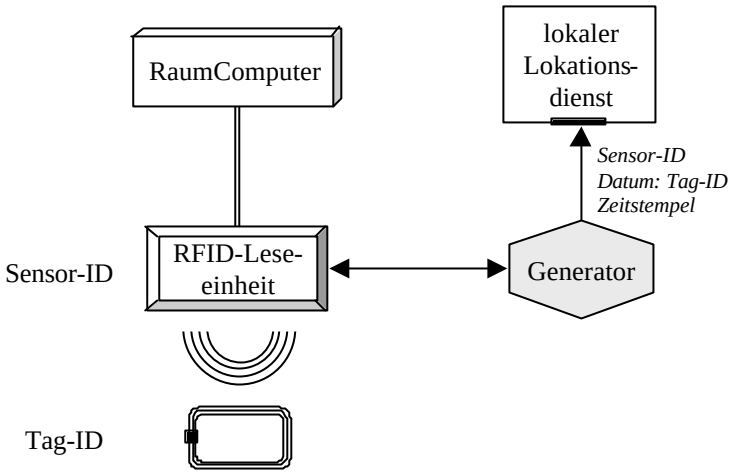

Abbildung 34 Datenakquisition

7.2.5 Plausibilitätsüberprüfung

Lokationsinformationen werden zu diskreten Zeitpunkten ermittelt, die RFID-Technologie arbeitet nicht 100%ig fehlerfrei und Umwelteinflüsse, wie das Stören durch metallische Gegenstände, können das Erkennen der RF-Tags verhindern. Diese Probleme führen zu Ungenauigkeiten und Fehler in der Datenakquisition und damit zu Lücken in den Lokationsinformationen.

Da die verwendete RFID-Technologie keine Aussage über die Bewegungsrichtung erlaubt, kommt es zu Fehlinterpretationen sobald ein Lokationsdatum verloren geht. Wird nicht erkannt, dass eine Person oder ein Gegenstand den Raum betritt bzw. verlässt, liegt eine fehlerhafte Information vor. Dies wird in der Regel nach spätesten zwei weiteren Lokationsmeldungen für dieses Objekt erkannt. Dazu werden im realisierten System zwei Überprüfungen vorgenommen und die Daten im Fehlerfall korrigiert:

1. Befindet sich das Objekt/die Person laut Lokationssystem innerhalb eines Raums A und betritt laut neuer Sensorinformation Raum B, so wurde eines der beiden Ereignisse „Betreten“ oder „Verlassen“ von Raum A nicht erkannt. Es wird das aufgezeichnete Ereignis für Raum A als fehlerbehaftet gekennzeichnet und die aktuelle Position auf „Raum B“ gesetzt.
2. Die Lokationsinformation einer Person, welche einen Raum über Nacht laut Protokoll nicht verlassen hat, verliert ihre Gültigkeit, da davon ausgegangen wird, dass Personen das Firmengebäude über Nacht verlassen.

7.2.6 Zuverlässigkeit von Lokationsdaten

Um Lokationsdaten angemessen verwenden zu können, ist eine Bewertung der Korrektheit und der Sicherheit der Lokationsdaten notwendig. Während manche ortsabhängige Anwendungen kein Problem mit falschen, veralteten oder nicht vorhandenen Ortsangaben haben, ist für eine Sicherheitsanwendung die Korrektheit der Daten ausschlaggebend. Daher ist es notwendig, dass ein Lokationsdienst mit den angeforderten Lokationsdaten auch eine Einschätzung der Zuverlässigkeit der Daten liefert.

Während bei geographischen Modellen wie GPS die Korrektheit durch Ungenauigkeiten bei der Messung beeinträchtigt wird, welche bekannt und daher spezifizierbar sind, ist die Korrektheit im vorgestellten RFID-System durch eine richtige Interpretation der Lokationsereignisse bedingt. Weisen die Lokationsereignisse Lücken auf, sind also unvollständig, kommt es zu fehlerhaften Interpretationen und damit zu falschen Lokationsinformationen. Es ist also ein Maß für die Zuverlässigkeit zu spezifizieren.

Die Zuverlässigkeit einzelner Lokationsdaten wird anhand des Objekttyps und der verstrichenen Zeit seit dem Eintritt des letzten Lokationsereignisses dieses Objekts festgelegt. Stationäre (z.B. im Raum montierte) und quasi-stationäre Objekte (z.B. nicht-mobile Rechner, Drucker) erhalten die Zuverlässigkeit eins. Da nicht erwartet wird, dass sich diese Objekte bewegen, erfolgt keine Reduzierung der Zuverlässigkeit der Lokationsinformation.

Lokationsinformationen mobiler Objekte und Personen verlieren an Zuverlässigkeit in Abhängigkeit der Zeit (in Minuten):

trustworthiness $tw: t \times aging_O \to [0.5, 1]$

$tw(t, aging_O) = 1 - min\ (0.5, t / aging_O)$

Der Alterungsfaktor $aging_O$ ist für Personen (Beispiel: „1440“, damit wird der Wert nach 12 Stunden ungültig) und jeden Typ von mobilen Objekten festzulegen. Über das Altern kann die Zuverlässigkeit *trustworthiness* maximal bis 0,5 reduziert werden. Dieser Wert besagt, dass mit einer Wahrscheinlichkeit von 50% die Lokationsinformation korrekt ist. Eine Zuverlässigkeit von Null kann eine Lokationsinformation nur durch eine Plausibilitätsüberprüfung und dem daraus folgenden expliziten Setzen auf „ungültig“ erlangen.

7.2.7 Lokationsabfragen

Die Schnittstelle von externen Klienten zum Lokationsdienst wird durch die möglichen Lokationsabfragen bestimmt. Lokationsabfragen können in zwei große Klassen eingeteilt werden: „Aufenthaltsort von Objekt X?“ und „welche Objekte befinden sich in der Lokation Y?“. Je nach verwendetem Raummodell und Akquisitionssystem können diese weiter konkretisiert werden.

Der realisierte Lokationsdienst unterstützt die folgenden Lokationsabfragen:

- *locationOfObject: objectId ´ accuracy ® location ´ trustworthiness*

  Diese Anfrage liefert den aktuellen Aufenthaltsort des spezifizierten Objekts mit einer Auflösungs- bzw. Abstraktionsgenauigkeit von *accuracy*. Der Faktor *trustworthiness* spezifiziert die Zuverlässigkeit der zurückgelieferten Lokationsinformation.

- *locationOfPerson: personId ´ accuracy ® location ´ trustworthiness*
  locationOfPerson: name ´ accuracy ® location ´ trustworthiness

  Personen können über eine eindeutige Identifikationsnummer oder aber über einen eindeutigen Namen identifiziert werden.

- *objectsAtLocation: location ´ objectType* ® (objectId ´ trustworthiness)**

  Die Abfrage o*bjectsAtLocation* liefert eine Liste von Objekten zurück, welche sich im spezifizierten Raum befinden. Für jedes Objekt wird die Zuverlässigkeit der Lokationsaussage festgelegt. Eine Einschränkung auf bestimmte Objekttypen ist möglich.

- *peopleAtLocation: location ® (personId ´ trustworthiness)* ´ numberOfInvisibles*

  Die Abfrage *peopleAtLocation* liefert eine Liste von Personen zurück, welche sich im angegebenen Raum befinden. Für jede Person wird eine Aussage über die Zuverlässigkeit der Lokationsinformation getroffen. Da Personen die Möglichkeit besitzen, ihren Aufenthaltsort zu verschweigen (siehe Kapitel 7.2.8), kann die Antwort unvollständig sein. Die Anzahl der Personen, welche die Antwort verweigern, drückt sich im Faktor *numberOfInvisibles* aus.

- *peopleNearByPerson: personId ´ accuracy ® (personId ´ trustworthiness)* ´ numberOfInvisibles*

  Die Anfrage *peopleNearByPerson* entspricht der Anfrage *peopleAtLocation* mit dem Unterschied, dass die Lokation indirekt über die spezifizierte Person und dem Abstraktionsniveau *accuracy* der Lokationsangabe bestimmt ist.

7.2.8 Datenschutz

Ortsabhängige Anwendungen verwenden oftmals den aktuellen Aufenthaltsort der Person, die die Anwendung bedient, bzw. des Geräts, auf dem die Anwendung läuft, als Eingabe. Beispiele dafür sind Navigationsgeräte im Auto, welche die Position des Autos benötigen, um den Weg zu berechnen, passende Staumeldungen zu finden oder die nächste Tankstelle anzuzeigen, oder auch das Sm@rtLibray-Projekt [MHR01, SLP], das in Abhängigkeit des Aufenthaltsorts dem Benutzer unterschiedliche, der Umgebung angepasste Dienste anbietet. Prinzipiell können diese Art von Anwendungen, die die eigene Position verwenden, so gestaltet werden, dass die Lokationsin-

formation der Infrastruktur und damit anderen Systemteilnehmern verborgen bleibt. Anwendungen wie der Conference Assistant [DSA+99], das Call Forwarding [WHF+92] oder auch die in dieser Arbeit entwickelte kontextabhängige Zugriffskontrolle verwenden jedoch die Lokationsinformationen von anderen Personen und Geräten, die in der Regel über einen Lokationsdienst bereitgestellt werden.

Durch die Aufzeichnung, Verarbeitung und Speicherung von Lokationsinformationen von Personen wie auch von mobilen Geräten wie beispielsweise PDAs oder Notebooks, welche meist einer Person zuordenbar sind, ist es möglich, Bewegungsprofile zu erstellen. Das „Big Brother"-Szenario einer totalen Überwachung ist zu befürchten. Geschieht diese Überwachung innerhalb einer Organisation, so ist es prinzipiell möglich, anhand dieses Bewegungsprofils den Arbeitnehmer zu überwachen und Rückschlüsse auf seine „Produktivität" zu ziehen. Geschützt durch das Arbeitsrecht ist dies aber in Deutschland nicht zulässig, genauso wenig wie die Erstellung von Bewegungsprofilen, die gegen das im Datenschutzgesetz verankerte Selbstbestimmungsrecht verstoßen [BDSG01]. Es ist somit nicht nur das Bedürfnis der Anwender nach Privatsphäre zu stillen, sondern auch gesetzliche Auflagen und Rahmenbedingungen einzuhalten.

Das Problem der Sicherheit und des Datenschutzes eines Lokationsdienstes wurde schon 1993 von Spreitzer und Theimer von XeroxPARC aufgegriffen. Sie argumentieren in [ST93, ST94], dass unterschiedliche Umgebungen unterschiedliche Sicherheitsstufen benötigen, und propagieren den Ansatz eines User Agents. Dieser unterliegt der Kontrolle des Benutzers und kontrolliert die Weitergabe der sensitiven Daten über Standard-Zugriffskontrollmechanismen.

Einen anderen Ansatz verfolgen Leonhardt und Magee in [LM97, LM98, Leo98, Mag01]. Sie setzen auf drei Barrieren: Access Policies, Visibility Policies und Anonymity Policies. Access Policies (oder Zugriffspolitiken) legen fest, welche Lokationsaktionen von welchen Personen beobachtet werden dürfen. Eine Visibility Policy stellt einen Filter dar, welcher festlegt, mit welcher Detailgenauigkeit Ortsinformationen aufgezeichnet werden. Über eine Anonymity Policy können verschiedene Pseudonyme für Personen spezifiziert werden, welche in dafür festgelegte Regionen verwendet werden sollen.

Der Ansatz von Spreitzer und Theimer [ST93, ST94] hat den Vorteil, dass die Kontrolle der sensitiven Lokationsdaten bei der Person selbst liegt, Nachteil ist jedoch, dass Standard-Zugriffskontrollmechanismen eingesetzt werden, die den Zugriff auf die Daten nicht situationsabhängig erlauben können. Der Ansatz von Leonhardt und Magee erlaubt die Spezifikation komplexer Politikregeln. Diese müssen jedoch alle Eventualitäten vorhersehen und eine entsprechende Regel dafür festlegen. Diese Komplexität erschwert die Handhabung und ist zu unflexibel, um auf sich ständig ändernde Gegebenheiten zu reagieren. Die Festlegung von unterschiedlichen Aufzeichnungsintervallen erscheint sehr aufwendig und wenig sinnvoll, da die aufzeichnende Infrastruktur bzw. die Sensoren für jeden Anwender andere Intervalle berücksichtigen müsste, was einen enormen Administrationsaufwand verursachen würde. Bei längeren Aufzeichnungsintervallen sinkt die Korrektheit und steigt die Unsicherheit der Lokationsdaten, womit die Verwendung in ortsabhängigen Anwendungen fragwürdig wird.

Die Verwendung von verschiedenen Pseudonymen zur Verschleierung der Identität ist nur in solchen Anwendungen sinnvoll, in denen eine Aufzeichnung von Aktivitäten aus Sicherheitsgründen notwendig ist. Bei einem Missbrauch ist es diesen Anwendungen dann möglich, im Nachhinein aus den Pseudonymen die eigentliche Identität der Person aufzudecken. Ein Lokationsdienst zählt jedoch nicht zu dieser Art von Anwendung. Ein Lokationsdienst ermöglicht die Abfrage der eigenen Position, wofür weder eine Identität noch ein Pseudonym benötigt wird, oder aber die Frage nach der Position von bekannten Personen. Für diesen Fall benötigt der LD die korrekte Identität der Person oder aber das aktuell verwendete Pseudonym. Dieses muss der anfragenden Person bekannt sein, da ansonsten keine Zuordnung zwischen der Lokation und dem Aufenthaltsort hergestellt werden kann. Die Verwendung von Pseudonymen erscheint bei einer seriellen Rollenaufteilung wie beispielsweise Privatmann oder Mitarbeiter sinnvoll, jedoch nicht eine zeitlich verzahnte, da der administrative Aufwand für die Person zum Wechseln der Pseudonyme unangebracht hoch wäre.

Der Vorteil von Pseudonymen ist, dass der LD nicht über das gesamte Profil eines Nutzers verfügt, sondern „nur“ über das Teilwissen der einzelnen Pseudonyme. Da es aber schwierig ist, keine Datenspuren beispielsweise durch Verwendung von persönlichen, ebenfalls überwachten Gegenständen (z.B. PDA, Notebook) zu hinterlassen, welche die Verknüpfung der einzelnen Profile ermöglichen, ist dieser Vorteil zweifelhaft.

Ein Ziel dieser Arbeit war es auch, ein Konzept zum Schutz der Privatsphäre zu entwickeln, das zum einen die Privatsphäre des Einzelnen schützt und zum anderen aber die Möglichkeiten ortsabhängiger Anwendungen so wenig wie möglich einschränkt. Um dieses zu erreichen, ist es notwendig, dem Einzelnen die Kontrolle über seine Daten zu übergeben und den Zugriff darauf flexibel und situationsgerecht gestalten zu können.

In dieser Arbeit werden die beobachteten „Objekte” in drei Schutzklassen eingeteilt:

1. Personen,
2. den Personen fest zugeordnete, mobile Objekte und
3. beliebige mobile oder stationäre Gegenstände und Geräte (ohne die personalisierten Objekte der zweiten Gruppe).

Zum Schutz der Lokationsdaten von Personen und personalisierten Geräten werden drei Prinzipien angewendet:

- Offenlegung der aktuellen lokalen Daten,
- Schutz der sensitiven Daten durch einen Persönlichen Sicherheitsmanager (engl. personal security manager) [RMB+99] und
- Verschlüsselung.

Das realisierte Datenschutzkonzept und die Funktionsweise des Persönlichen Sicherheitsmanagers wird im Folgenden erläutert (siehe Abbildung 35).

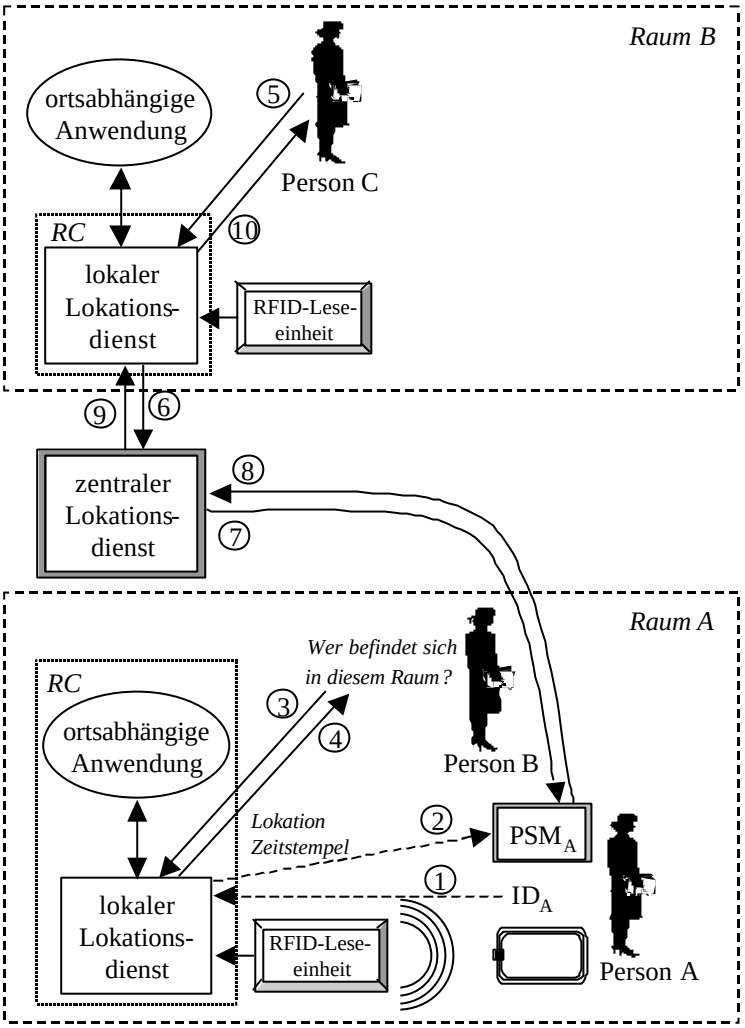

Abbildung 35 Datenschutzkonzept für personenbezogene Lokationsdaten

Betritt eine Person A einen überwachten Raum, so wird deren Aufenthaltsort im lokalen Lokationsdienst registriert (Schritt 1) und an den Persönlichen Sicherheitsmanager (PSM) der Person A weitergemeldet. Wird eine Anfrage nach dem Aufenthaltsort von Person A aus dem gleichen Raum an den lokalen LD gestellt (Schritt 3), so liefert der lokale LD die Lokationsdaten (Schritt 4) ohne Einschränkungen aus. Damit wird die Tatsache, dass eine Person B im gleichen Raum wie Person A durch den visuellen Kontakt den Aufenthaltsort von A ermitteln kann, nachgebildet. Es werden lokal ausschließlich aktuelle Daten ohne Einschränkungen angeboten, jedoch keine Historie-Daten. Diese werden wie eine Anfrage aus einem anderen Raum behandelt.

Stellt eine Person C in einem anderen Raum eine Lokationsabfrage nach Person A (Schritt 5), so leitet der zuständige lokale LD des Bereichs, in dem sich Person C befindet, diese Anfrage mit der Identität des Anfragenden an den zentralen LD (Schritt 7), welcher diese wiederum an den PSM von Person A weiterleitet (Schritt 8). Der PSM von A erteilt die gewünschte Auskunft oder verweigert diese (Schritte 9-11).

Der PSM verwaltet die Lokationsdaten der Person und speichert sie asymmetrisch verschlüsselt ab, so dass auch ein Administrator mit globalen Rechten, welcher unverschlüsselte Daten in der Regel (missbräuchlich) lesen könnte, keinen Zugriff erlangen kann. Der erlaubte Zugriff auf die Daten wird über eine konfigurierbare Zugriffspolitik geschützt.

Auf diese doch sehr aufwendige Prozedur eines Persönlichen Sicherheitsmanagers wird bei der Verwaltung von Lokationsdaten von personalisierten Gegenständen verzichtet. Im Gegensatz zu Personendaten werden diese an den zentralen LD gemeldet, der sie mit seinem Schlüssel verschlüsselt speichert. Nur der aktuelle Aufenthaltsort eines personalisierten Objekts kann ohne Zugriffsbeschränkung abgefragt werden. Hier greift jedoch wiederum eine vom Systemverwalter konfigurierbare Zugriffspolitik.

7.3 Kontextdienst und Kontextserver

Der Kontextdienst als Teil der Kontextinfrastruktur besteht aus den drei funktionalen Hauptkomponenten

- Ereignisdienst,
- Datenmanagement und
- Abfragedienst.

Für all diese Komponenten existieren kommerzielle Produkte und Prototypen, doch keine, welche alle drei Bereiche abdecken. In den folgenden Abschnitten werden unterschiedliche Systeme und Produkte vorgestellt und auf ihre Einsetzbarkeit in einem Kontextsystem untersucht.

Eine wichtige Entscheidung bei der Entwicklung eines Kontextsystems ist es, wie eng die Bindung zwischen den drei funktionalen Einheiten Ereignisdienst, Datenmanagement und Abfragedienst sein soll. Wird der Ereignisdienst integraler Bestandteil des Kontextdienstes oder als eigenständiger Dienst „nur" genutzt.

Bei einer losen Kopplung, in welcher der Kontextdienst den Ereignisdienst als Kommunikationsplattform mit dessen Möglichkeiten nutzt, kann auf Standardprodukte zurückgegriffen werden. Es ist kein Implementierungsaufwand notwendig, doch entsteht eine Abhängigkeit von der Produktpolitik. Erscheint eine neue Software-Version, so ist evtl. eine Anpassung notwendig. Es können ausschließlich die Möglichkeiten, die das Produkt anbietet, genutzt werden. Einschränkungen bezüglich der Ereignistypen, der Filter, der Verteilung und Skalierbarkeit sind gegeben, da ein Produkt in der Regel nicht auf ein spezielles Anwendungsgebiet abzielt.

Eine enge Kopplung, in welcher der Ereignisdienst durch eine vollständige Integration Teil des Kontextdiensts wird, erfordert in der Regel einen hohen Entwicklungsaufwand. Die Vorteile sind jedoch geringerer Kommunikationsaufwand, Einsparungen durch die enge Anbindung des Datenmanagements an den Ereignis-Beobachter, bessere Unterstützung der benötigten Ereignistypen und an das Problem angepasste Filterfunktionen. Eine Portierung auf neue Plattformen ist schneller, wenn auch mit gewissem Aufwand möglich.

In dieser Arbeit wird im Wesentlichen die zweite Alternative der engen Kopplung präferiert, da nur so die benötigte Effizienz und die Unterstützung des Raumkonzepts durch Verteilung der anfallenden Kontextdaten erreicht werden kann. Um jedoch die Vorteile vorhandener Ereignissysteme nutzen zu können, wurden verschiedene Systeme verglichen und auf deren Einsatzmöglichkeit untersucht. Aufbauend auf den gewonnenen Erkenntnissen wird eine Lösung zur Realisierung eines Kontextsystems auf Basis eines existierenden Ereignisdienstes vorgestellt.

7.3.1 Ereignisdienst

Mit der Zunahme der Mobilkommunikation und immer größeren verteilten Systemen, wie sie auch im Ubiquitous Computing vorliegen, steigt die Notwendigkeit der asynchronen, anonymen 1-zu-m-Kommunikation. Ereignismodelle sind anwendungsunabhängige Infrastrukturen, welche eine ereignisbasierte Kommunikation ermöglichen, in denen Ereignis-Erzeuger asynchron mit Ereignis-Konsumenten kommunizieren. Dieses Paradigma verspricht eine flexible und effiziente Möglichkeit der Interaktion in großen rekonfigurierbaren verteilten Softwaresystemen.

Die Vorteile von ereignisbasierter Kommunikation sind:

- Das Ereignis-Paradigma erlaubt komplexe Interaktionen. Ereignisse können verwendet werden, um synchrone und asynchrone Kommunikation wie auch Multicast zu implementieren.
- Durch die lose Kopplung der Komponenten wird weitestgehende Unabhängigkeit erreicht und damit die Möglichkeit einfacher Umkonfiguration geschaffen.
- Durch Ereignisfilterung kann die Skalierbarkeit des Systems erhöht werden.
- Eine effiziente Implementierung einer Monitoring- und Logging-Komponente ist möglich.

Ereignisbasierte Kommunikation wird inzwischen in vielen Anwendungen verwendet. Es existieren verschiedene Ereignisdienste bis hin zu Ereignis-Infrastrukturen, welche für die verschiedensten Zwecke entwickelt wurden. Sie unterscheiden sich in ihrer Architektur und in ihren Leistungsmerkmalen.

Die für diese Arbeit wichtigsten Leistungsmerkmale eines Ereignis-Dienstes sind:

- Skalierbarkeit:

  Ausgehend von der Vision des Ubiquitous Computing der totalen Vernetzung, muss mit einer sehr hohen Zahl von Ereignis-Produzenten gerechnet werden.

- Unterstützte Ereignistypen:

  Da die unterschiedlichsten Typen von Ereignissen auftreten können, müssen entsprechende Mechanismen zur Spezifikation von Ereignissen vorhanden sein.

- Qualität der Ereignis-Filterung:

  Ausgehend von einer sehr hohen Zahl von Ereignissen und unterschiedlichsten Ereignistypen, müssen die Filterungsmöglichkeiten den anwendungsspezifischen Anforderungen genügen.

Weitere Bewertungskriterien für Ereignisdienste können die Qualität des Benachrichtigungsdienstes (Datenmenge und Frequenz), garantierte Auslieferung, Realzeit-Bedingungen und Sicherheit sein.

Im Folgenden wird ein kurzer Überblick über die wichtigsten Vertreter von Ereignissystemen gegeben und deren Einsatzmöglichkeit in dieser Arbeit bewertet.

7.3.1.1 Überblick

OMG CORBA Event and Notification Service

Die Common Object Request Broker Architecture (CORBA) wurde von der Object Management Group (OMG) spezifiziert und ist ein offener Standard zum Objektmanagement. Ein Object Request Broker (ORB), zentraler Teil der CORBA-Architektur, ermöglicht den einzelnen Komponenten eine Netzwerk-übergreifende Kommunikation. Diese Grundfunktionalität wurde in der CORBA Spezifikation 2.0 [Corba2.0] um einige allgemeine Dienste erweitert. Einer davon stellt der CORBA Event Service [CorbaES01] dar, welcher eine Kommunikation mittels Ereignissen ermöglicht.

Der Event Service spezifiziert zwei unterschiedliche Rollen für Komponenten: den Lieferanten (engl. supplier), welcher die Ereignisse generiert, und den Konsumenten (engl. consumer), welcher die Daten empfängt und verarbeitet. Eine Kommunikation kann sowohl über Push- wie auch über Pull-Anfragen von beiden Komponententypen initiiert werden.

Um anonyme Kommunikation zu ermöglichen, wurde die Event Channel Architecture eingeführt. Die Event Channel Architecture besteht aus dem Event Channel, den Verwaltungskomponenten und verschiedenen Proxies. Ein Event Channel kann somit als ein zwischen Ereignis-Lieferanten und Konsumenten geschaltetes Objekt, das sowohl als Lieferant, wie auch als Konsument agieren kann, betrachtet werden. Je nach Implementierung ermöglicht der Event Channel die Filterung von Ereignissen und bietet Quality of Service (QoS)-Fähigkeiten wie Ereignis-Prioritäten und Auslieferungsgarantien. Es existieren zwei unterschiedliche Ausprägungen von Event Channels: generische, welche ausschließlich generische Ereignisse unterstützen, und typisierte, welche beide Ereignis-Typen – generisch und typisiert – verarbeiten.

Der CORBA Event Service wird durch den CORBA Notification Service ergänzt. Er spezifiziert unter anderem die im Event Service fehlende Möglichkeit zur Filterung von Ereignissen und die QoS-Fähigkeiten. Um die Interoperabilität zum Event Channel zu garantieren, erbt der Notification Channel die Schnittstellen zum Ereignis-Lieferanten und -Konsumenten. Er ermöglicht mehrere Instanzen der Verwaltungskomponenten und bietet ein Typ-Repository.

Zur Filterung von Ereignissen stehen Filterobjekte zur Verfügen. Sie können dem Notification Channel, Proxy-Objekten oder Verwaltungsobjekten zugeordnet werden,

womit eine hierarchische Filterung realisierbar ist. Die Spezifikation erfolgt in einer Constraint Sprache, welche eine Erweiterung der Trader Constraint Language [Gro97] darstellt. Die Definition eigener Constraint-Sprachen ist möglich.

Beide Ereignis-Modelle von CORBA ermöglichen einen Einsatz in allgemeinen Anwendungsgebieten. Sie stellen eine Vielzahl von Schnittstellen und Fähigkeiten bereit. Doch unterstützen sie keine Föderation von Channels. Basierend auf dem Objektmodell von CORBA wird ein Ereignis als Nachricht eines Objekts an ein anderes Objekt über definierte Schnittstellen verstanden. Es steht kein Observation-Mechanismus und damit auch kein Mechanismus zur Erkennung von Ereignis-Mustern zur Verfügung. Dies wird dem Ereignis-Konsumenten überlassen. Das Einsatzgebiet des Event und Notification Channels wird durch die Ressourcenanforderungen für CORBA beschränkt.

Java Event Modell

Java ist eine von Sun entwickelte objektorientierte Programmiersprache. Die Java Architektur umfasst ein Delegation Event Model [Sun97] sowie ein Distributed Event Model [Sun98]. Das Delegation Event Model ist für die Kommunikation von kleinen, zentralisierten Anwendungen innerhalb einer einzelnen Java Virtual Maschine (JVM), daher für diese Arbeit nicht weiter von Interesse. Das Distributed Event Model ist für die Kommunikation von Objekten auf beliebigen Rechnern. Es basiert auf Remote Method Invocation (RMI) und ermöglicht den Aufruf von Methoden auf entfernten Rechnern. Das Java Distributed Event Model wurde in der Java Intelligent Network Infrastructure (Jini) [Jini01] übernommen.

Die Event Architektur besteht aus dem Event Generator, dem Remote Event Listener und einem optionalen dritten Objekt, dem Event Adapter. Der Event Listener registriert sein Interesse an einem bestimmten Event beim Generator. Die Gültigkeit einer Registrierung wird durch die Spezifikation eines *lease* beschränkt. Der Event Generator sendet ein Ereignis an den Listener durch einen synchronen Aufruf der *notify*-Methode. Das eigentliche Ereignis wird als Parameter übergeben.

Zwischen Event Generator und Listener kann ein Distributed Event Adapter mit entsprechender *notify*-Schnittstelle gestellt werden. Er ermöglicht eine Erweiterung der Funktionalität ohne die Schnittstelle verändern zu müssen. Filterfunktionen wie auch Verwaltungsfunktionalität können implementiert werden. Wie schon bei CORBA erlaubt eine dazwischen geschaltete dritte Instanz die Anonymisierung der Kommunikation, verringert aber auch die Auslieferungsgeschwindigkeit.

Das Java Distributed Event Model ist ein schlankes Event-Modell. Es spezifiziert keine Methode zur Registrierung von Events, jedoch ermöglicht es die Interoperabilität von unterschiedlichen Anbietern. Filterung und QoS können durch das Einführen von entsprechenden Adaptern implementiert werden.

JEDI

JEDI (Java Event-based Infrastructure) ist eine in Java implementierte objektorientierte Infrastruktur, welche die Entwicklung von ereignisbasierten Anwendungen unterstützt [CDF98]. Jedi wurde am Politecnico di Milano entwickelt

und verwendet, um das verteilte Workflow-Managementsystem OPSS (ORCHESTRA Process Support System) zu implementieren.

Die Architektur setzt sich aus Active Objekts (AO) und Event Dispatcher (ED) zusammen. AOs können sowohl als Ereignis-Produzenten wie auch als Ereignis-Konsumenten fungieren, welche ihr Interesse an Ereignissen über subscribe- und unsubscribe-Mechanismen beim ED bekunden können. Der ED ist auch für die geordnete Auslieferung der Ereignisse verantwortlich.

JEDI bietet zwei Implementierungen des ED an: eine zentralistische und eine verteilte. Die verteilte Version des ED besteht aus einer Menge von Dispatching Servern, welche in einer Hierarchie organisiert sind. Damit ist das Problem eines Flaschenhalses, das bei einer zentralistischen Komponente entstehen kann, behoben. Eine Filterung der Ereignisse ist über die Spezifikation von Ereignis-Mustern möglich, die, wie auch die Ereignisse selbst, aus einer Menge von Zeichenketten bestehen.

Eine Besonderheit von JEDI stellt die Mobilitätsunterstützung dar. Reactive Objects, welche eine Unterklasse von Active Objects mit einem bestimmten Verhalten bilden, bieten eine Methode *move* an. Wird diese Methode aktiviert, so erfolgt eine Trennung vom ED, der Java Byte Code wird serialisiert, auf einen anderen Host übertragen, aktiviert und wieder an den ED gekoppelt.

JEDI stellt eine einfach zu verwendende Infrastruktur mit Active Objects und Event Dispatcher zur ereignisbasierten Kommunikation zur Verfügung. Der verteilte ED, welcher in einer Baumstruktur organisiert ist, scheint auf den ersten Blick das Problem eines Flaschenhalses einer zentralistischen Komponente zu beseitigen, doch durch das Weiterleiten jedes Ereignisses an den Eltern-Knoten bis zur Wurzel des Baums wird diese zum Flaschenhals.

Ein weiterer Vertreter eines Ereignisdienstes ist SIENA. Da die vorliegende Arbeit auf SIENA aufbaut, wird SIENA im folgenden Abschnitt genauer vorgestellt.

7.3.1.2 SIENA

SIENA (Scalable Internet Event Notification Architecture) wurde, wie auch JEDI, am Politecnico di Milano entwickelt [Car98, Siena]. Das Designziel von SIENA war es, eine Unterstützung sowohl für ereignisbasierte Kommunikation in Weitverkehrsnetzen wie auch von mobilem Code zu realisieren.

Ereignismodell

Die SIENA Infrastruktur realisiert ein allgemeines, skalierendes Ereignis-Modell durch verteilte Ereignis-Server. Ereignisse werden durch so genannte *Objects of Interest* erzeugt und durch *Interested Parties* konsumiert. Die Verteilung der Ereignisse wird durch die Mechanismen *advertisement*, *subscription*, *publication* und *notification* geregelt (siehe Abbildung 36).

Während die *subscription*- und *publication*-Mechanismen zur Registrierung des Interesses an bestimmten Ereignissen bzw. Publizierung in den meisten Event-Modellen unterstützt werden, bietet SIENA einen weiteren, den *advertisement*-Mechanismus an. Ein Ereignis-Produzent kann über die *advertise*-Operation sein

Interesse zur Generierung von einem Ereignis eines bestimmten Typs bekunden und über die Umkehroperation *unadvertise* wieder zurücknehmen.

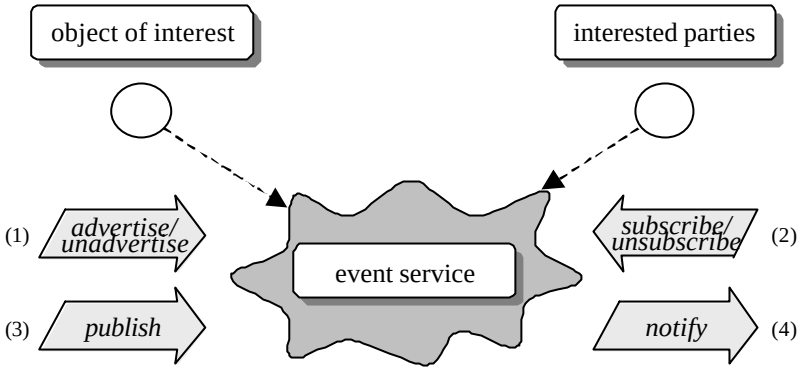

Abbildung 36 SIENA Ereignisdienst

Topologie

Die Auslieferung der Ereignisse wird durch einen oder mehrere Event Server realisiert. Diese können in unterschiedlicher Weise miteinander vernetzt werden: zentralisiert, hierarchisch, azyklisch peer-to-peer und peer-to-peer (siehe Abbildung 37). Hybride Topologien aus hierarchischer und peer-to-peer-Anordnungen sind ebenfalls möglich. Die Peer-to-Peer-Topologie erfordert – im Gegensatz zur hierarchischen Server-Anordnung - ein eigenes Protokoll für die Server, welches eine bidirektionale Kommunikation ermöglicht.

Carzaniga untersuchte in seiner Arbeit [Car98] diese vier unterschiedlichen Topologien zur Anordnung der Event Server auf Flexibilität und Skalierbarkeit. Es hat sich gezeigt, dass die verteilten Lösungen bei zunehmender Objektzahl gegenüber der zentralistischen Anordnung die besseren Alternativen darstellen.

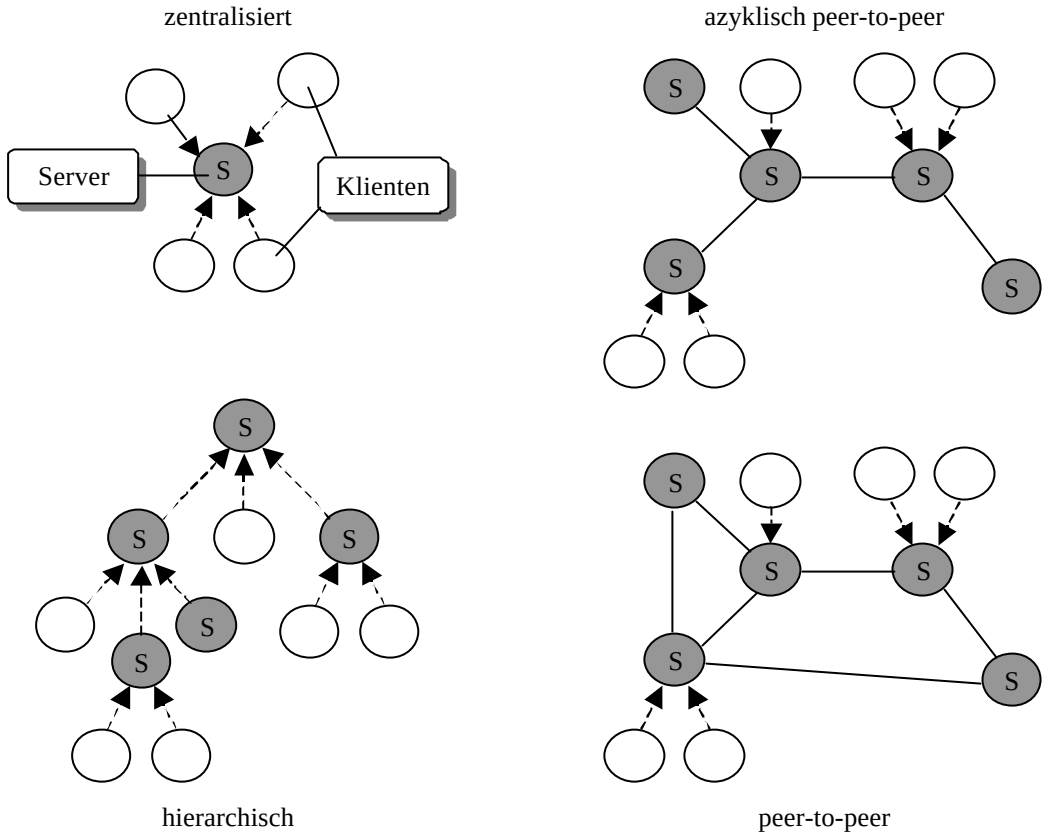

Abbildung 37 Server-Topologien in SIENA

Naming

Jedes Objekt benötigt einen eindeutigen Identifikator und einen Handler, um eine Kommunikation zu ermöglichen. Für beides wird in SIENA das Konzept einer URI eingesetzt. Es werden die meisten Schemata unterstützt sowie die Protokolle *mailto* und *http*. Das System legt nur die URIs der Server fest. Die Klienten bestimmen ihre URI selbst und teilen diese dem System bei der Registrierung mit. Eine Überprüfung auf Korrektheit findet nicht statt.

Ereignisse, Filter und Pattern

Ereignisse werden als Tripel Name, Typ und Wert dargestellt. Attribute von Ereignissen sind eindeutig durch ihre Namen identifiziert. Bei der Festlegung des Typsystems für Ereignisse wurde ein Mittelweg gewählt. Ein ungetyptes System hat den Vorteil, dass es einfach zu implementieren ist. Es ist universell einsetzbar und daher für verteilte, heterogene Systeme geeignet. Die Realisierung von Filtern ist jedoch schwieriger, da zum Vergleich das Ereignis in den entsprechenden Typ transformiert werden muss. Ein benutzerdefiniertes Typsystem, wie in einer objektorientierten Sprache, erleichtert dagegen die Filterung, verursacht jedoch Performanzprobleme. Es ist auch nicht davon auszugehen, dass alle zu unterstützenden Endsysteme objektorientierte Sprachen verstehen. Aus diesem Grunde werden ausschließlich einfache Typen wie *string*, *time*, *int* und *float* unterstützt.

Ein *Ereignis-Filter* ist eine Menge von Attribut-Filtern. Ein Attribut-Filter spezifiziert den Attributnamen, den Typ, einen booleschen Operator und einen Attributwert. Weitere Einschränkungen für den Wertebereich sind möglich. Eine feste Menge von Operationen ist vorgegeben. Ein *Ereignis-Muster* wird durch die Kombination von Filtern über spezielle algebraische Operatoren realisiert.

Zeit

Ein Problem bei der Auslieferung von Ereignissen ist die Zeit. Das Problem, Ereignisse in die richtige temporale Reihenfolge zu bringen, erfordert die Zwischenpufferung, Sortierung und vor allem einen Zeitstempel für alle Ereignisse.

In SIENA werden verschiedene Annahmen über die Zeit getroffen:

- es existiert eine globale Uhr, welche für die Zeitstempel der Ereignisnachrichten verwendet wird und
- es existiert eine untere Schranke für die Verzögerungen im Netz.

Während die erste Annahme durch GPS-Dienste, Synchronisationsprotokolle und INTP (Internet Network Time Protocol) zufrieden stellend erfüllt werden kann, stellt die zweite Annahme ein Problem dar. In einer Peer-to-Peer-Topologie, weniger in einer zentralistischen, können lange Wege entstehen und Wartezeiten in den einzelnen Knoten können sich addieren. Dieses Problem wird als Aufgabe der darunter liegenden Kommunikationsschicht betrachtet und nicht weiter behandelt.

Mobilität

Mobilität wird in SIENA dadurch unterstützt, dass Objekte über eine eindeutige URI lokalisiert werden. Damit ist der physische Aufenthaltsort verborgen. Die in JEDI angebotene *move*-Operation wird nicht unterstützt, deren Funktionalität jedoch über eine Erweiterung des Schichtenmodells ermöglicht.

Bewertung

SIENA realisiert einen universalen Ansatz zur ereignisbasierten Kommunikation in Weitverkehrsnetzen. Durch die Verteilung des Ereignisdienstes auf eine Menge von Event Servern, wird eine gute Skalierung erreicht. Die zusätzlich angebotene *advertise*-Funktionalität optimiert *subscription* und *publication*, doch sie erfordert sie auch einen erhöhten Managementaufwand, da eine zusätzliche Komponente zur Verwaltung der *advertisements* benötigt wird.

Wie in den anderen beschriebenen Systemen auch, ist SIENA nicht konzipiert, Ereignisse längerfristig zu speichern, um diese für weitere Auswertungen und Anfragen zu einem späteren Zeitpunkt verwenden zu können. Dies bedeutet, dass ein eigenes Datenmanagement zur Archivierung von Ereignissen benötigt wird.

7.3.2 Datenmanagement

Ereignisbasierte Infrastrukturen vermitteln im Allgemeinen nur aktuelle Ereignisse. Ereignis-Konsumenten können nur auf die Ereignisse zugreifen, die nach der Etablierung einer Verbindung zum Ereignis-Produzenten oder Registrierung beim

Benachrichtigungsdienst aufgetreten sind. Um jedoch auf vergangene Ereignisse zurückgreifen zu können, wie es beispielsweise für die Erkennung von Ereignis-Mustern notwendig ist oder zur Verfolgung von Entwicklungen eines bestimmten Kontexttyps (z.B. Temperatur-Entwicklung über einen bestimmten Zeitraum), müssen die Ereignisse aufgezeichnet und geeignet gespeichert werden. Diese Speicherung kann in unterschiedlicher Form erfolgen. Eine Möglichkeit wäre, dass jeder Ereignis-Produzent seine Ereignis-Historie archiviert. Dies ist aber in einem Umfeld, in dem die Ereignis-Produzenten über sehr geringe Ressourcen verfügen nicht realisierbar. Auch wäre die Erkennung von Ereignis-Mustern verteilter Produzenten aufwendig.

Eine Speicherung im Ereignis-Konsumenten wäre eine weitere Alternative. Jeder Konsument speichert die Ereignisse, welche für ihn von Interesse sind. Das wirft mehrere Probleme auf: der Konsument kann, wie der Produzent, evtl. durch Ressourcenknappheit diese Aufgabe nicht erfüllen. Mobile Konsumenten haben das Problem der Erreichbarkeit und bevorzugen oftmals als Kommunikationsform die Pull-Methode vor der Push-Methode. Verwenden mehrere Konsumenten die gleichen Ereignisdaten, so entsteht ein enormer Overhead inklusive redundanter Datenhaltung.

Die dritte Alternative ist eine gesonderte Datenhaltungskomponente, welche die Ereignisse sammelt, filtert, speichert und den interessierten Konsumenten über den Benachrichtigungsdienst oder aber über eine Anfrage-Schnittstelle anbietet. Diese Datenhaltungkomponente kann sowohl als externer Ereignis-Konsument realisiert werden oder aber als integraler Bestandteil des Ereignissystems um unnötigen Kommunikationsoverhead zu vermeiden. Die letztgenannte Alternative wurde für diese Arbeit gewählt.

Für die Einbindung der Datenhaltungskomponente in die Gesamtarchitektur des Ereignissystems bieten sich zwei Alternativen an: der schichtenorientierte Ansatz und der Build-In-Ansatz. In einem schichtenorientierten Ansatz kann eine konventionelle Datenbank verwendet werden, auf der eine Abfrage- (beispielsweise SQL) und Managementschicht aufsetzt. Der Vorteil dieses Vorgehens ist, dass auf die von einer konventionellen Datenbank angebotene Funktionalität (wie beispielsweise Transaktionsmanagement, Fehlertoleranz oder Indexing) zurückgegriffen werden kann ohne Modifikationen an der Datenbank vornehmen zu müssen. Nachteil ist, dass die Datenbank nicht speziell auf die Bedürfnisse eines Ereignissystems, wie beispielsweise spezielle temporale Abfragesprachen oder hohe Schreibraten durch hohes Ereignis-Aufkommen, abgestimmt und durch die Verallgemeinerung für beliebige Einsatzszenarien in der Performanz nicht für dieses Szenario optimiert ist.

In einer Build-In-Architektur wird die Abfrage- und Managementfunktionalität in das Datenbanksystem integriert. Dadurch kann die Datenbank an die speziellen Anwendungsbedürfnisse angepasst werden. Durch die Anpassung der Datenbank beispielsweise an die speziellen Ereignis-Daten, temporalen Aspekte von Ereignissen oder Ereignis-Muster, kann eine hohe Performanz erreicht werden. Nachteil ist, dass die Datenbank nicht speziell auf die Bedürfnisse eines Ereignissystems abgestimmt ist, wie beispielsweise spezielle temporale Abfragesprachen oder hohe Schreibraten durch hohes Ereignis-Aufkommen, und nicht durch die Verallgemeinerung für beliebige Einsatzszenarien in der Performanz optimiert ist.

Da die Entwicklung einer neuen Datenbank bzw. eines neuen Datenbanktyps nicht Ziel dieser Arbeit war, wurde der schichtenorientierte Ansatz verwendet. Dabei

wurde bei der Auswahl des DB-Produkts nicht nur Augenmerk auf die Optimierung bezüglich des Ereignissystems geachtet, sondern vor allem auch auf die Eignung bezüglich der Spezifikationssprache für Kontextinformationen und vorhandene Abfragesprachen.

Wird RDF/XML zur Spezifikation von Kontextdaten verwendet (siehe Abschnitt 5.4.1), so ist es sinnvoll, eine spezielle RDF-Datenbank zu verwenden. Diese Technik steht am Anfang ihrer Entwicklung, wodurch nur wenige Prototypen zur Verfügung stehen. Eine der ersten RDF-Datenbanken, und wohl die bekannteste, ist die rdfDB von Guha [rdfDB01]. RdfDB ist ein RDF-Datenbankserver auf Basis der Berkeley Sleepycat Datenbank und stellt Routinen zum Im- bzw. Export von RDF-Dateien bereit. Das Ziel der Entwickler von rdfDB war es, eine einfache, skalierbare Open Source-DB zu entwickeln. Durch die Verwendung von C als Programmiersprache und TCP/IP-Sockets ist eine Integration in annähernd jede Umgebung möglich. Obwohl diese Datenbank als innovativer Ansatz betrachtet werden muss, wurde die Entwicklung nicht konsequent weiter betrieben. Die aktuelle Version ist nicht vollständig ausgereift und von schlechter Performanz. Doch diente dieser erste Ansatz als Vorbild für andere Prototypen.

Ein weiterer Prototyp einer RDF-Datenbank wurde im EU-Projekt On-To-Knowledge entwickelt. Sesame [Sesame] basiert auf der Open Source-Datenbank PostgresSQL und unterstützt neben RDF auch RDF Schema. Während eine einfache Online-Demo zum Testen von Sesame früh verfügbar war, wurde die erste Version 0.2 erst März 2002 freigegeben, zu spät, um in dieser Arbeit eingesetzt zu werden. Da dieser Prototyp vielversprechend erscheint, sollte die weitere Entwicklung beobachtet werden.

Eine ebenfalls auf einer objekt-relationalen Datenbank aufbauender Prototyp RSSDB (RDF Schema Specific DataBase) stammt von Sofia Alexaki von ICS-FORTH, Griechenland [RDFS]. ICS-FORTH verwendet bei der Entwicklung ebenfalls PostgresSQL als ORDBMS, doch kann, da RSSDB mit Java und JDBC implementiert wurde, laut Aussage der Entwickler, „jedes" beliebige ORDBMS verwendet werden. Die Plattformabhängigkeit, die beispielsweise bei RDFDB (Linux, BSD, Solaris) gegeben ist, wurde damit aufgehoben. RSSDB ist ein Open Source-Projekt und kann unter [RDFS] heruntergeladen werden.

7.3.2.1 Benachrichtigungs- und Abfragedienst

Das realisierte Kontextsystem verfügt über zwei Schnittstellen zu kontextabhängigen Anwendungen: den Benachrichtigungsdienst und die Abfrage von Kontext-information.

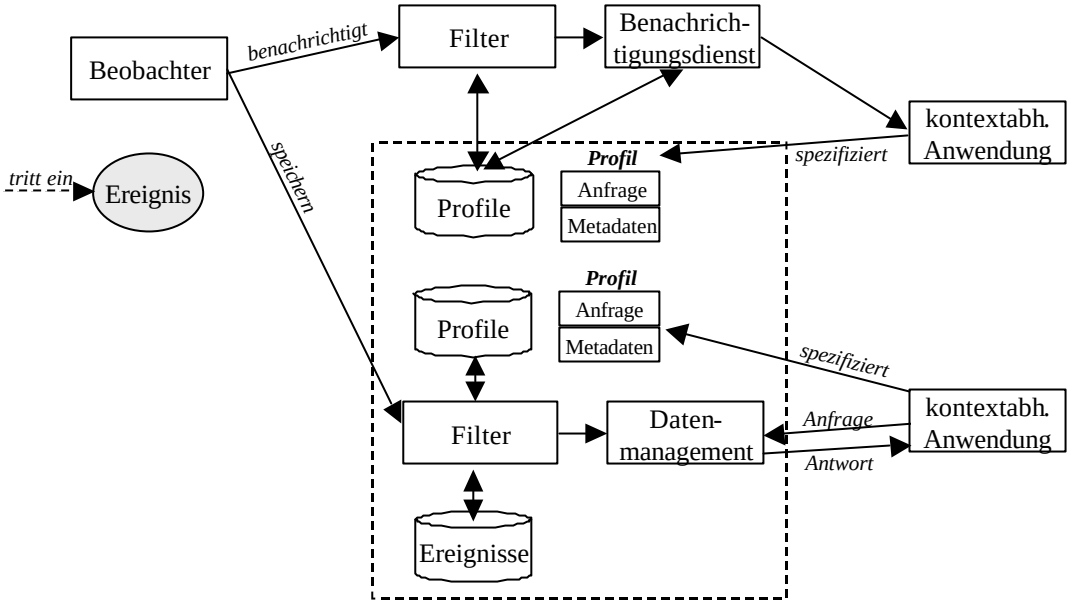

Abbildung 38 Benachrichtigungs- und Abfragedienst

Damit eine Anwendung über das Eintreffen eines Ereignisses informiert werden kann, muss diese ihre Wünsche in einem Profil spezifizieren und das Ereignis abonnieren. Das Profil besteht aus zwei Teilen, der eigentlichen Anfrage und Meta-Daten, welche festlegen, wer das Ereignis abonniert hat und wie die Benachrichtigungsbedingungen sind.

Tritt ein Ereignis auf, so informiert der Beobachter den Filter und das Datenmanagement, das das Ereignis archiviert. Der Filter vergleicht das ankommende Ereignis mit den gespeicherten Profilen und informiert den Benachrichtigungsdienst, wenn das Ereignis mit einer Anfrage übereinstimmt. Der Benachrichtigungsdienst überprüft die Bedingungen der Anfrage und schickt an die entsprechende Anwendung eine Nachricht.

Sowohl die aus Sicht der kontextabhängigen Anwendung passive Anfrage über einen Benachrichtigungsdienst wie auch die aktive Anfrage erfordern eine Sprache für die Spezifikation der gewünschten Ereignisdaten. Diese können aus einfachen oder aber aus einer Kombination von mehreren Ereignissen, so genannten Ereignis-Muster (siehe Abschnitt 5.1.3) bestehen. Eine Anfragesprache muss somit das Format der unterstützten Ereignisse berücksichtigen und die Möglichkeiten der Kombination.

Da zur Realisierung des Datenspeichers RSSDB ausgewählt wurde, steht eine Implementierung von RQL als Abfragesprache zur Verfügung.

Kapitel 8

Zugriffskontrollinfrastruktur

Neben einer geeigneten Politikbeschreibung sind die Politikverwaltung und die Politikdurchsetzung von zentraler Bedeutung für die Sicherheit des Systems. Eine geeignete Infrastruktur muss zum einen die Heterogenität der einzelnen Endsysteme unterstützen, das Skalierungsproblem berücksichtigen sowie ein verteiltes Management realisieren. An alle Komponenten der Zugriffskontrollinfrastruktur sind besondere Anforderungen bezüglich der Fehleranfälligkeit und Ausfallsicherheit zu stellen.

Dieses Kapitel beschreibt die Architektur und die Implementierung der realisierten kontextabhängigen Zugriffskontrolle. In Abschnitt 8.1 werden verschiedene Vorschläge für eine Implementierung einer verteilten Zugriffskontrolle betrachtet und bewertet bezüglich deren Einsatzmöglichkeiten im realisierten System. Abschnitt 8.2 gibt dann einen Überblick über die Gesamtarchitektur des realisierten Systems. Die beiden Hauptkomponenten Politikmanagement und Zugriffskontrollmechanismen des Zugriffskontrollsystems werden in den Abschnitten 8.3 bzw. 8.4 detaillierter vorgestellt.

8.1 Architekturen für verteilte Zugriffskontrolle

Ein Zugriffskontrollsystem überwacht und kontrolliert den Zugriff auf Objekte. Dieser Service ist ein wesentlicher Teil der Sicherheitsinfrastruktur, welcher die Vertraulichkeit, die Integrität von Daten und die Zurechenbarkeit eines Zugriffs garantiert. Die Regeln, nach denen das Zugriffskontrollsystem arbeitet, werden in einer Politik festgelegt, die Mechanismen, die zur Durchsetzung benötigt werden, sind Teil des IT-Systems.

Um eine ausreichende Flexibilität bei der Festlegung der Rechte und der Durchsetzungsart in einem verteilten System zu erreichen, werden beide Teile – Politik und Durchsetzungsmechanismen – voneinander getrennt. Dadurch wird es möglich, für unterschiedliche Systemressourcen unterschiedliche Politiken zu definieren, unterschiedliche Durchsetzungskomponenten zu verwenden oder auch mehrere Politiken für eine Ressource zu spezifizieren.

Für die Durchsetzung einer verteilten Zugriffskontrolle stehen verschiedene Ansätze und Systeme zur Verfügung. Diese können in zwei große Gruppen eingeteilt werden:

- Mechanismen, welche unabhängig vom restlichen System eingesetzt werden können, und
- die Middleware-Gruppe.

Zu den unabhängigen Mechanismen zählen beispielsweise Kerberos oder SESAME (Secure European System for Applications in a Multi-Vendor Environment). Kerberos ist ein Authentifikations- und Schlüsselverteilsystem, das im Rahmen des Athena-Projekts am Massachusetts Institute of Technology (MIT) entwickelt wurde [MNS+87]. SESAME ist eine europäische Implementierung der von der European Computer Manufacturer Association (ECMA) stammenden OSI-Sicherheitsarchitektur, die auf Kerberos als Authentisierungsdienst setzt [Sesame].

Middleware-Technologien, welche einen verteilten Autorisierungsdienst anbieten, sind beispielsweise CORBA der Object Management Group (OMG) mit dem CORBA Security Service [OMG98], DCE der Open Software Foundation (OSF) [GH95] oder DCOM der Microsoft Corporation[DCOM96].

Die OMG entwickelte die Resource Access Decision (RAD)-Architektur [OMG99, RAD99, EBK00], welche die Trennung der Zugriffskontrolle von der Anwendung propagiert. RAD konzipiert eine Architektur, welche die Autorisierungslogik in einen Autorisierungsservice kapselt. Dieser ist sowohl vom Sicherheitsmodell wie auch von der Politik und der darunter liegenden Sicherheitsinfrastruktur unabhängig. RAD wurde speziell für verteilte Anwendungen in Unternehmen entwickelt.

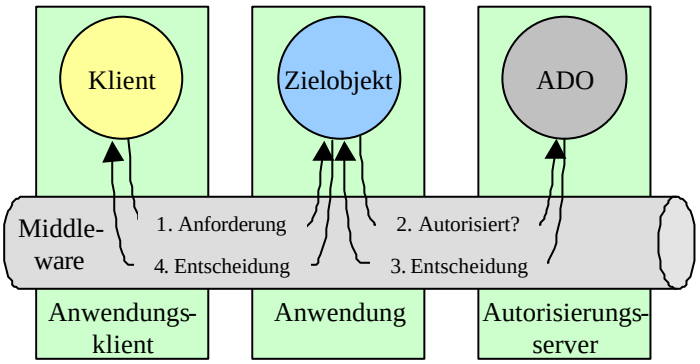

Abbildung 39 Autorisierungsinteraktion in RAD

Fordert ein Klient in RAD einen Zugriff auf ein Objekt, so überprüft die Anwendung dessen Berechtigung durch eine Anfrage bei der zentralistischen Entscheidungskomponente (engl. Access Decision Object; ADO) nach (siehe Abbildung 39). Die Entscheidung wird an die Anwendung zurückgeliefert, die diese durchsetzt [Bez00].

Ein Standardisierungsansatz für Netzmanagement-Politiken und deren Management wird derzeit bei der Internet Engineering Task Force (IETF) in Zusammenarbeit mit der Distributed Management Task Force (DMTF) diskutiert [MES+01]. Die DMTF arbeitet an der Spezifikation zur Repräsentation von Politiken, dem korrespondierenden Informationsmodell und einem Framework für politikbasiertes Netzmanagement, während die IETF sich auf die Spezifikation der einzelnen Komponenten und der Protokolle für die Kommunikation zwischen den Komponenten konzentriert. Ziel des Gesamtvorschlags ist es, Mechanismen festzulegen, welche eine Vereinfachung und Automation der QoS-Konfiguration für IP-Netzwerke ermöglichen.

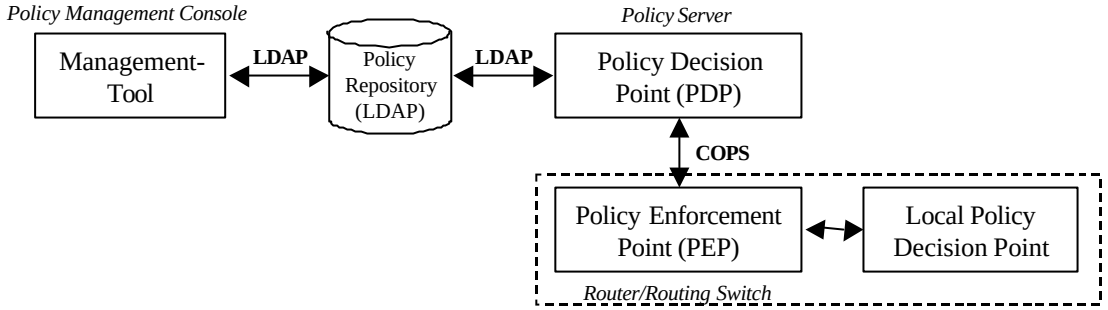

Abbildung 40 Architektur des IETF Policy Framework

Das IETF Policy Framework [SS98, YP00, HMS+00] besteht aus vier Komponenten:

- dem Management-Tool zur Spezifikation der Politik
- einen LDAP-Server zur Speicherung der Politiken,
- den Entscheidungskomponenten (engl. Policy Decision Points; PDPs) und
- den Durchsetzungskomponenten (engl. Policy Enforcement Points; PEPs).

Zur Kommunikation zwischen den Komponenten und dem LDAP-Server wird das LDAP-Protokoll verwendet, für die Kommunikation zwischen den Klienten und dem zentralen Autorisierungsserver das eigens spezifizierte Common Open Policy Service (COPS)-Protokoll. Eine Entscheidung über ein Zugriffs-Ereignis wird durch den zentralen PDP getroffen und an den anfragenden PEP zurückgeliefert. Ist der zentrale PDP nicht erreichbar, so sieht die Architektur lokale PDPs vor. Diese werden in der Spezifikation nicht weiter festgelegt, genauso wenig das Kommunikationsprotokoll zwischen den PEPs und den lokalen PDPs.

Die IETF-Architektur ähnelt der RAD-Architektur sehr, stellt jedoch im Gegensatz zu RAD und den genannten Implementierungen konkrete Vorschläge zum Management von Politiken und der Realisierung vor. Da die Zielgruppe des IETF-Vorschlags Netzmanagementpolitiken sind, ist dieser Vorschlag nicht vollständig auf Autorisierungspolitiken anwendbar, da diese eine Untermenge der Managementpolitiken darstellen und von geringerer Komplexität sind.

8.2 Gesamtarchitektur

Die in dieser Arbeit entwickelte Infrastruktur für eine kontextabhängige Zugriffskontrolle (siehe Abbildung 41) ähnelt im Grobaufbau dem Vorschlag der IETF für eine verteilte Zugriffskontrolle. Alle Spezifikationen bzw. Teile der Spezifikationen, welche spezifisch für das eigentliche IETF-Zielszenario Netzmanagement sind, wurden nicht weiter beachtet. Dazu gehört beispielsweise die vorgeschlagene Spezifikationssprache für Politiken, die so genannte Network Policy Language, da diese für kontextabhängige Zugriffspolitiken nicht geeignet ist. Das Kommunikationsprotokoll COPS wird zur Kommunikation zwischen den PEPs und PDPs verwendet. Alternative Protokolle wären beispielsweise SOAP oder XML-RPC, da beide einen Fernaufruf ermöglichen und ebenfalls XML als Nachrichtenformat benutzen. Der Nachteile dieser Protokolle ist jedoch, dass beide nicht auf einem einfachen Transportprotokoll basieren, sondern auf http oder SMTP, während COPS auf TCP aufsetzt.

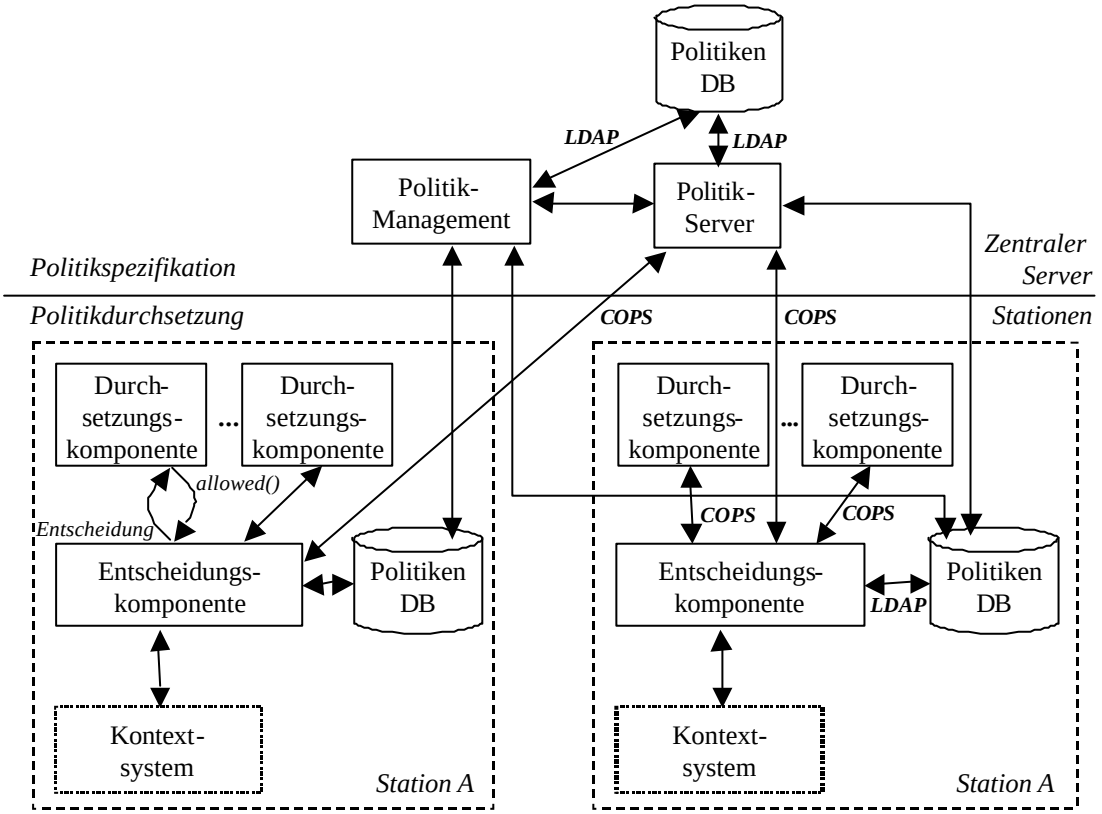

Abbildung 41 Aufbau des kontextabhängigen Zugriffskontrollsystems

Das realisierte kontextabhängige Zugriffskontrollsystem verfügt über eine zentrale *Management-Komponente* auf einem Management-Server, welcher für die Definition und Speicherung aller Sicherheitspolitiken zuständig ist (siehe Abbildung 41). Jede Station mit geringen zur Verfügung stehenden Ressourcen besitzt eine eigene *Entscheidungskomponente*, ein *Politik-Repository* mit den lokal benötigten Daten und je nach gewählter Implementierung, eine oder mehrere *Durchsetzungskomponenten*. Die Durchsetzungskomponenten können unterschiedlich sein, je nach Art

des Zielobjekts und der Kommunikationsinfrastruktur. Die Entscheidungskomponente ist vom restlichen IT-System unabhängig.

Das Kontextsystem ist nicht Teil des Zugriffskontrollsystems, da es als eigenständige Infrastruktur realisiert wurde, um auch anderen kontextabhängigen Anwendungen außer der kontextabhängigen Zugriffskontrolle Kontextinformationen zur Verfügung stellen zu können. Das Kontextsystem liefert die benötigten Kontextdaten über eine sichere Verbindung der Entscheidungskomponente des Zugriffskontrollsystems. Die Entscheidungskomponente ist somit eine kontextabhängige Anwendung, die über die in den Abschnitten 7.2.7 und 7.3.2 beschriebenen Schnittstellen mit dem Kontextsystem kommuniziert.

Die einzelnen Komponenten dieser Zugriffskontrollinfrastruktur werden in den folgenden Abschnitten ausführlicher beschrieben.

8.3 Politikmanagement

Die Politikmanagement-Komponente steuert und überwacht den Lebenszyklus der Zugriffspolitiken. Dieser besteht aus den Zuständen „schlafend“, „deaktiviert“, „aktiviert“ und „gelöscht“ (siehe Abbildung 42). Das Politikmanagement umfasst somit die drei Grundaufgaben „Politikerstellung“, „Verteilung“ an die verteilten Stationen und „Aktivierung/Deaktivierung“.

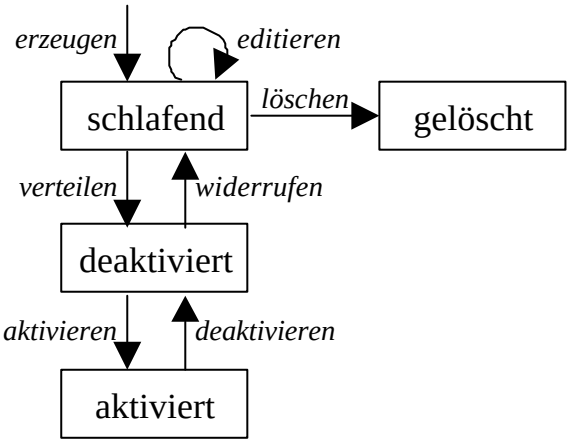

Abbildung 42 Lebenszyklus einer Politik

Das Politikmanagement wird durch eine zentrale Management-Komponente in Zusammenarbeit mit dezentralen Komponenten auf den einzelnen Stationen realisiert. Jede Aktion im Lebenszyklus wird von der zentralen Managementkomponente aus gesteuert (Abbildung 43).

Das Editieren einer kontextabhängigen Zugriffspolitik durch die zentrale Managementkomponente (zMS) wird in der aktuellen Implementierung nicht unterstützt. Hier kommt der Vorteil von XML gegenüber anderen Politiksprachen zum Tragen. Für XML sind die verschiedensten Tools zum Editieren und Validieren erhältlich. Die Entwicklung eines Editors, eines Parsers für die Sprache sowie eines Precompilers,

welcher Politikpezifikationen in einem Logikformat in ein handhabbares Zwischenformat umwandelt, entfällt.

Wurde eine valide Politik erstellt, wird eine Konsistenzprüfung durchgeführt. Dabei werden die in der Politik spezifizierten Subjekte, Objekte und Rollen mit den gespeicherten abgeglichen und evtl. vorhandene Inkonsistenzen aufgedeckt. Werden neue Einträge identifiziert, so erfolgt eine Rückfrage beim Ersteller. So wird verhindert, dass Schreibfehler unbeabsichtigt neue Einträge in der Datenbank erzeugen oder Kollisionen von neuen Objekten mit vorhandenen Kennungen auftreten. Entsprechendes gilt für die verwendeten Kontextinformationen, welche darauf überprüft werden, ob sie mit den im Kontextsystem angebotenen Informationen übereinstimmen.

Jede Zugriffspolitik muss an die Stationen, in denen die im Scope spezifizierten Objekte liegen, verteilt und aktiviert werden. Es werden die Objekte kontrolliert und die betroffen Stationen ermittelt. Die Politikregeln werden nach Zielstation aufgeteilt und mit den restlichen Informationen an die entsprechende Station (lokalen Management Server, lMS) übermittelt und dort lokal gespeichert. Über ein Aktivierungssignal von der Managementkomponente an das lokale Management wird die neue Politik eingelesen, aufgearbeitet und in den Entscheidungsprozess mit einbezogen.

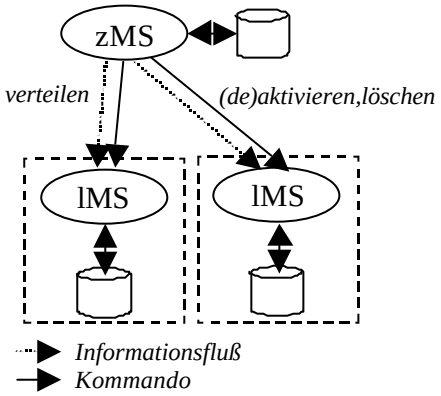

Abbildung 43 Komponenten des Politikmanagements

Datenhaltungskomponente

Die Datenhaltungskomponente ist für die konsistente und persistente Speicherung der Politiken, der Benutzer-, Objekt- und teilweise Kontextdaten verantwortlich. Von der Datenhaltungskomponente verwaltete Kontextinformationen können beispielsweise Systemzustände sein, wie lokale Zeit, oder Gruppen- und Rolleninformationen. Der überwiegende Teil der Kontextdaten wird jedoch erst zur Laufzeit über Abfragen an das Kontextsystem ermittelt.

Die Speicherung der Zugriffskontrolldaten kann prinzipiell in einer Datenbank oder einem LDAP-Server erfolgen. Bei Performanzabwägungen und Berücksichtigung von Standards, ist ein LDAP-Server vorzuziehen. Jedoch entscheiden letztlich die auf dem Endsystem zur Verfügung stehenden Ressourcen und angebotenen LDAP- bzw. DB-Produkte. Da die Architektur vorsieht, dass der Zugriff auf den Endsystemen

durch die Entscheidungskomponente auf das Politik-Repository über das LDAP-Protokoll erfolgt, muss dieses das LDAP-Protokoll unterstützen.

Da es sich bei der RaumComputer-Architektur, für die die Zugriffskontrolle konzipiert wird, um ein verteiltes System mit geringen Ressourcen in den einzelnen Endsystemen handelt, ist die Verteilung der Zugriffskontrolldaten von zentraler Bedeutung (siehe Abbildung 44). Jede Station erhält nur die für sie relevanten Daten über Politiken und die Objekte, welche lokal in der Station gespeichert werden. Die Verteilung erfolgt anhand der Informationen über die zu überwachenden Objekte und einer gespeicherten Liste der Stationen.

Daten über Subjekte werden zentral verwaltet. Die einzelnen Stationen stellen bei Bedarf eine Anfrage an die zentrale Datenmanagement-Komponente. Die erhaltenen Subjekt-Informationen werden über einen festgelegten Zeitraum lokal zwischengespeichert, da davon ausgegangen wird, dass der gleiche Benutzer mehrere Zugriffsanforderungen in Serie stellt. Um Inkonsistenzen bei einer replizierten Datenhaltung vorzubeugen, sollte die Caching-Zeit kurz gehalten werden.

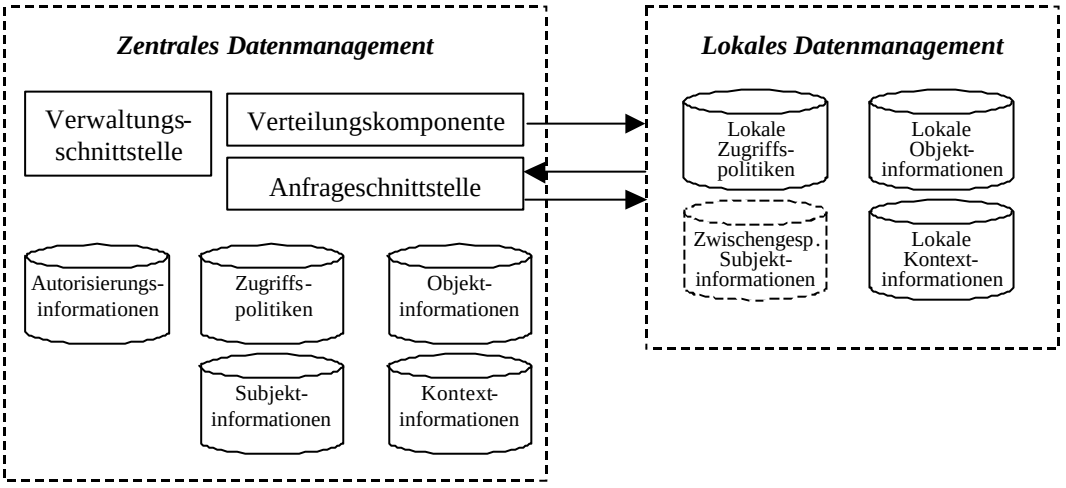

Abbildung 44 Verteiltes Datenmanagement

8.4 Zugriffskontrollmechanismen

Die beteiligten aktiven Komponenten einer Zugriffskontrolle sind nach [LaP90] die politik-unabhängige Durchsetzungskomponente (engl. Access Control Decision Facility) und die politik-abhängige Entscheidungskomponente (engl. Access Control Enforcement Facility). Passive Komponenten sind die Zugriffskontrolldaten der Benutzer und Objekte, die Politiken, die Kontextdaten und die Autoritäten, welche den Zugriff auf die Komponenten der Zugriffskontrolle festlegen [AEL+90]. Diese werden durch eine lokale Datenhaltungskomponente bzw. das lokale Kontextsystem verwaltet.

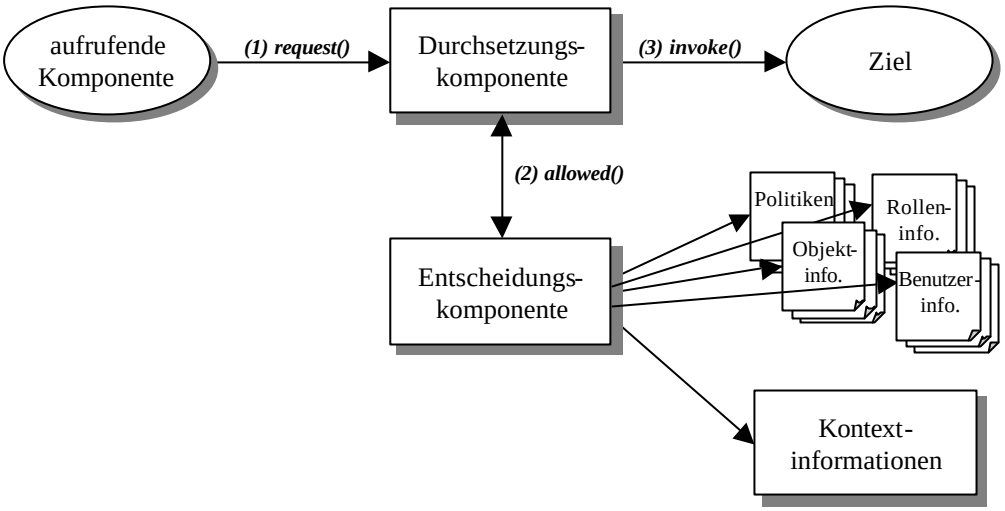

Abbildung 45 Zugriffskontrolle

8.4.1 Durchsetzungskomponente

Eine Durchsetzungskomponente für die Zugriffskontrolle muss in der Lage sein, jeden Zugriffsversuch zu kontrollieren. Es darf keine Möglichkeit bestehen, diese zu umgehen. Diese Eigenschaft wird auch „complete mediation“ [DoD85] genannt und die Durchsetzungskomponente als „Reference Monitor“ bezeichnet.

Erfolgt eine Zugriffsanforderung, so wird durch die dem Zielobjekt vorgeschaltete Durchsetzungskomponente eine entsprechende Anfrage an die Entscheidungskomponente gestellt. Diese überprüft anhand der erhaltenen Anfragedaten, der gespeicherten Zugriffspolitiken sowie der Benutzer-, Objekt- und Gruppeninformationen die Zulässigkeit und liefert eine Entscheidung zurück.

Die Durchsetzungskomponente oder der Policy Enforcement Point (PEP) ist dafür zuständig, dass die Anforderung einer aufrufenden Komponente an ein Zielobjekt unterbrochen und erst nach einer positiven Entscheidung durch die Entscheidungskomponente fortgesetzt bzw. bei einer negativen Entscheidung mit einer Fehlermeldung abgebrochen wird. Die Hauptaufgaben des PEP sind:

- Unterbrechung des Zugriffs,
- Weiterleitung der Anfrage an die Entscheidungskomponente,
- Entgegennahme der Entscheidung und
- Protokollierung des Ereignisses mit Entscheidung.

Im Gegensatz zum Ansatz des IETF werden von der realisierten Durchsetzungskomponente bis auf die Protokollierung keine administrativen Aufgaben übernommen, wodurch die Komponente so einfach wie möglich gehalten wird. Dies ist von Bedeutung, da der PEP im Gegensatz zur Entscheidungskomponente von der Art des Zielobjekts und dem Kommunikationsmechanismus zwischen der aufrufender Einheit und dem Zielobjekt abhängig ist. Je nach darunter liegendem Kommunikationsmodell

(z.B. CORBA, RPC, SOAP, HTTP) muss diese Komponente den Gegebenheiten angepasst werden.

Es existieren verschiedene Möglichkeiten, auf Anwendungsebene eine Durchsetzungskomponente für verteilte Anwendungen zu implementieren: Proxies und Interceptoren, Policy Agents und Autorisierungsserver.

Interceptoren und Proxies

Interceptoren und Proxies werden oftmals eingesetzt, um auf Middleware- oder Anwendungsebene eine Zugriffskontrolle zu realisieren. CORBA (siehe Abbildung 46) und DCOM bedienen sich beispielsweise eines so genannten access control Interceptors. Eine Variante eines Interceptors auf Anwendungsebene stellen Proxies dar (siehe Abbildung 46). Diese sind der eigentlichen Anwendung vorgeschaltet und bieten dem Zielobjekt äquivalente Schnittstellen an. Vor der Weiterleitung des Aufrufs an die Anwendung wird die Berechtigung des Aufruft durch die aufrufende Komponente überprüft.

Diesen Ansatz verfolgte beispielsweise Riechmann in seiner Arbeit [Rie99], um seine SMOs (Security Management Object) zu realisieren, eine Art erweiterte Capabilities, welche die Semantik von Objektreferenzen modifizieren. SMOs sollen die Interaktion zwischen Anwendungen überwachen. Beliebige SMOs werden an eine Objektreferenz angeheftet und können so auch mehrere Politiken für ein Objekt realisieren.

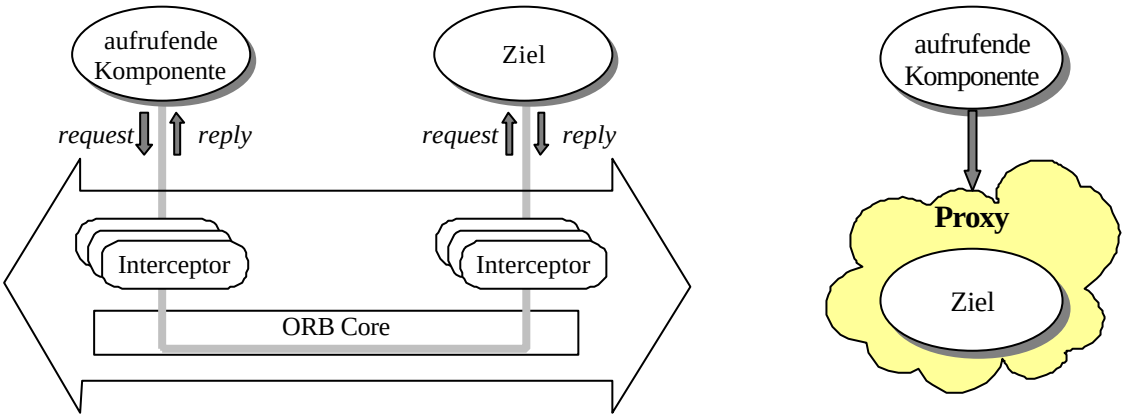

Abbildung 46 CORBA Interceptoren und Proxy-Lösung

Proxies oder Interceptoren haben den Vorteil, dass die Funktionalität einer Objektschnittstelle erweitert werden kann, ohne Änderungen an der Anwendung selbst vornehmen zu müssen. Die Zugriffsentscheidung wird lokal getroffen und verschlechtert somit die Skalierbarkeit des Systems nicht. Ein Nachteil von Interceptoren und Proxies kann sein, dass die Granularität des Zugriffs gegenüber der Schnittstelle der Anwendung nicht erhöht werden kann. Der Hauptnachteil liegt jedoch in der Administration. Durch die Verteilung der Zugriffskontrolle auf jede zu kontrollierende Anwendung, wird es schwierig und aufwendig, die Konsistenz der Zugriffspolitiken für einzelne Zielobjekte zu gewähren. Um dieses Problem zu lösen, wurde beispiels-

weise von Hailpern und Ossher [HO90] das View-Konzept vorgeschlagen und von Barkley [Bar95] das dem View-Konzept ähnliche Rollen-Konzept.

Politik-Agenten

Ein weiterer Ansatz, welcher das Problem der Konsistenzerhaltung von Zugriffspolitiken bei verteilten Durchsetzungskomponenten nicht hat, sind die so genannten Politik-Agenten (engl. Policy-Agents). Unter dem Begriff „Policy-Agent“ versteht man zentrale Mechanismen in der den Anwendungen unterliegenden Infrastruktur. Diese können als Betriebssystem-Add-Ons, in der Sicherungsschicht des Datenmanagementsystems oder in der Middleware realisiert werden. Je Station wird eine Durchsetzungskomponente zentral eingesetzt, die die benötigten Zugriffspolitiken in eine für das Endsystem verständliche Form übersetzt und verteilt [Bez00]. Eine Implementierung dieser Variante ist beispielsweise der Policy Mediator der Universität von Tulsa [GBH+00].

Durch diesen zentralistischen Ansatz in einer Station wird die Fehleranfälligkeit gemindert. Die Verteilung der Zugriffskontrolle lokal auf die einzelnen Stationen erhöht die Skalierbarkeit und die Performanz gegenüber einem systemweiten zentralistischen Ansatz. Handelt es sich um ein sehr heterogenes verteiltes System, so kann die Übersetzung der Politiken in eine der Zielstation verständliche Form einen enormen administrativen Aufwand zur Folge haben. Ein Neustart nach Download der Politik in eine Station kann – je nach Implementation – einen Neustart der Durchsetzungskomponente erfordern, was zu Performanzproblemen führt [Bez00].

Autorisierungsserver

Eine weitere Alternative für eine mögliche Implementierungsart einer Durchsetzungskomponente stellen ein Autorisierungsserver dar. Dieser bieten einen zentralen Autorisierungsdienst in der Regel für eine gesamte Politik-Domäne an. Dieser wird entfernt aufgerufen und trifft die Entscheidung, welche an die Durchsetzungskomponente zurückgeliefert wird. Bekannte Projekte sind beispielsweise das GFAC (Generalized Framework for Access Control) der MITRE Corporation [AEL+90], der Autorisierungsserver von HP [VCP98] oder Adage (Authorization Toolkit for Distributed Applications and Groups) [OSF98].

Die Entscheidungsfindung wird vollständig von der Anwendung entkoppelt und kann dadurch beliebige Granularität erhalten. Der zentralistische Ansatz garantiert die Konsistenz der Politiken, da die Erstellung und Speicherung an einer Stelle erfolgt und keine Verteilung erforderlich ist. Der daraus entstehende Nachteil ist ein erhöhte Kommunikationsbedarf und die damit verbundenen Performanzprobleme, nämlich erhöhte Antwortzeiten und Fehleranfälligkeit. Um einen Totalausfall durch einen Ausfall des Autorisierungsservers vorzubeugen, wird Replikation erforderlich, die den Vorteil der konsistenten Datenhaltung etwas relativiert.

Lösungsansatz

Betrachtet man das zugrunde liegende Einsatzszenario (siehe Abschnitt 2.2), so können zwei sinnvolle Alternativen der Implementierung auf Anwendungsebene identifiziert werden: eine Implementierung als Proxy-Objekt und eine im HTTP-Server integrierte Lösung als Policy Agent. Aus Performanzgründen wurde für den entwickelten Prototypen die zweite Alternative gewählt. Da jeder Dienst auf einer Station über den HTTP-Server aufgerufen wird, ist die Realisierung eines zentralen Policy Agents kein Problem. Bei der Realisierung wurde darauf geachtet, dass eine Anpassung an die erste Alternative ohne große Probleme möglich ist.

8.4.2 Entscheidungskomponente

Die Entscheidungskomponente – auch Policy Decision Point (PDP) genannt – ist die Kernkomponente der Zugriffskontrolle. Jede Station enthält mindestens eine Instanz eines PDPs, welche die Anfragen aller Durchsetzungskomponenten (PEPs) dieser Station bedient. Stehen genügend Hard- und Softwareressourcen in dieser Station zur Verfügung, so können mehrere PDPs angeboten werden, um die Antwortzeiten zu verbessern. Es ist jedoch nicht möglich, dass ein PEP mehrere PDPs konsultiert.

Problematisch ist es, wenn die Ressourcen der Station nicht ausreichen, um mindestens einen PDP zu integrieren. Dann muss dieser als zentrale Komponente realisiert und entfernt aufgerufen werden. Diese Variante verursacht jedoch eine verzögerte Antwortzeit und erfordert eine Absicherung der Kommunikationsverbindung, wodurch wiederum Ressourcen verbraucht werden. Es ist daher abzuwägen, welche dieser beiden Alternativen die optimale für die betrachtete Station ist.

Die Hauptaufgaben der Entscheidungskomponente sind:

- Management der lokalen Zugriffspolitiken, Objektdaten sowie der nicht über den Kontextserver ermittelten Kontextdaten,
- Kommunikation mit dem zentralen Management-Server zur Aktualisierung der lokalen Daten (vgl. Kapitel 8.3),
- Entgegennahme einer Anfrage von der Durchsetzungskomponente und Abarbeitung der entsprechenden Politik-Regeln,
- Ermittlung von Kontextdaten und Abfrage von Kontextdaten vom Kontextserver sowie
- Rückgabe der Entscheidung an die Durchsetzungskomponente.

Die beiden ersten Punkte werden durch die lokale Datenhaltungskomponente des Politikmanagements übernommen. Diese wurde – im Gegensatz zu vielen anderen Implementierungen – als Teil der Entscheidungskomponente und nicht der Durchsetzungskomponente realisiert, da auch die Entscheidungskomponente den Zugriff auf die Daten benötigt, welche für die Zugriffsentscheidung erforderlich sind.

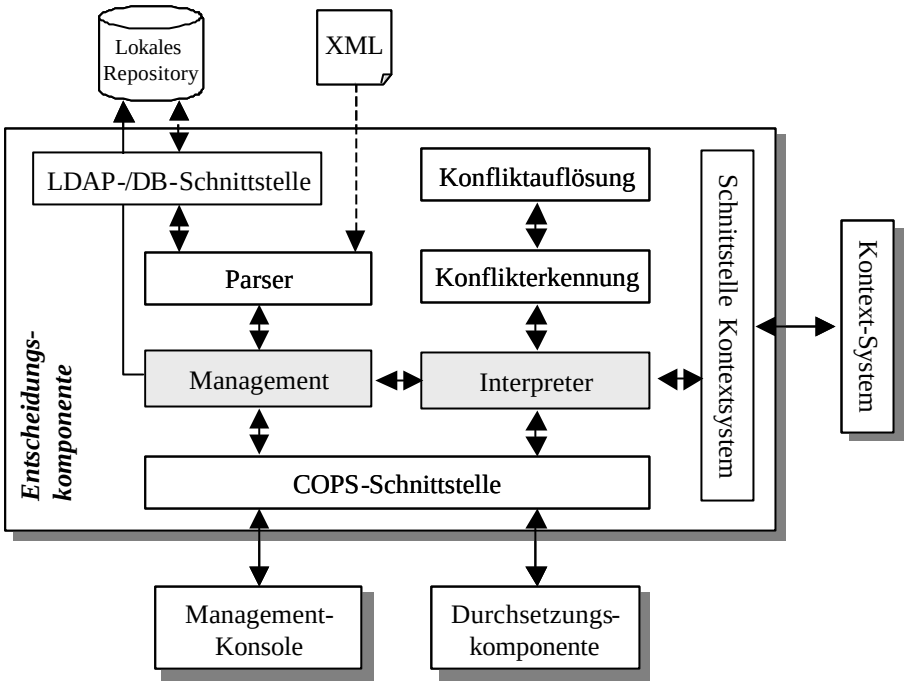

Abbildung 47 Entscheidungskomponente

Die Entscheidungskomponente besteht aus den zwei Hauptkomponenten Management und Interpreter. Das Management nimmt die Informationen von der zentralen Management-Konsole entgegen, parst die Politiken und kontrolliert das lokale Datenmanagement. Der Interpreter wird durch eine Anfrage der Durchsetzungskomponente aktiviert. Er bestimmt die zutreffenden Politikregeln, führt diese aus, bestimmt notwendige Kontextinformationen durch entsprechende Anfragen an das Kontext-System, erkennt zur Laufzeit auftretende Konflikte und löst diese. Anschließend liefert er die ermittelte Entscheidung an die Durchsetzungskomponente zurück.

8.4.3 Authentifikation und Sessionmanagement

Eine herkömmliche Zugriffskontrolle basiert auf eine vorangehende Authentifikation des Benutzers. Die Authentifikation bestätigt zuverlässig die Informationen über die Identität des Benutzers und liefert seine evtl. vorhandenen Gruppen- oder Rollenzugehörigkeiten. Da das Konzept einer kontextabhängigen Zugriffskontrolle eine Ergänzung herkömmlicher Methoden darstellt, muss dieser Grundsatz auch hier gelten.

In der vorliegenden Implementierung erfolgt die Authentifikation eines Benutzers über ein Passwort, das über eine SSL-gesicherte Verbindung übertragen wird. Eine Integration einer Authentifikation über ein X.509-Zertifikat stellt lediglich ein Ressourcenproblem dar, da zusätzliche Komponenten zur Verwaltung der Zertifikate notwendig sind. Eine frei verfügbare Erweiterung des Apache-Web-Servers beispielsweise ist Akenti [Akenti]. Akenti erlaubt die Authentifikation über ein X.509- oder ein spezielles Akenti-Zertifikat. Akenti stellt sowohl die Authentifikationsmechanismen wie auch Verwaltungskomponenten für Zertifikate zur Verfügung.

Erfolgt ein Zugriff auf eine Ressource, welche eine Authentifikation erfordert, werden die notwendigen Schritte durchgeführt und bei Erfolg eine Session aufgebaut. Die ermittelten Benutzerdaten werden in einem so genannten Sessionkontext gespeichert und sind für die Entscheidungskomponente zugreifbar. Die Session wird entweder aktiv durch den Benutzer beendet oder passiv durch die Überschreitung eines Zeitlimits.

Kapitel 9

Zusammenfassung und Ausblick

9.1 Zusammenfassung

Durch die zunehmende Miniaturisierung der Computertechnologie ist es möglich, beliebige alltägliche Gegenstände und die Umgebung mit Prozessoren und kleinste Sensoren auszustatten. Durch diese Durchdringung unserer alltäglichen Umgebung mit Rechenkapazität in Verbindung mit der totalen Vernetzung wird die Umgebung „smart“ und ist in der Lage, auf den Menschen und ihre Bedürfnisse zu reagieren. Neue Arten von Anwendungen entstehen, die den Menschen in seinen alltäglichen Tätigkeiten transparent unterstützen und mit ihm kooperieren. Diese Anwendung einer ubiquitären Kommunikationsinfrastruktur erlaubt einen unaufdringlichen Zugang zu wichtigen Informationen, Ressourcen und Diensten. Dabei ist sicherzustellen, dass der Zugriff auf die Informationen und Dienste nur autorisierten Personen möglich ist, ohne dass diese mit komplexen Sicherheitsmechanismen und Zugangsbeschränkungen belastet und in ihren Handlungen unnötig eingeschränkt werden.

Herkömmliche Ansätze zur Autorisierung von Zugriffen auf Daten und Diensten erfordern eine vom Benutzer aktiv durchgeführte Authentisierung, die nach den Prinzipien „etwas tragen” oder „etwas wissen” arbeitet. Dies wirft für den Bereich des Ubiquitous Computing zwei Probleme auf:

1. In einem offenen System, wie es im Ubiquitous Computing oftmals vorliegt, sind Benutzer zu berücksichtigen, die dem IT-System a priori nicht bekannt sind und daher über keine Authentifikations- und Autorisationsdaten verfügt.
2. Das Ubiquitous Computing propagiert neue Mensch-Maschine-Interaktionsformen, die benutzerfreundlich sind und nur wenige Aktionen durch den Nutzer erfordern.

Der Lösungsansatz, der in dieser Arbeit verfolgt wurde, basiert auf der These, dass viele der ubiquitären Anwendungen und angebotenen Informationen kontextabhängig sind und somit auch der Zugriff darauf durch die Verwendung von Kontextinformationen abgesichert werden kann. Eine Autorisierung basierend auf einer

vorangehenden Authentisierung ist für ubiquitäre Anwendungen und Informationen nur selten notwendig.

Die Verwendung der Begriffe „Kontext“, „Kontextinformationen“ und „Kontextabhängigkeit“ in der Literatur wurde erläutert und spezifiziert, wie sie in dieser Arbeit zu verstehen sind und welche Kontextarten bei einer kontextabhängigen Zugriffskontrolle berücksichtigt werden müssen. Darauf aufbauend wurde eine Methode zur Spezifikation von Kontextinformationen entwickelt.

Es wurde gezeigt, dass die Verwendung von wenigen Kontextdaten wie Zeit und Lokation zur Einschränkung des Zugriffs auf Informationen und Dienste im Ansatz schon existiert und sinnvoll ist. Doch erst die zunehmende Durchdringung der Umgebung mit Rechenkapazität und neuartiger Sensoren ermöglicht ein breites Spektrum an Kontextinformationen und ermöglicht damit aber auch neue Wege in der Verarbeitung und Bereitstellung von Kontextinformationen für den Bereich der Zugriffskontrolle.

Es wurde eine neue Spezifikationssprache zur Beschreibung einer kontextabhängigen Zugriffspolitik entwickelt. Da bisher Kontextinformationen für die Zugriffskontrolle nur in sehr begrenztem Umfang verwendet wurden, sind heutige Spezifikationssprachen zur Festlegung einer Zugriffspolitik nicht ausdrucksstark genug, um Einschränkungen durch Kontextinformationen beliebiger Art zu spezifizieren.

Die Verifikation des Konzepts einer kontextabhängigen Zugriffskontrolle erfolgte durch die Entwicklung eines Prototyps einer Kontextinfrastruktur, welche interessierten Anwendungen beliebige Kontextinformationen zur Verfügung stellt, sowie durch Entwicklung eines kontextabhängigen Zugriffskontrollsystems, das über eine sichere Schnittstelle auf die Informationen des Kontextsystems zugreifen kann und die Autorisierungsüberprüfung anhand dieser Kontextinformationen durchführt.

9.2 Ausblick

Eine Weiterentwicklung dieser Arbeit kann auf beiden Schwerpunkten – der Spezifikation der Zugriffspolitik und dem Kontextsystem – erfolgen. Einige Erweiterungsmöglichkeiten seien hier kurz vorgestellt.

Die Spezifikation der Zugriffspolitik erfolgt unabhängig von der darunterliegenden Standard-Sicherheitspolitik. Es wurde gezeigt, dass verschiedene Sicherheitsmodelle spezifiziert werden können, doch ist in einigen Bereichen, wie beispielsweise der Spezifikation von Sicherheitszertifikaten und eines „Web-of-Trust”, noch Forschungsleistung zu erbringen.

Bei der Realisierung des Kontextsystems konnten nur wenige Sensoren und Kontextarten verwendet werden. Es ist daher sinnvoll, diese Erfahrung auszuweiten, um die Skalierbarkeit unter erweiterten Bedingungen verifizieren zu können.

Die Entwicklungen in den Bereichen Semantikspezifikation und Ontologien stehen noch am Anfang. Beschreibungssprachen, die auch die Möglichkeit der Inferenz von Wissen ermöglichen, sind in der Entwicklung, doch noch nicht in einer stabilen und endgültigen Version verfügbar. Geeignete Editoren fehlen. Daher kann

der vorgestellte Ansatz einer einheitlichen Spezifikation von Kontextinformationen nur als erster Schritt betrachtet werden.

Bei der Entwicklung der Kontextinfrastruktur wurde ein spezielles Raummodell verwendet. Damit jedoch vorhandene Raummodelle wie DNS oder GPS integriert werden können, sind geeignete Algorithmen und Beschreibungssprachen zu entwickeln, die eine Überführung von Daten eines Raummodells in andere Modelle erlauben.

Literaturverzeichnis

Literatur

[3GP00] The 3rd Generation Partnership Project (2000). Universal Geographical Area Description (GAD), Technical Specification, Release 1999, 3G TS 23.032, v. 3.1.0, Mai 2000, http://www.quintillion.co.jp/3GPP/

[Adage99] Adage System Overview. Technical Report, The Open Group Research Institute, Eleven Cambridge Center, Cambridge, MA 02142, 1999

[AEL+90] Abrams, M.D.; Eggers, K.W.; LaPadula, L.J.; Olsen, I.M.: A Generalized Framework for Access Control: An Informal Description. In: Proceedings of the 13th National Computer Security Conference, Oktober 1990, S. 135-143

[AGS+93] Adams, Norman; Gold, Rich; Schilit, Bill N.; Tso, Michael; Want, Roy: An Infrared Network for Mobile Computers. In. Proceedings of the USENIX Symposium on Mobile and Location-independent Computing, Cambridge, MA, August 1993, S. 41-52

[Ahn99] Ahn, Gail-Joon: The RCL 2000 Language for Specifying Role-Based Authorization Constraints. Dissertation, George Mason University Fairfax, Virginia, USA, 1999, http://www.list.gmu.edu/dissert/diss-gail.pdf

[All83] Allen, James F.: Maintaining Knowledge about Temporal Intervals. In: Communications of the ACM 26 (1983), Nr. 11, pp. 832–843

[AQM+97] Abiteboul, S.; Quass, D.; McHugh, J.; Widom, J.; Wiener, J.: The Lorel Query Language for Semistructured Data. In: Inter. Journal on Digital Libraries, 1(1), April 1997, S. 68-88

[Bar94] Barners-Lee, T.: Universal Resource Identifiers in WWW, A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web. Internet Requests For Comments (RFC) 1630, Juni 1994

[Bar95] Barkley, J.: Implementing Role-based Access Control Using Object Technology. In: The First ACM Workshop on Role-Based Access Control, Gaithersburg MD 20899, Nov. 1995, http://www.itl.nist.gov/div897/staff/barkley/rbacot/titlewkshp.html

[BBF+99] Bertino, Elisa; Buccafurri, Francesco ; Ferrari, Elena ; Rullo, Pasquale : A Logical Framework for Reasoning on Data Access Control Policies. In: Proceedings of the 1999 IEEE Computer Security Foundations Workshop, Mordano, Italy, 1999

[BBM95] Buvac, S.; Buvac, V.; Mason, I.A.: Metamathematics of contexts. In: Fundamenta Informaticae 23(3), 1995

[BC00] Bonifati, Angela; Ceri, Stefano: Comparative Analysis of Five XML Query Languages. In: SIGMOD Record, vol. 29, no.1, 2000, S. 68-79

[BC01] Byun, Hee Eon; Cheverst, Keith: Exploiting User Models and Context-Awareness to Support Personal Daily Activities. In: Proceedings of Workshop in UM2001 on User Modelling for Context-Aware Applications, Sonthofen, Germany, 2001

[BDSG01] Bundesdatenschutzgesetz (BDSG). http://www.datenschutzzentrum.de/material/recht/bdsg/bdsg.htm, http://www.datenschutzzentrum.de/material/recht/bdsg2001/bdsg2001.htm

[Bez00] Benosov, Konstantin: Engineering Access Control For Distributed Enterprise Applications. Dissertation, Florida International University, Miami, Florida, 2000

[BG01] Brickley, Dan;. Guha, R.V.: Resource Description Framework (RDF) Schema, Specification 1.0. W3C Candidate Recommendation 27 March 2001, http://www.w3.org/TR/rdf-schema/

[BHM97] Bracio, Boris R. ; Horn, Wolfgang ; Moller, Dietmar P.F.: Sensor Fusion in Biomedical Systems. In: Proceedings - 19th International Conference - IEEE/EMBS, Chicago, IL. USA, 1997

[BI97] Brooks, Richard R.; Iyengar, Sundararaja, S.: Multi-sensor fusion: Fundamentals and applications with software. 1997

[BL73] Bell, D.E.; LaPAdula, L.J.: Secure Computer Systems: Mathematical Foundations and Model. Technical Report M74-244, MITRE Corp., Bedford MA, 1973

[BL75] Bell, D.E.; LaPAdula, L.J.: Unified exposition and Multics interpretation. Technical Report ESD-TR-75-306, The Mitre Corporation,Bedford, MA, March 1975

[BM01] Biron, P.; Malhotra, A.: XML Schema Part 2: Datatypes. W3C Recommendation 02 May 2001, http://www.w3.org/XML/Group/-xmlschema-current/datatypes/datatypes.html, 21 September, 1999

[BN89] Brewer, D.F.C.; Nash, M.J.: The Chinese Wall Security Policy. In: Proceedings 1989 IEEE Symposium on Security and Privacy, 1989, S. 206-214

[BPS+00] Bray, T.; Paoli, J.; Sperberg-McQueen, C. M.; Maler, E.: Extensible Markup Language (XML) 1.0. W3C Recommendation 6 October 2000, ttp://www.w3.org/TR/REC-xml

[Bre99] Brézillon, Patrick: Context in Artificial Intelligence: I. A Survey of the literature. In: Computer & Artificial Intelligence, 18(4), 1999, S. 321-340

[Bro96a] Brown, P. J.: The stick-e document: a framework for creating context-aware applications. In: Proceedings of EP'96, Palo Alto, Sept 1996, S. 259-272

[Bro96b] Brown, P. J.: Facilitating the creation of context-aware applications. Computing Lab., Univ. of Kent at Canterbury, 1996

[BSI89] Bundesamt für Sicherheit in der Informationstechnik (BSI): IT-Sicherheitskriterien: Kriterien für die Bewertung der Sicherheit von Systemen der Informationstechnik (IT) – 1. Fassung. 1989

[BY97] Bevier, William R.; Young, William D.: A constraint language for adage. Technical report, Computational Logic, Inc., April 1997

[Car98] Carzaniga, A.: Architectures for an Event Notification Service Scalable to Wide-area Networks. Dissertation, Politecnico di Milano, Milano, Italy, December 1998

[CAS01] Covington, Michael J.; Ahamad, Mustaque; Srinivasan, Srividhya: A Security Architecture for Context-Aware Applications. Georgia Tech, College of Computing, Technical Report GIT-CC-01-12, Mai 2001

[CCD+01] Cox, Simon; Cuthbert, Adrian; Daisey, Paul; e.a.: Geography Markup Language (GML) 2.0. OpenGIS Implementation Specification, OGC Document Number: 01-029, 20 February 2001, 2001, http://www.opengis.net/gml/01-029/GML2.html,

[CCP11-98] Common Criteria for Information Technology, Security Evaluation Part 1: Introduction and general model. 15 November 1998

[CC99] CC Project: Common Criteria for Information Technology Security Evaluation Criteria (CC), Version 2.0. 1999

[CDCE97] Dublin Core metadate Initiative: Coverage Element Working Draft. Sept. 1997, http://dublincore.org/documents/1997/09/30/coverage-element/

[CDF98] Cugola, G.; Di Nitto, E.; Fuggetta, A.: The JEDI event-based infrastructure and its application to the development of the OPSS WFMS. Technical report, CEFRIEL - Politecnico di Milano, Italy, August 1998

[CFM+00] Chamberlin, D.; Fankhauser, P.; Marchiori, M.; Robie, J.: XML Query Requirements. W3C Working Draft 15 August 2000, http://www.w3.org/TR/2000/WD-xmlquery-req-20000815

[Chi89] Chin, D. N.: KNOME: Modeling what the User Knows in UC. In: Kobsa & Wahlster, 1989

[CK00] Chen, Guanling; Kotz, David: A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, Nov. 2000

[CMA00] Covingtony, Michael J.; Moyer, Matthew J.; Ahamad, Mustaque: Generalized Role-Based Access Control for Securing Future Applications. In: Proceedings of the 23rd National Information Systems Security Conference (NISSC), Baltimore, Maryland, USA, October 2000, S. 40-51

[Corba2.0] OMG: CORBAservices: Common Object Services Specification. In: November 1997, http://www.infosys.tuwien.ac.at/Research/-Corba/archive/docu/97-12-02.pdf.gz

[CorbaES01] Object Management Group: CORBAservices: Common Object Services Specification - Event Service Specification, Version 1.1. http://www.omg.org/cgi-bin/doc?formal/01-03-01.pdf

[Cox00] Cox, Simon: DCMI Period Encoding Scheme: specification of the limits of a time interval, and methods for encoding this in a text string. Juli 2000, http://dublincore.org/documents/2000/07/11/dcmi-period/

[Cri89] Cristian, F.: Probabilistic Clock Synchronization. In: Distributed Computing 3(3), Springer, 1989, S. 146-158

[CS95] Chen, F. ; Sandhu, R.: Constraints for role based access control. In: Proceedings of the 1st ACM Workshop on Role-Based Access Control, Gaithersburg, MD, Nov. 1995, S. 39-46

[CW87] Clark, D.D.; Wilson, D.R.: A Comparison of Commercial and Military Computer Security Policies. In: Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA. May 1987. Apr. 1987, S. 184-194

[DA00a] Dey, Anind K.; Abowd, Gregory D: Towards a Better Understanding of Context and Context-Awareness. In: Proceedings of Workshop on The What, Who, Where, When, and How of Context-Awareness, as part of the 2000 Conference on Human Factors in Computing Systems (CHI 2000), The Hague, Netherlands, April 3 2000, ftp://ftp.cc.gatech.edu/pub/gvu/tr/1999/99-22.pdf

[DA00b] Dey, A. K.; Abowd, G. D.: The Context Toolkit: Aiding the Development of Context-Aware Applications. In: Proceedings of Workshop on Software Engineering for Wearable and Pervasive Computing, Limerick, Ireland, 2000

[Dan00] Dana, Peter H.: Global Positioning System Overview. 2000, http://www.colorado.Edu/geography/gcraft/notes/gps/gps_f.html

[DBm01] DER BROCKHAUS multimedial 2001. Bibliographisches Institut & F. A. Brockhaus AG, 2001

[DBS+98] Decker, St.; Brickley, D.; Saarela, J.; Angele, J.: A Query and Inference Service for RDF. http://www.w3.org/TandS/QL/QL98/pp/queryservice.html

[DCOM96] Microsoft Corporation: DCOM Technical Overview. 1996, http://www.microsoft.com/vfoxpro/sourcebook/whitepapers/dcomtech.htm

[DDL+00] Damianou, Nicodemos; Dulay, Naranker; Lupu, Emil; Sloman, Morris: Ponder: A Language for Specifying Security and Management Policies for Distributed Systems - The Language Specification Version 2.3. Imperial College Research Report DoC 2000/1, Okt. 2000, http://www-dse.doc.ic.ac.uk/policies

[DDL+01] Damianou, Nicodemos; Dulay, Naranker; Lupu, Emil; Sloman, Morris: The Ponder Policy Specification Language. In: M. Sloman, J. Lobo, and E. Lupu (Eds.): POLICY 2001, LNCS 1995, 2001, S. 18-38

[Den76] Denning, D.E.: A lattice modell of secure information flow. In: Communications of ACM, 19(5), 1976, S. 236-243

[Dey00] Dey, Anind K.: Providing Architectural Support for Building Context-Aware Applications. Dissertation, College of Computing, Georgia Institute of Technology, Dez. 2000

[DFn01] Duden Fremdwörter neu. Bibliographisches Institut & F. A. Brockhaus AG und Langenscheidt KG, Dudenverlag Mannheim/Wien/Zürich, 2001

[DG97] Dalton, C.I; Griffin, J.F.: Applying Military Grade Security to the Internet. In: Computer Networks and ISDN Systems 29(15), 1997, S. 1799-1808

[DH98] Dawson, Frank; Hoffman, Paul: The vCard v3.0 XML DTD. Internet Draft draft-dawson-vcard-xml-dtd-02.txt, Nov. 1998, http://www.globecom.net/ietf/draft/draft-dawson-vcard-xml-dtd-02.html

[DoD85] Department of Defense: Trusted Computer System Evaluation Criteria (Orange Book). Department of Defense Trusted Computer System Evaluation Criteria, Dezember 1985, http://www.acme.ibilce.unesp.br/documents/sec/sec-trusted_computer.pdf

[DS98] Dawson, F.; Stenerson, D.: Internet Calendaring and Scheduling Core Object Specification (iCalendar). Request for Comment 2445, November 1998, http://www.ietf.org/rfc/rfc2445.txt

[DSA+99] Dey, A.K.; Salber, D.; Abowd, G.D.; Futakawa, M.: The Conference Assistant: Combining context-awareness with wearable computing. In: 3rd International Symposium on Wearable Computers, San Francisco, California, Okt. 1999, S. 21-28

[DTF97] Wolf, Misha; Wicksteed, Charles: Date and Time Format, S. 1997, http://www.w3.org/TR/NOTE-datetime

[DVG+96] Davis, C.; Vixie, P.; Goodwin, T.; Dickinson, I.: A Means for Expressing Location Information in the Domain Name System. RFC 1876, Jan. 1996, http://www.faqs.org/rfcs/rfc1876.html

[EAC98] Edjlali, Guy; Acharya, Anurag; Chaudhary, Vipin: History-based Access-control for Mobile Code. In: Proceedings of the Fifth ACM Conference on Computer and Communications Security. San Francisco, CA, USA, Nov. 1998, http://www.cs.ucsb.edu/~acha/publications/ccs98-submitted.html

[EBK00] Espinal, Luis; Beznosov, Konstantin; Deng, Yi: Design and Implementation of Resource Access Decision Server. Miami Technical Report 2000-01, Florida International University, Jan. 2000

[Eck96] Eckert, Claudia: Leitlinien zur Klassifikation und Bewertung von Sicherheitsmodellen. In: Proceedings der Fachtagung Sicherheit in Informationssystemen, SIS'96, Wien, Österreich, März 1996, http://www.informatik.uni-bremen.de/grp/ag-sec/Personen/Eckert/-Papers/eck_WienSIS96.ps

[Eck97] Eckert, Christian: Zustandsabhängige Sicherheitsspezifikation und ihre Durchsetzung. Dissertation, Universität Hildesheim, Fachbereich Mathematik, Informatik, Naturwissenschaften, Shaker Verlag, ISBN 3-8265-2920-0, 1997

[Eck00] Eckert, Claudia: IT-Sicherheit - Konzepte, Verfahren, Protokolle. Oldenbourg Wissenschaftsverlag GmbH, 2000, 3-486-25298-4

[FCK95] Ferraiolo, D.; Cugini, J.; Kuhn, R.: Role Based Access Control: Features and Motivations. In: Proceedings of Annual Computer Security Applications Conference, IEEE Computer Society Press, Baltimore, 1995

[FGL93] Ferraiolo, D.F.; Gilbert, D.M.; Lynch, N.: An Examination of Federal and Commercial Access Control Policy Needs. In: Proceedings of the NIST-NCSC National Computer Security Conference, 1993

[Fid91] Fidge, C.J.: Logical time in distributed computing systems. In: IEEE Computers, 24(8), 28-33, 1991, S. 28-33

[Fin99] Finkenzeller, K.: RFID-Handbook, Wiley, ISBN 0-471-98851-0, 1999

[Fis01] Fischer-Hübner, Simone: IT-Security and Privacy. LNCS 1958, Springer Verlag Berlin Heidelberg, ISBN 3-540-42142-4, 2001

[FK92] Ferraiolo, D.; Kuhn, D.R.: Role-Based Access Control. In: Proceedings, 15th Natl. Computer Security Conference. NIST/NSA, 1992. http://xsun.sdct.itl.nist.gov/rbac/paper/rbac1.html

[FK00] Fink, Josef; Kobsa, Alfred: A Review and Analysis of Commercial User Modeling Servers for Personalization on the World Wide Web. In: User Modeling and User-Adapted Interaction 10(3-4), Special Issue on Deployed User Modeling, 2000, S. 209-249, http://www.ics.uci.edu/%7Ekobsa/papers/2000-UMUAI-kobsa.pdf

[Gas88] Gasser, M.: Building a Secure Computer System. Van Nostrand Reinhold Company, New York, 1988

[GBH+00] Galiasso, P.; Bremer, O.; Hale, J.; Shenoi, S.; Ferraiolo, David F.; Hu, Vincent C.: Policy Mediation for Multi-Enterprise Environments. In: Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC'00), 2000, S. 100-106

[GBK99] Gellersen, Hans-W.; Beigl, Michael; Krull, Holge: The MediaCup: Awareness Technology embedded in an Everyday Object. In: First International Symposium on Handheld and Ubiquitous Computing (HUC99), Karlsruhe, Germany, LNCS 1707, Springer-Verlag, Sept. 1999

[GH95] Gittler, F.;Hopkins, A.C.: The DCE Security Service. Hewlett-Packard Journal, vol. 46(6), 1995, S. 41-48

[GI97] Giuri, L.; Iglio, P.: Role templates for content-based access control. In: Proceedings of the Second ACM Role-Based Access Control Workshop, 1997

[GM82] Goguen, J.A.;. Meseguer, J.: Security Policies and Security Models. In: Proceedings of the 1982 Symposium on Security and Privacy, New York, NY, USA, 1982, S. 11-20

[GPS02a] The Aerospace Corporation: The Global Positioning System - Primer. 2002, http://www.aero.org/publications/GPSPRIMER/GPS-Primer.pdf

[GPS02b] Global Positioning System (GPS) Lab. Stanford University, 2002, http://www.stanford.edu/group/GPS

[Gri94] Grimm, R.: Sicherheit für offene Kommunikation - Verbindliche Tele-kooperation. In: Horster P. (Hrsg.): Sicherheit in der Informations- und Kommunikationstechnik, Band 4, BI Wissenschaftsverlag, Mannheim,1994

[Gro97] Object Management Group: CORBAservices: Common Object Services Specification - chapter 16, Trading Object Service Specification. März 1997, http://www.omg.org/library/csindx.html.

[Guh91] Guha, R.V.: Contexts: A Formalization and Some Applications. Dissertation, Stanford University, 1991

[GXML01] G-XML Platform Construction Project Team: G-XML Geospatial-eXtensible Markup Language. Version 2..0, In: Mai 2001, http://www.dpc.or.jp

[GZ89] Gusella, Riccardo; Zatti, Stefano: The accuracy of the clock cynchronization achieved by TEMPO in Berkeley UNIX 4.3BSD. In: IEEE Transaction on Software Engineering, Vol. 15, No. 7, 1989, S. 847 - 853

[HB00] Hupfeld, Felix; Beigl, Michael: Spatially aware local communication in the RAUM system. In: Proceedings of IDMS 2000, Enschede, Niederlande, October 17-20, 2000, S. 285-296

[Hel98] Held, Albert: Zugriffskontrolle in verteilten mobilen Systemen. Dissertation, Technische Universität Dresden, Fakultät Informatik, Juli 1998

[Hil99] Hills, Alex: Wireless andrew. In: IEEE Spectrum, 36(6), Juni 1999, S. 49-53

[Hil01] Hillmann, Diane: Using Dublin Core. http://dublincore.org/documents/2001/04/12/usageguide

[HK00] Hada, Satoshi; Kudo, Michiharu: XML Access Control Language: Provisional Authorization for XML Documents. Tokyo Research Laboratory, IBM Research, Oct 2000, http://www.trl.ibm.com/projects/xml/xacl/xacl-spec.html

[HKL+99] Hohl, Fritz; Kubach, Uwe; Leonhardi, Alexander; Rothermel, Kurt; Schwehm, Markus: Next Century Challenges: Nexus - An Open Global Infrastructure for Spatial-Aware Applications In: Proceedings of Mobicom '99, Seattle, Washington, USA, T. Imielinski, M. Steenstrup, (Eds.), ACM Press, 1999, S. 249-255

[HMS+96] Hurley, Marty; Meta, Nimsha; Simon, Rich; ZurkoMary Ellen: Authorization for distributed applications and groups. Technical report, OSF, 1996

[HMS+00] Huang, H.; Meissner, A.; Schönfeld, W.; Steinmetz, R.: QoS Policy Framework and its Implementation. In: Proc. of IEEE International Conference on Communication Technology, Vol. 1, 2000, 860-867

[HNB97] Hull, Richard; Neaves, Philip; Bedford-Roberts, James: Towards Situated Computing. In: Proceedings of the 1st International Symposium on Wearable Computers (ISWC '97), Okt. 1997, S. 146-153, http://wearables.cs.bris.ac.uk/public/papers/tositcom2.htm

[HO90] Hailpern, B.; Ossher, H.: Extending Objects to Support Multiple Interfaces and Access Control. In: IEEE Transactions on Software Engineering, vol. 16, 1990, S. 1247-1257

[Hoa00] Hoagland, James: Specifying and Implementing Security Policies using LaSCO, the Language for Security Constraints on Objects. PH.D. Dissertation, University of California, Davis Department of Computer Science, 2000

[Hos92] Hosmer, Hilary H.: Metapolicies II. In: Proceedings of the 15th National Computer Security Conference, NIST-NCSC, United States Government Printing Office, 1992, S. 369-378

[Hos93] Hosmer, Hilary H.: The Multipolicy Paradigm for Trusted Systems. In: Proceedings of the New Security Paradigms Workshop, 1993, S. 19-32

[HPL98] Hoagland, James A.; Pandey, Raju; Levitt, Karl N.: Security Policy Specification Using a Graphical Approach. Technical Report CSE-98-3, 1998

[HPL98b] Hoagland, James A; Pandey, Raju; Levitt, Karl N.: A Graph-based Language for Specifying Security Policies. 1998, http://citeseer.nj.nec.com/144987.html

[HPL01] Hoagland, James A; Pandey, Raju; Levitt, Karl N.: Specifying and Enforcing Policies Using LaSCO: the Language for Security Constraints on Objects. In: Proc. of POLICY 2001, 2001

[KBM+01] Kingberg, T.; Barton, J.; Morgan, J.; at al.: People, Places, Things: Web Presence for the real world. www.cooltown.com/dev/wpapers/-webpres/WebPresence, 2001

[Ian01] Iannella, Renato: Representing vCard Objects in RDF/XML. W3C Note 22 February 2001, http://www.w3.org/TR/vcard-rdf

[ISO8601] ISO 8601:2000 Data elements and interchange formats -- Information interchange -- Representation of dates and times. http://www.iso.ch/

[ITSEC91] Commission of the European Communities: Information Technology Security Evaluation Criteria (ITSEC), 1991, http://www.iwar.org.uk/comsec/resources/standards/itsec.htm

[Jini01] Inc. Sun Microsystems: Jini Distributed Event Specification. 2001, http://www.sun.com/jini/specs/index.html

[KCP+00] Karvounarakis, G.; Christophides, V.; Plexousakis, D.; Alexaki, S.: Querying community web portals. Technical report, Institute of Computer Science, FORTH, Heraklion, Greek, 2000, http://www.ics.forth.gr/proj/isst/RDF/RQL/rql.pdf

[KCP00] Karvounarakis, G.; Christophides, V.; Plexoussakis, D.: Querying Semistructured (Meta)Data and Schemas on the Web: The case of RDF and RDFS. Technical Report 269, ICS-FORTH, Heraklion, Crete, Greece, Feb. 2000

[Ken96] Kenney, John Jerome: Executable Formal Models of Distributed Transaction Systems Based on Event Procecessing. Dissertation, Department of Electrical Engineering, Stanford University, Juni 1996

[KH00] Kudo, Michiharu; Hada, Satoshi: XML Document Security based on Provisional Authorization. In: Proceedings of 7th ACM Conference on Computer and Communication Security (CCS 2000), Nov. 2000

[Kno01] Knorz, Gerhard: Informationswissenschaft. Vorlesungsfolien, Fachhochschule Darmstadt, Fachbereich IuD, http://www.iud.fh-darmstadt.de/iud/wwwmeth/publ/slide/diw1.htm, Nov. 2001

[Kob93] Kobsa, Alfred: User Modeling: Recent Work, Prospects and Hazards. In: M. Schneider-Hufschmidt, T. Kühme and U. Malinowski, eds. (1993): Adaptive User Interfaces: Principles and Practise. Amsterdam: North Holland Elsevier, 1993

[KRW+01] Klyne, Graham; Reynolds, Franklin; Woodrow, Chris; Ohto, Hidetaka: Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies, W3C Working Draft 15 March 2001, http://www.w3.org/TR/CCPP-struct-vocab/

[KS01] Kokkelink, Stefan; Schwänzl, Roland: Expressing Qualified Dublin Core in RDF/XML. 2001, http://dublincore.org/documents/2001/11/30/dcq-rdf-xml/

[KT01] Korkea-aho, Mari; Tang, Haitao: Spatial Location Payload. Internet-Draft draft-korkea-aho-spatial-location-payload-00.txt, Mai 2001, http://www.hut.fi/~mkorkeaa/papers/draft-korkea-aho-spatial-location-payload-00.txt

[Kud01] Kudo, M.: XACML Policy Proposal. http://xml.coverpages.org/-xacmlprop.pdf

[Kün98] Kühnhauser,Winfried: Metapolitiken. Habilitation, ISBN 3-88457-362-4, 1999

[KUJ+01] Kagal, Lalana; Undercoffer, Jeffrey, Joshi, Anupam; Finin, Tim: Vigil : Enforcing Security in Ubiquitous Environments. In: Grace Hopper Celebration of Women in Computing, 2002, http://www.cs.umbc.edu/~lkagal1/papers/vigil.pdf

[Kun99] Kunze, J.: Encoding Dublin Core Metadata in HTML. Request for Comments 2731, Dez. 1999, http://www.ietf.org/rfc/rfc2731.txt

[Lam71] Lampson, B.W.: Protection. In: Proceedings of 5th Princeton Symposium on Information Science and Systems, 1971, S. 437-443

[Lam78] Lamport: Time, Clocks and the Ordering of Events in a Distributed System. In: Communications of the ACM, vol. 21, pp. 558-564, Juli 1978

[LaP90] La Padula, L.: Formal Modeling in a Generalized Framework for Access Control. In: Proc. IEEE Symposium on Security and Privacy, Oakland, CA, 1990

[Leo98] Leonhardt, Ulf: Supporting Location-Awareness in Open Distributed Systems. Dissertation, Department of Computing, Imperial Colleg of Science, Technology ans Medicin, University of London, Mai 1998

[LF94] Lamming, M.; Flynn, M.: Forget-me-not: Intimate Computing in Support of Human Memory. In: Proceedings of FRIEND 21, International Symposium on Next Generation Human Interfaces, Tokyo, 1994, S. 125-128

[LH01] Langenscheidts Handwörterbuch. Langenscheidt KG, Berlin und München, 2001

[Lie01] Liebert, Michael: Rechte, Benutzerrollen und Inhaltsversionierung für verteilte Multimedia-Autorensysteme. Dissertation, Technische Universität Darmstadt, Fachbereich Informatik, Dez. 2001

[LK99] Leonhardi, A.; Kubach, U.: An architecture for a universal, distributed location service. In: Proceedings of the European Wireless '99 Conference, Munich, Germany, VDE Verlag, 1999, S. 351-355

[LKA+95] Luckham, David C.; Kenney, John J.; Augustin, Larry M.; Vera, James; Bryan, Doug; Mann, Walte: Specification and Analysis of System Architecture Using Rapid. In: IEEE Transactions on Software Engineering, 21(4), Apr. 1995, 336-355,

[LKR+99] Leonhardi, Alexander; Kubach, Uwe; Rothermel, Kurt; Fritz, Andreas: Virtual Information Towers - A Metaphor for Intuitive, Location-Aware Information Access in a Mobile Environment. In: Proceedings of the Third International Symposium on Wearable Computers (ISWC'99), San Fransisco, CA, USA, IEEE Press, 1999, S. 15-20

[LLD96] Lagoze, Carl; Lynch, Clifford A.; Daniel, Ron: The Warwick Framework: A Container Architecture for Aggregating Sets of Metadata. Cornell University Technical Report, 1996, http://cs-tr.c,s.cornell.edu/Dienst/UI/2.0/Describe/ncstrl.cornell/TR96-1593

[LM97] Leonhardt, Ulf; Magee, Jeff: Security Considerations for a Distributed Location Service. In: 4th Workshop of the OpenView University Association. Madrid, April 2-4, 1997

[LM98] Leonhardt, Ulf; Magee, Jeff: Security Considerations for a Distributed Location Service. In: Journal of Network and Systems Management, 6(1); 1998, S. 51-70

[LS00] Lieberman, Henry; Selker, Ted: Out of context: Computer systems that adapt to, and learn from, context. In: IBM Systems Journal, Vol. 39, No. 3 & 4 - MIT Media Laboratory, 2000, http://www.research.ibm.com/journal/sj/393/part1/lieberman.html

[LS97] Lupu, E.; Sloman, M.: Conflict Analysis for Management Policies.In: Proceedings of the 5th International Symposium on Integrated Network Management IM '97, San-Diego, USA, 1997, S. 430-443

[LS99] Lupu, E.; Sloman, M.: Conflicts in Policy-based Distributed Systems Management. In: IEEE Transactions on Software Engineering - Special Issue on Inconsistency Management, 25(6), 1999, S. 852-869

[Lup98] Lupu, E. C.: A Role-Based Framework for Distributed Systems Management. Dissertation, Department of Computing, Imperial College, London, U. K, Juli 1998.

[LV93] Luckham, David C.; Vera, James: Event-Based Concepts and Language for System Architecture. 1993, http://citeseer.nj.nec.com/152331.html

[MA01] Moyer, Matthew J.; Ahamad, Mustaque: Generalized Role-Based Access Control. In: Proceedings of the IEEE International Conference on Distributed Computing Systems, Mesa, Arizona, USA, 2001

[Mag01] Magee, Jeff: Location Service for Mobile MultiMedia Environments.ERSRC Location Service Final Report, Research Grant Final Report - Grant GR/L06010, 2001, http://www.doc.ic.ac.uk/~jnm/final-report.html

[Mat88] Mattern, Friedemann: Virtual time and globalstates of distributed systems.In: Proceedings of Parallel and Distributed Algorithms, 1988, S. 215-226

[MB94] McCarthy, John; Buvac, Sasa: Formalizing Context (Expanded Notes). 1994, http://www-formal.stanford.edu

[McC93] McCarthy, J.: Notes on Formalizing Context. In: Proceedings of the IJCAI, Chambéry, 1993, S. 555-560

[MES+01] Moore, B.; Ellesson, E.; Strassner, J.; Westerinen, A.: Policy Core Information Model - Version 1 Specification. IETF, RFC 3060, Feb. 2001, http://www.faqs.org/rfcs/rfc3060.html

[MHR01] Moschgath, Marie-Luise; Hähner, Jörg; Reinema, Rolf: Sm@rtLibrary - An Infrastructure for Ubiquitous Technologies and Applications. In: Proceedings of 21st International Conference on Distributed Computing Systems Workshops (ICDCS 2001), International Workshop on Smart Appliances and Wearable Computing (IWSAWC), Mesa, Arizona, IEEE Computer Society, ISBN 0-7695-1080-9, 2001, S. 208-213

[Mil94] Mills, David L.: Internet Time Synchronization: The Network Time Protocol. In: Zhonghua Yang and T. Anthony Marsland (eds.), Global States and Time in Distributed Systems, IEEE Computer Society Press, 1994, S. 91-102

[Mil01] Miller, Libby: RDF Squish query language and Java implementation. 2001, http://ilrt.org/discovery/2001/02/squish/

[MNS+87] Millers.P.; Neuman, B.C.; Schiller, J. I.; Saltzer,J. H.: Kerberos Authentication and Authorization System. M.I.T. Project Athena, Cambridge, Massachusetts, 1987, ftp://athena-dist.mit.edu/pub/kerberos/doc/techplan.PS

[Mot95] Motschnig-Pitrik, Renate: An Integrated View on the Viewing Abstraction: Contexts and Perspectives in Software Development, AI, and Databases.In: Journal of Systems Integration, 5(1), Apr. 1995, S. 23-60

[MRH00] Moschgath, Marie-Luise; Reinema, Rolf; Hoffmann, Mario: A Security Infrastructure for the Virtual Project Office. In: Proceedings of the IEEE International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2000), FESB-Split, Okt. 2000, S. 563-572

[MS94] Moffett, Jonathan D.; Sloman, Morris.: Policy Conflict Analysis in Distributed System Management. In: Journal of Organizational Computing, 4(1), 1994, S. 1-22

[MS98a] Marchiori, Massimo; Saarela, Janne: Metalog - Query language for RDF. 1998, http://www.w3.org/RDF/Metalog/paper981124.html

[MS98b] Marchiori, Massimo; Saarela, Janne: Query + Metadata + Logic = Metalog. IW3C Query Languages meeting. 1998, http://www.w3.org/TandS/QL/QL98/pp/metalog.html

[MST90] Moffett, Jonathan ; Sloman, Morris; Twidle, Kevin: Specifying Discretionary Access Control Policy for Distributed Systems. Computer Communications, 13(9):571{580, November 1990.

[Nel98] Nelson, Giles J.: Context-Aware and Location Systems. Ph.D. Theses, University of Cambridge, Computer Lab., UK, Jan. 1998

[NVML99] Sekiguchi, et al.: NaVigation Markup Language (NVML). W3C Note 6 Aug 1999, http://www.w3.org/TR/NVML

[OCL97] Object Constraint Language Specification version 1.1 Sept. 1997, http://www.software.ibm.com/ad/ocl

[OMG98] Object Management Group: CORBA Security Service Specification, 1.2 edition, Dez. 1998

[OMG99] Object Management Group: Resource Access Decision (RAD). OMG document number:cor-bamed/99-05-04.10, Mai 1999

[Opp97] Oppliger, Rolf: IT-Sicherheit - Grundlagen und Umsetzung in der Praxis. DuD Fachbeiträge, 1997, ISBN 3-528-05566-9

[P3P01] The Platform for Privacy Preferences 1.0 (P3P1.0) Specification. W3C Working Draft 28 Sept. 2001, http://www.w3.org/TR/2001/WD-P3P-20010928/

[Pas97] Pascoe, Jason: The Stick-e Note Architecture: Extending the Interface Beyond the User. In: Proceedings of International Conference on Intelligent User Interfaces, In Johanna Moore, Ernest Edmonds, Angel Puerta (Eds.), 1997, S. 261-264

[PRM98] Pascoe, J.; Ryan, N.; S. ; Morse, D.R.: Human Computer Giraffe Interaction - HCI in the Field. In: Workshop on Human Computer Interaction with Mobile Devices, 1998

[PRM99] Pascoe, J.; Ryan, N.; S.; Morse, D.R.: Issues in Developing Context-Aware Computing. In: Proceedings of the International Symposium on Handheld and Ubiquitous Computing, Springer-Verlag, Sept. 1999, S. 208-221

[RAD99] Object Management Group Healthcare Domain Task Force: Resource Access Decision (RAD), Revised Submission, OMG TC Document corbamed/99-04-04, Apr. 1999

[RBB+00] Reinema, R.; Bahr, K.; Burkhardt, H.-J.; Hovestadt, L.: Cooperative Rooms -- Symbiosis of Real and Virtual Worlds. In: Proceedings of 8th International Conference on Telecommunication Systems, Modelling and Analysis, ATSMA, Nashville, Texas, 2000, S. 181-187

[rdfDB01] Guha, R.V.: rdfDB. 2001, http://web1.guha.com/rdfdb/

[Rei01] Reinema, Rolf: WWFM - Worldwide Facility Management. In: O. Gassmann, H. Meixner (Eds.): Sensors in Intelligent Buildings. Volume 2, WILEY-VCH Verlag, 2001, S. 469-481, ISBN 3-527-29557-7

[Rei02] Reinema, Rolf: Co-operative Workplaces - Workspaces of the Future. Dissertation, Fachbereich Informatik, Technische Universität Darmstadt, April 2002

[Rho97] Rhodes, B.J.: The wearable remembrance agent: A system for augmented memory. In: Personal Technologies Journal Special Issue on Wearable Computing, Personal Technologies, 1997, S. 218-224

[Ric79] Rich, Elaine: User Modeling via Stereotypes. In: Cognitive Science 3, 1997, S. 329-354

[Rie99] Riechmann, Th.: Sicherheit in verteilten, objektorientierten Systemen. Dissertation, Technische Fakultät, Universität Erlangen-Nürnberg, 1999

[RLF00] Rakotonirainy, A.; Loke, S.W.; Fitzpatrick, G.: Context-Awareness for the Mobile Environment. Position Paper in the Workshop on The What, Who, Where, When, and How of Context-Awareness, as part of the 2000 Conference on Human Factors in Computing Systems (CHI 2000), The Hague, The Netherlands, April 3, 2000

[RLS98] Robie, Jonathan; Lapp, Joe; Schach, David: XQL Proposal. 1998, http://www.w3.org/TandS/QL/QL98/pp/xql.html

[RMB+99] Reinema, R.; Moschgath, M.-L.; Bahr, K.; Hovestadt, L.: roomComputers - Bridging Spaces. In: Proceedings of 7th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS '99), Cape Town, South Africa, Dez. 1999

[RMH00] Reinema, Rolf; Moschgath, Marie-Luise; Hoffmann, Mario: Eine Sicherheitsinfrastruktur für das Virtuelle Projektbüro. In: Sicherheit in Netzen und Medienströmen, M. Schumacher; R. Steinmetz (Hrsg.). Springer-Verlag Berlin Heidelberg, Sept. 2000, S. 73-82

[RT02] Reinema, R.; Thielmann, H.: RoomComputer - Ubiquitous Computing in Cooperative Rooms. In: thema Forschung, Technische Universität Darmstadt, Darmstadt, Jan. 2002

[SA98] Salber, Daniel; Abowd, Gregory D.: The Design and Use of a Generic Context Server. In: Proceedings of the Perceptual User Interfaces Workshop (PUI '98), San Francisco, CA, Nov. 1998, S. 63-66

[SA99] Stajano, Frank; Anderson, Ross: The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In: Security Protocols, 7th International Workshop Proceedings, Lecture Notes in Computer Science, 1999, S. 172-194

[SAG+93] Schilit, Bill N.; Adams, Norman; Gold, Rich; Tso, Michael; Want, Roy: The PARCTAB Mobile Computing System. In: Proceedings of the Fourth Workshop on Workstation Operating Systems (WWOS-IV), Napa, CA, IEEE Computer Society, Okt. 1993, S. 34-39

[Sal00] Salber, Daniel: Context-awareness and multimodality. In: Colloque sur la multimodalité, IMAG, Grenoble, Mai 2000, http://www.irit.fr/M3/CM10ans/Articles/Salber.pdf

[SAW94] Schilit, B. N.; Adams, N. I.; Want, R.: Context-aware computing applications. In: Proceedings of the Workshop on Mobile Computing Systems and Applications, IEEE Computing Society, Santa Cruz, 1994, S. 85-90

[SBG99] Schmidt, Albrecht; Beigl, Michael; Gellersen, Hans-W.: There is more to Context than Location. In: Computers & Graphics Journal, Elsevier, Volume 23, No.6, Dez. 1999, S. 893-902

[SCF+96] Sandhu, R.; Coyne, E.J.; Feinstein, H.L.; Youman, C.E: Role Based Access Control Models. In: IEEE Computer, 29(2), 1996, S. 38-47

[Sch95] Schilit, William Noah: A System Architecture for Context-Aware Mobile Computing. Dissertation, Columbia University, 1995

[Sch99] Schier, Kathrin: Vertrauenswürdige Kommunikation im elektronischen Zahlungsverkehr. Dissertation, Universität Hamburg, Fachbereich Informatik, 1999

[Sch00] Schreck, Jörg: Security and Privacy in User Modeling. Dissertation im Fachbereich Mathematik und Informatik der Universit¨at Gesamthochschule Essen, Sept. 2000

[SDA99] Salber, Daniel; Dey, Anind K.; Abowd, Gregory D.: The Context Toolkit: Aiding the Development of Context-Enabled Applications. In: Proceedings of the 1999 Conference on Human Factors in Computing Systems (CHI '99), Pittsburgh, PA, Mai 1999, S. 434-441

[See01] Seeberg, Cornelia: Modulare Wissensbasen zur Erzeugung adaptiver und kohärenter Lehrdokumente. Dissertation, Fachbereich Informatik, Technische Universität Darmstadt, Juni 2001

[SKA97] Starner, T.; Kirsch, D.; Assefa, S.: The Locust Swarm: An Environmentally-Powered, Networkless Location and Messaging System. In: Proceedings of the First International Symposium on Wearable Computers (ISWC '97), 1997, http://dlib.computer.org/conferen/iswc/8192/pdf/81920169.pdf

[SL99] Sloman, Morris; Lupu, Emil: Policy Specification for Programmable Networks. In: Proceedings of First International Working Conference on Active Networks (IWAN'99), Berlin, Juni 1999

[SM98] Malhotra, A.; Sundaresan, N.: RDF Query Specification. 1998, http://www.w3.org/TandS/QL/QL98/pp/rdfquery.html

[SMR+96] Starner, Thad; Mann, Steve; Rhodes, Bradley; Levine, Jeffrey: Augmented Reality Through Wearable Computing. M.I.T Media Laboratory Perceptual Computing Section Technical Report No. 397, 1996

[SS94] Sandhu, Ravi; Samarati, Pierangela: Access Control: Principles and Practice. In: IEEE Communications, 32(9), 1994, S. 40-48

[SS98] Strassner, John; Schleimer, Stephen: Policy Framework Definition Language. Internet Engineering Task Force, Internet Draft, Nov. 1998, http://www.ietf.org/proceedings/98dec/I-D/draft-ietf-policy-framework-pfdl-00.txt

[SSS00] Smailagic, Asim; Small, Jason; Siewiorek, Daniel P.: Determining User Location For Context Aware Computing Through the Use of a Wireless LAN Infrastructure. Institute for Complex Engineered Systems Carnegie Mellon University, Pittsburgh, PA 15213, 2000, http://www.cs.cmu.edu/~aura/docdir/small00.pdf

[ST93] Spreitzer, Mike; Theimer, Marvin: Providing location information in a ubiquitous computing environment. In: Proceedings of the Fourteenth ACM Symposium on Operating Systems Principles, Asheville, NC, ACM Press, Dez. 1993, S. 270-283

[ST94] Spreitzer, M.; Theimer, M.: Providing location information in a ubiquitous computing environment. In: Proceedings of the 14th International Conference on Distributed Computing Systems, 1994, S. 29-38

[Sta00] Stajano, Frank: The Resurrecting Duckling - what next? In: Security Protocols, 8th International Workshop Proceedings, Lecture Notes in Computer Science, 2000, http://www-lce.eng.cam.ac.uk/~fms27/papers/duckling-what-next.pdf

[Sun97] Inc. Sun Microsystems: Java AWT: Delegation event model. 1997, http://java.sun.com/products/jdk/1.1/docs/guide/awt/designspec/events.html

[Sun98] Inc. Sun Microsystems: Java Distributed Event Specification. 1998, http://www.javasoft.com/products/javaspaces/specs

[SZ97] Simon, S.; Zurko, M.E.: Adage: An Architecture for Distributed Authorization. OSF Research Institute, Cambridge, 1997

[TBM+01] Thompson, Henry S.; Beech, David; Maloney, Murray; Mendelsohn, Noah: XML Schema Part 1: Structures. W3C Recommendation 2 May 2001, http://www.w3.org/1999/05/06-xmlschema-1/, 6 May, 1999

[TCSEC85] Trusted Computer System Evaluation Criteria. Tech. Rep. DoD 5200.28-STD, Department of Defense, 1985

[Tho01] Thompson, Mary R.: Akenti Policy Language. Draft, Juni 2000, http://www-itg.lbl.gov/security/Akenti/Papers/PolicyLanguage.html

[UMI01] User Modeling and User-Adapted Interaction (UMUAI). 2001, http://umuai.informatik.uni-essen.de

[UML97] Rational: UML Summary version 1.1. Sept. 1997, http://www.rational.com/uml

[VCP98] Varadharajan, Vijay; Crall, Chris; Pato, Joe: Authorization in enterprise-wide distributed system: A practical design and application. In: Proceedings of 14th Annual Computer Security Application Conference, IEEE, 1998, S. 178-189

[VSF+00] Volz, St.; Sester, M.; Fritsch, D.; Klinec, D.: NEXUS - Eine Plattform für ortsabhängige, verteilte Geodatennutzung. In: Albertz, J. (Hrsg.): Publikationen der Deutschen Gesellschaft für Photogrammetrie und Fernerkundung, Band 8, 2000, S. 137-150

[War98] Ward, Andrew Martin Robert: Sensor-driven Computing. Dissertation, Corpus Christi College University of Cambridge, Aug. 1998

[Wei91] Weiser, M.: The Computer for the 21st Century. Scientific American, Sept. 1991, S. 66-75

[WH92] Want, R.; Hopper, A.: Active Badges and Personal Interactive Computing Objects. Technical Report 92-2, Olivetti and Oracle Research Limited, Cambridge, UK, 1992

[WHF+92] Want, R.; Hopper, A.; Falcao, V.; Gibbons, J.: The Active Badge Location System. In: ACM Transactions on Information Systems, Vol. 10, No. 1, Jan. 1992, S. 91-102

[Win93] Winklhofer, Andreas: Zeitrepräsentation und merkmalsgesteuerte Suche zur Terminplanung. Dissertation, Bayrisches Forschungszentrum für Wissensbasierte Systeme an der Technische Universität München, Juli 1993

[WSA+95] Want, Roy; Schilit, Bill N. ; Adams, Norman I.; Gold, Rich; Petersen, Karin; Goldberg, David;. Ellis, John R.; Weiser, Mark: An overview of the PARCTAB ubiquitous computing experiment. In: IEEE Personal Communications, 2(6), Dez.1995, S. 28-43

[YP00] Yavatkar, R.; Pendarakis, R.:A Framework for Policy-based Admission Control, IETF RFC 2753, Jan. 2000

[XMLQL98] XML-QL: A Query Language for XML. Aug. 1998, http://www.w3.org/TR/NOTE-xml-ql/

Homepages

[Akenti] Akenti - Distributed Access Control. Distributed Security Research Group, Distributed Systems Department, Lawrence Berkeley National Laboratory, http://www-itg.lbl.gov/Akenti

[DC] Dublin Core Initiative. http://dublincore.org

[Cooltown] Hewlett-Packard Company: Cooltown. http://www.cooltown.hp.com/

[DSZ] Datenschutzzentrum: Gesetze und Verordnungen. http://www.datenschutzzentrum.de/ldsh/gesetze.htm

[Findentity] Thax Software GmbH: Findentity. 2002, http://www.thax.de

[IBMAW] IBM AlphaWorks. http://www.alphaworks.ibm.com

[Locust] LOCUST: An Experiment in Private Localization. MIT Media Lab, http://www.media.mit.edu/wearables/lizzy/locust/, Jan. 2002

[LORE] LORE Project. Stanford University, Database Group, http://www-db.stanford.edu/lore/

[MediaCup] MediaCup. Telecooperation Office, Institut für Telematik, Fakultät für Informatik, Universität Karlsruhe (TH), http://mediacup.teco.edu

[MUSE] MUSE Projekt. http://medialab.cs.ucla.edu/muse

[Nexus] NEXUS Projekt. University of Stuttgart, http://www.nexus.uni-stuttgart.de

[RCP] RaumComputer AG: RaumComputer. http://www.raumcomputer.com

[RDF] Resource Description Framework (RDF). W3C, http://www.w3.org/RDF

[RDFS] ICS-FORTH: RDF Suite. http://139.91.183.30:9090/RDF/

[Sesame] Aidministrator Nederland bv: Sesame. http://sesame.aidministrator.nl

[Siena] Siena. Software Engineering Research Laboratory, University of Colorado, http://www.cs.colorado.edu/users/carzanig/siena

[SLP] Sm@rtLibrary-Projekt. GMD-SIT, http://sit.gmd.de/COR/smartlibrary

[Tamino] Software AG: Tamino. 2002, http://www.softwareag.com

[TTS] TeleTravel System. http://www.teletravelsystem.de

[UMI] User Modeling Inc. http://bistrica.usask.ca/UM, im Aufbau: http://www.um.org

[XQEngine] XML Query Engine. http://www.fatdog.com/

[XQLE] XQL Engine. http://xml.darmstadt.gmd.de/xql/

Anhang

A. XML-Schema der CDACL

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="CDACPolicy">
        <xsd:annotation>
            <xsd:documentation>Context-Dependent Access Control Policy</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="PolicyLabel" type="xsd:string" minOccurs="0"/>
                <xsd:element name="Subjects" type="SubjectsType" minOccurs="0"/>
                <xsd:element name="Actions" type="ActionsType" minOccurs="0"/>
                <xsd:element name="Objects" type="ObjectsType" minOccurs="0"/>
                <xsd:element name="Events" type="EventsType" minOccurs="0"/>
                <xsd:element name="Sets" type="SetsType" minOccurs="0"/>
                <xsd:element name="ACRuleBase" type="ACRuleBaseType" minOccurs="0">
                    <xsd:annotation>
                        <xsd:documentation>CDAC Rules</xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="version" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="SetsType">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element name="ActionSet" type="ActionSetType"/>
            <xsd:element name="ObjectSet" type="ObjectSetType"/>
            <xsd:element name="SubjectSet" type="SubjectSetType"/>
            <xsd:element name="EventSet" type="EventSetType"/>
        </xsd:choice>
    </xsd:complexType>

    <xsd:complexType name="ACRuleBaseType">
        <xsd:sequence maxOccurs="unbounded">
            <xsd:element name="ACRule" type="ACRuleType"/>
        </xsd:sequence>
```

```
</xsd:complexType>

<xsd:complexType name="ACRuleType">
    <xsd:sequence>
        <xsd:element name="RScope" type="RScopeType">
            <xsd:annotation>
                <xsd:documentation>Rule Scope</xsd:documentation>
            </xsd:annotation>
        </xsd:element>
        <xsd:element name="RContext" type="RContextType">
            <xsd:annotation>
                <xsd:documentation>Rule Condition</xsd:documentation>
            </xsd:annotation>
        </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="Priority" type="xsd:integer" use="optional"/>
    <xsd:attribute name="ID" type="xsd:string" use="optional"/>
    <xsd:attribute name="ref" type="xsd:IDREF" use="optional"/>
</xsd:complexType>

<xsd:complexType name="RScopeType">
    <xsd:annotation>
        <xsd:documentation>Rule Scope</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="Subjects" type="SubjectsType" maxOccurs="unbounded"/>
        <xsd:element name="Actions" type="ActionsType" maxOccurs="unbounded"/>
        <xsd:element name="Objects" type="ObjectsType" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="SubjectsType">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="Subject" type="SubjectType"/>
        <xsd:element name="SubjectSet" type="SubjectSetType"/>
    </xsd:choice>
</xsd:complexType>
<xsd:complexType name="ActionsType">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="Action" type="ActionType"/>
        <xsd:element name="ActionSet" type="ActionSetType"/>
    </xsd:choice>
</xsd:complexType>
<xsd:complexType name="ObjectsType">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="Object" type="ObjectType"/>
        <xsd:element name="ObjectSet" type="ObjectSetType"/>
    </xsd:choice>
</xsd:complexType>
<xsd:complexType name="EventsType">
    <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="Event" type="EventType"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="SubjectType">
    <xsd:choice minOccurs="0">
        <xsd:element name="uname" type="xsd:string"/>
```

```
            <xsd:element name="href" type="xsd:string"/>
            <xsd:element name="uid" type="xsd:string"/>
        </xsd:choice>
        <xsd:attribute name="ID" type="xsd:ID" use="optional"/>
        <xsd:attribute name="ref" type="xsd:IDREF" use="optional"/>
    </xsd:complexType>
    <xsd:complexType name="SubjectSetType">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element name="Subject" type="SubjectType"/>
            <xsd:element name="SubSet" type="SubjectSetType"/>
            <xsd:choice minOccurs="0">
                <xsd:element name="gname" type="xsd:string"/>
                <xsd:element name="href" type="xsd:string"/>
                <xsd:element name="gid" type="xsd:string"/>
            </xsd:choice>
        </xsd:choice>
        <xsd:attribute name="ID" type="xsd:ID" use="optional"/>
        <xsd:attribute name="ref" type="xsd:IDREF" use="optional"/>
    </xsd:complexType>
    <xsd:complexType name="SubSetType">
        <xsd:attribute name="Value" type="xsd:IDREF" use="required"/>
    </xsd:complexType>
    <xsd:complexType name="ActionType">
        <xsd:sequence minOccurs="0">
            <xsd:element name="Name" type="xsd:NMTOKEN"/>
            <xsd:element name="Args" type="xsd:NMTOKENS" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="ID" type="xsd:ID" use="optional"/>
        <xsd:attribute name="ref" type="xsd:IDREF" use="optional"/>
    </xsd:complexType>
    <xsd:complexType name="ActionSetType">
        <xsd:choice minOccurs="0">
            <xsd:element name="Action" type="ActionType"/>
            <xsd:element name="ActionSet" type="ActionSetType"/>
        </xsd:choice>
        <xsd:attribute name="ID" type="xsd:ID" use="optional"/>
        <xsd:attribute name="ref" type="xsd:IDREF" use="optional"/>
    </xsd:complexType>
    <xsd:complexType name="ObjectType">
        <xsd:choice minOccurs="0">
            <xsd:element name="href" type="xsd:string"/>
        </xsd:choice>
        <xsd:attribute name="ID" type="xsd:ID" use="optional"/>
        <xsd:attribute name="ref" type="xsd:IDREF" use="optional"/>
    </xsd:complexType>
    <xsd:complexType name="ObjectSetType">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element name="Object" type="ObjectType"/>
            <xsd:element name="Include" type="xsd:string"/>
            <xsd:element name="Exclude" type="xsd:string"/>
            <xsd:element name="SubSet" type="ObjectSetType"/>
        </xsd:choice>
        <xsd:attribute name="ID" type="xsd:ID" use="optional"/>
        <xsd:attribute name="ref" type="xsd:IDREF" use="optional"/>
    </xsd:complexType>

    <xsd:complexType name="RContextType">
```

```
    <xsd:choice>
        <xsd:element name="ContextPredicate" type="ContextPredicateType"/>
        <xsd:element name="EventPredicate" type="EventPredicateType"/>
        <xsd:element name="TimePredicate" type="TimePredicateType"/>
        <xsd:element name="ScopePredicate" type="ScopePredicateType"/>
        <xsd:element name="AND" type="ANDType"/>
        <xsd:element name="OR" type="ORType"/>
        <xsd:element name="NOT" type="NOTType"/>
    </xsd:choice>
</xsd:complexType>

<xsd:complexType name="ANDType">
    <xsd:choice minOccurs="2" maxOccurs="unbounded">
        <xsd:element name="ContextPredicate" type="ContextPredicateType"/>
        <xsd:element name="TimePredicate" type="TimePredicateType"/>
        <xsd:element name="ScopePredicate" type="ScopePredicateType"/>
        <xsd:element name="EventPredicate" type="EventPredicateType"/>
        <xsd:element name="AND" type="ANDType"/>
        <xsd:element name="OR" type="ORType"/>
        <xsd:element name="NOT" type="NOTType"/>
    </xsd:choice>
</xsd:complexType>

<xsd:complexType name="ORType">
    <xsd:choice maxOccurs="unbounded">
        <xsd:element name="ContextPredicate" type="ContextPredicateType"/>
        <xsd:element name="TimePredicate" type="TimePredicateType"/>
        <xsd:element name="ScopePredicate" type="ScopePredicateType"/>
        <xsd:element name="EventPredicate" type="EventPredicateType"/>
        <xsd:element name="AND" type="ANDType"/>
        <xsd:element name="OR" type="ORType"/>
        <xsd:element name="NOT" type="NOTType"/>
    </xsd:choice>
</xsd:complexType>

<xsd:complexType name="NOTType">
    <xsd:choice>
        <xsd:element name="ContextPredicate" type="ContextPredicateType"/>
        <xsd:element name="TimePredicate" type="TimePredicateType"/>
        <xsd:element name="ScopePredicate" type="ScopePredicateType"/>
        <xsd:element name="EventPredicate" type="EventPredicateType"/>
        <xsd:element name="AND" type="ANDType"/>
        <xsd:element name="OR" type="ORType"/>
        <xsd:element name="NOT" type="NOTType"/>
    </xsd:choice>
</xsd:complexType>

<xsd:complexType name="ContextPredicateType">
    <xsd:choice>
        <xsd:element name="Equal" type="EqualType"/>
        <xsd:element name="Greater" type="GreaterType"/>
        <xsd:element name="Less" type="LessType"/>
        <xsd:element name="SubsetOf" type="SubsetOfType"/>
        <xsd:element name="ElementOf" type="ElementOfType"/>
        <xsd:element name="Rel" type="RelType"/>
    </xsd:choice>
</xsd:complexType>
```

```
<xsd:complexType name="TimePredicateType">
    <xsd:choice>
        <xsd:element name="During" type="DuringType"/>
        <xsd:element name="At" type="AtType"/>
        <xsd:element name="Before" type="BeforeType"/>
        <xsd:element name="After" type="AfterType"/>
    </xsd:choice>
</xsd:complexType>

<xsd:complexType name="ScopePredicateType">
    <xsd:choice>
        <xsd:element name="Equal" type="EqualType"/>
        <xsd:element name="ElementOf" type="ElementOfType"/>
        <xsd:element name="SubsetOf" type="SubsetOfType"/>
    </xsd:choice>
</xsd:complexType>

<xsd:complexType name="EventPredicateType">
    <xsd:choice>
        <xsd:element name="ProtocolDep" type="ProtocolDepType"/>
        <xsd:element name="CausalDep" type="DepType"/>
        <xsd:element name="TemporalDep" type="DepType"/>
    </xsd:choice>
</xsd:complexType>

<xsd:complexType name="EqualType">
    <xsd:choice>
        <xsd:sequence>
            <xsd:element ref="Var"/>
            <xsd:element name="Const" type="ConstType"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="Const" type="ConstType"/>
            <xsd:element ref="Var"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element ref="Var"/>
            <xsd:element ref="Var"/>
        </xsd:sequence>
    </xsd:choice>
</xsd:complexType>

<xsd:complexType name="GreaterType">
    <xsd:choice>
        <xsd:sequence>
            <xsd:element ref="Var"/>
            <xsd:element name="Const" type="ConstType"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="Const" type="ConstType"/>
            <xsd:element ref="Var"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element ref="Var"/>
            <xsd:element ref="Var"/>
        </xsd:sequence>
```

```
    </xsd:choice>
</xsd:complexType>

<xsd:complexType name="LessType">
    <xsd:choice>
        <xsd:sequence>
            <xsd:element ref="Var"/>
            <xsd:element name="Const" type="ConstType"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="Const" type="ConstType"/>
            <xsd:element ref="Var"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element ref="Var"/>
            <xsd:element ref="Var"/>
        </xsd:sequence>
    </xsd:choice>
</xsd:complexType>

<xsd:complexType name="SubsetOfType">
    <xsd:choice>
        <xsd:sequence>
            <xsd:element ref="Var"/>
            <xsd:element name="Const" type="ConstType"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="Const" type="ConstType"/>
            <xsd:element ref="Var"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element ref="Var"/>
            <xsd:element ref="Var"/>
        </xsd:sequence>
    </xsd:choice>
</xsd:complexType>
<xsd:complexType name="ElementOfType">
    <xsd:choice>
        <xsd:sequence>
            <xsd:element ref="Var"/>
            <xsd:element name="Const" type="ConstType"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="Const" type="ConstType"/>
            <xsd:element ref="Var"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element ref="Var"/>
            <xsd:element ref="Var"/>
        </xsd:sequence>
    </xsd:choice>
</xsd:complexType>

<xsd:complexType name="RelType">
    <xsd:sequence>
        <xsd:element ref="RelOp"/>
        <xsd:choice maxOccurs="unbounded">
```

```
            <xsd:element ref="Var"/>
            <xsd:element name="Const" type="ConstType"/>
        </xsd:choice>
    </xsd:sequence>
</xsd:complexType>
<xsd:element name="RelOp" type="xsd:string"/>

<xsd:complexType name="DuringType">
    <xsd:choice>
        <xsd:sequence>
            <xsd:element ref="tStart"/>
            <xsd:choice>
                <xsd:element ref="tEnd"/>
                <xsd:element ref="Var"/>
                <xsd:element name="Const" type="ConstType"/>
            </xsd:choice>
        </xsd:sequence>
        <xsd:element ref="Var"/>
    </xsd:choice>
</xsd:complexType>
<xsd:complexType name="AtType">
    <xsd:choice>
        <xsd:sequence>
            <xsd:element ref="tStart"/>
            <xsd:element name="Const" type="ConstType"/>
        </xsd:sequence>
        <xsd:element ref="Var"/>
    </xsd:choice>
</xsd:complexType>
<xsd:complexType name="BeforeType">
    <xsd:choice>
        <xsd:sequence>
            <xsd:element ref="tStart"/>
            <xsd:element name="Const" type="ConstType"/>
        </xsd:sequence>
        <xsd:element ref="Var"/>
    </xsd:choice>
</xsd:complexType>

<xsd:complexType name="AfterType">
    <xsd:choice>
        <xsd:sequence>
            <xsd:element ref="tStart"/>
            <xsd:element name="Const" type="ConstType"/>
        </xsd:sequence>
        <xsd:element ref="Var"/>
    </xsd:choice>
</xsd:complexType>

<xsd:complexType name="DepType">
    <xsd:sequence>
        <xsd:element name="Event" type="EventType"/>
        <xsd:element name="TimeCond" type="TimeCondType" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ProtocolDepType">
```

```
    <xsd:sequence>
        <xsd:element name="Initial" type="EventType" minOccurs="0"/>
        <xsd:element name="Protocol" type="ProtocolType" maxOccurs="unbounded"/>
        <xsd:element name="TimeCond" type="TimeCondType" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ProtocolType">
    <xsd:choice maxOccurs="unbounded">
        <xsd:element name="Event" type="EventType" minOccurs="0"/>
        <xsd:element name="ESeq" type="ProtocolType" minOccurs="0"/>
        <xsd:element name="EAlt" type="ProtocolType" minOccurs="0"/>
        <xsd:element name="EIt" type="EventType" minOccurs="0"/>
    </xsd:choice>
</xsd:complexType>

<xsd:complexType name="ESeqType">
    <xsd:sequence>
        <xsd:element name="Event" type="EventType" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="TimeCondType">
    <xsd:choice>
        <xsd:element name="During" type="DuringType"/>
        <xsd:element name="At" type="AtType"/>
        <xsd:element name="Before" type="BeforeType"/>
        <xsd:element name="After" type="AfterType"/>
    </xsd:choice>
</xsd:complexType>

<xsd:element name="tStart" type="xsd:string"/>
<xsd:element name="tEnd" type="xsd:string"/>
<xsd:simpleType name="dateTyp">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="[0-3][0-9]-[0-1][0-9]-[0-9][0-9]"/>
    </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="ConstType">
    <xsd:attribute name="Value" type="xsd:NMTOKEN" use="required"/>
    <xsd:attribute name="Unit" type="xsd:NMTOKEN" use="optional"/>
</xsd:complexType>
<xsd:element name="Var">
    <xsd:complexType>
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="Unit" type="xsd:string" use="optional"/>
                <xsd:attribute name="Certainty" type="xsd:float" use="optional"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="EventSetType">
    <xsd:sequence maxOccurs="unbounded">
        <xsd:element name="Event" type="EventType"/>
    </xsd:sequence>
```

```
    </xsd:complexType>
    <xsd:complexType name="EventType">
        <xsd:sequence maxOccurs="unbounded">
            <xsd:element name="ESubject" type="xsd:string"/>
            <xsd:element name="EObject" type="xsd:string"/>
            <xsd:element name="EAction" type="xsd:string"/>
        </xsd:sequence>
        <xsd:attribute name="ID" type="xsd:ID" use="optional"/>
        <xsd:attribute name="ref" type="xsd:IDREF" use="optional"/>
    </xsd:complexType>
</xsd:schema>
```

B. Code

Packages und Klassenhierarchie des Policy Decision Point (PDP) sowie Code der Hauptfunktion des Regel-Interpreters:

Packages:

- cdac.pdp.cops
- cdac.pdp.cops.configuration
- cdac.pdp.cops.md5
- cdac.pdp.cops.message
- cdac.pdp.cops.protocol
- cdac.pdp.cops.thread
- cdac.pdp.cops.util
- cdac.pdp.dtmanag
- cdac.pdp.dtmanag.jdom
- cdac.pdp.dtmanag.ldap
- cdac.pdp.policy
- cdac.pdp.policy.lib.math
- cdac.pdp.policy.lib.text
- cdac.pdp.policy.rules
- cdac.pdp.policy.rules.context
- cdac.pdp.policy.rules.context.location
- cdac.pdp.policy.rules.context.rql
- cdac.pdp.policy.rules.date
- cdac.pdp.policy.rules.scope
- cdac.pdp.policy.util

Klassenhierarchie:

Class

|-----ActivePEPHandle
|-----Attribute
|-----Attributes
|-------- AuthenticationAttribute
|-------- PolicyErrorObject
|-----AuthenticationFailureException
|-----AuthenticationRequiredException
|-----BuilderErrorHandler
|-----ByteArrayInputOutput
|-----CDACLDriver
|-----CDACLException
|---------IllegalAddException
|---------IllegalDataException
|---------IllegalNameException
|---------IllegalTargetException
|---------JDOMException
|----------DataConversionException
|-----CDATA
|-----ClientHandle
|-----ClientServerConstants
|-----ClientSettings
|-----Command
|-----Comment
|-----ConditionIterator
|-------- AbstractAutoFactIterator
|---------- AutoFactIterator
|---------- SingleAutoFactIterator
|-------- AbstractConditionIterator
|---------- CachedConditionIterator
|---------- MergedConditionIterator
|---------- MultipleConditionSetIterator
|---------- SingleConditionSetIterator
|-------- SingleConditionIterator
|-----ConditionSetChangeEvent
|-----ConditionSetChangeListener
|-------- RuleBase
|-----Configuration
|-------- InferenceEngineConfiguration
|-------- RuleBaseConfiguration
|-----Configuration
|-------- PDPServerConfig
|-----ConfigurationPDP
|-----ConnectionInfo
|-----ConnectionParameters
|-----Conversion
|-----COPS_Util
|-----COPSConstants
|-----COPSData
|-----COPSHeader
|-----COPSInit
|-----COPSOperations
|-----Constructor
|-------- Function
|---------- DateFunction
|---------- DoubleFunction
|---------- IntFunction
|---------- SetFunction
|---------- StringFunction
|-------- CConstructor
|---------- CFunction
|---------- CPredicate
|-------- Predicate
|---------- ContextPredicate
|---------- DatePredicate
|---------- ScopePredicate
|-----Conversion
|-----COPS_Util
|-----COPSConstants
|-----COPSData
|-----COPSHeader
|-----COPSInit
|-----COPSOperations
|-----DataManager
|-----DataMessageSingleton
|-----DataSource
|-------- StaticDataSource
|-----DateArithmetic
|-----DecisionHandler
|-----DecisionMaker
|-----DecisionMessage
|-----DeleteRequestState
|-----Derivation
|-----DocType
|-----Document
|-----DOMAdapter
|---------AbstractDOMAdapter
|----------CrimsonDOMAdapter
|----------JAXPDOMAdapter
|----------XercesDOMAdapter
|----------XML4JDOMAdapter
|-----DOMBuilder
|-----DOMOutputter
|-----Element
|-----EntityRef
|-----InferenceEngine
|-----JDOMFactory
|---------DefaultJDOMFactory
|-----JDOMResult
|-----JDOMSource
|-----LDAP
|-------- Policy
|-------- PolicyCondition
|-------- PolicyValidityPeriod
|-------- Profile

|--------- UserIDCondition
|-----LDAPController
|-----LDAPEngine
|-----LDAPServerConfig
|-----LogCategories
|-----LogicFactorySupport
|-----LoopCheckingAlgorithm
|-----MD5
|-----MD5Authentication
|-----MD5InputStream
|-----MD5OutputStream
|-----MD5State
|-----Message
|-----MultipleCollectionIterator
|-----Namespace
|-----NamespaceStack
|---------XMLOutputter::NamespaceStack
|-----ObjectIDs
|-----ObjectListener
|-----PartialList
|-----PDPData
|-----PDPException
|--------- TimedOutException
|--------- UnknownCOPSObjectException
|--------- UnsupportedClientTypeException
|-----PDUElement
|--------- COPSobject
|---------- AcctTimerObject
|---------- ClientSIObject
|---------- ContextObject
|---------- DecisionObject
|---------- ErrorObject
|---------- HandleObject
|---------- IntegrityObject
|---------- LastPDPAddrObject
|---------- LDPDecisionObject
|---------- Out_InterfaceObject
|---------- PDPRedirAddr
|---------- PDPRedirAddrObject
|---------- PEPIDObject
|---------- TimerObject
|--------- PDUHeader
|-----PolicyData
|-----PolicyDataAppli
|-----PolicyElement
|-----PolicyOptions
|-----PolicySelection
|-----ProcessingInstruction
|-----ReqTimeOut
|-----RequestHandler
|-----RequestMessage
|-----RequestMessageFrame
|-----RequestTimeOut
|-----RuleOwner
|--------- ConditionSet
|---------- ConditionSet
|---------- Condition
|--------- RuleBase
|-----ServerSideConnector
|-----SimpleConstructor
|--------- SimpleFunction
|--------- SimplePredicate
|-----SynchronizeStateRequestMessage
|-----Term
|--------- ComplexTerm
|--------- ConstantTerm
|--------- VariableTerm
|-----TermContainer
|--------- ComplexTerm
|-----Text
|-----Time
|-----TimerEvent
|-----TimerListener
|-----Timer
|--------- COPS_PDP
|-----UnificationAlgorithm
|-----Utils
|-----Verifier
|-----WorkerThread
|-----XMLDriver
|-----XMLOutputter

Hauptfunktion des Regelinterpreter:

```
/**
* Determine the decision about the access request
* @return dec        "allowed" or "denied"
* @param subject_p   subject identifier
* @param action_p    type of action/name of action
* @param object_p    target identifier/URI or OID
* @param ip_p        IP-address of client
*/
```

```
public byte getDecision(String subject_p, String action_p, String object_p, String ip_p) {

    Rule[]  actRules;
    Rule[]  resRules;
    byte dec=0;  //Decision
    int  prio;   //priority of rule
    int  j=0;

    // determine conflict set for actual access triple of
    // subject, action and target object
    actRules=getActiveRules(subject_p, action_p, object_p);
    // sort list by decreasing priority
    actRules=sortDecPrio(actRules);
    // evaluate rules with highest priority
    prio=actRules[0].eval(subject_p, action_p, object_p, ip_p);
    for (int i=1;i<actRules.length;i++) {
        if (actRules[i].eval(subject_p, action_p, object_p, ip_p) == prio) {
            resRules[j++]=actRules[i];
        }
        else break;
    }

    // if there is more than one rule with the same priority
    // then global security strategy decides
    dec=resRules[0].getDecision();
    for (int i=1; i<resRules.length; i++) {
        byte dect=resRules[i].getDecision();
        if (dect!=dec) dec=globSecStrat(dec, dect);
    }
    return dec;
}
```

C. Eigene Veröffentlichungen

Schumacher, Markus; Roedig, Utz; Moschgath, Marie-Luise: Hacker Contest: Sicherheitsprobleme, Lösungen, Beispiele. Springer, Heidelberg, 2002

Moschgath, Marie-Luise; Hähner, Jörg; Reinema, Rolf: Sm@rtLibrary - An Infrastructure for Ubiquitous Technologies and Applications. In: Proceedings of 21st International Conference on Distributed Computing Systems Workshops (ICDCS 2001), International Workshop on Smart Appliances and Wearable Computing (IWSAWC), Mesa, Arizona, IEEE Computer Society, ISBN 0-7695-1080-9, 2001, S. 208-213

Moschgath, Marie-Luise; Reinema, Rolf; Hoffmann, Mario: A Security Infrastructure for the Virtual Project Office. In: Proceedings of the IEEE International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2000), FESB-Split, Okt. 2000, S. 563-572

Reinema, Rolf; Moschgath, Marie-Luise; Hoffmann, Mario: Eine Sicherheitsinfrastruktur für das Virtuelle Projektbüro. In: Sicherheit in Netzen und Medienströmen, M. Schumacher; R. Steinmetz (Hrsg.). Springer-Verlag Berlin Heidelberg, Sept. 2000, S. 73-82

Schumacher, Markus; Moschgath, Marie-Luise; Roedig, Utz: Angewandte Informationssicherheit: Ein Hacker-Praktikum an Universitäten. In: Informatik-Spektrum. 23 (2000) 3, S. 202-211

Reinema, Rolf; Moschgath, Marie-Luise; Bahr, Knut; Hovestadt, Ludger: roomComputers - Bridging Spaces. In: Proceedings of IEEE Workshop on Future Trends of Distributed Computing Systems FTDCS 99, Cape Town, South Africa, Dez. 1999, S. 75-81

Mattern, Friedemann; Moschgath, Marie-Luise; Fünfrocken, Stefan: Die JavaCard als Programmier- und Ausführungsplattform für verteilte Anwendungen. In: Proceedings of JIT'99, Springer-Verlag, Sept. 1999, S. 100-109

Mattern, Friedemann; Moschgath, Marie-Luise; Fünfrocken, Stefan: Die JavaCard als Programmier- und Anwendungsplattform für verteilte Anwendungen. In: Online 99 Congressband VI, Symposium VI-2, Web Programmierung: Das Internet als verteilte Programmierplattform, C622, Friedrich Vogt (Hrsg.), Feb. 1999

Vogler, Hartmut; Spriestersbach, Alex; Moschgath, Marie-Luise: Protecting Competitive Negotiation of Mobile Agents. In: Proceedings of IEEE Workshop on Future Trends of Distributed Computing Systems FTDCS 99, Cape Town, South Africa, Dec. 1999, S.145-150

Vogler, Hartmut; Moschgath, Marie-Luise; Kunkelmann, Thomas; Grünewald, Jürgen: The Transaction Internet Protocol in Practice: Reliability for WWW Applications. In Proc. of the Internet Workshop'99 (IWS'99), Osaka, Japan, IEEE Computer Society, 1999

Vogler, Hartmut; Moschgath, Marie-Luise; Kunkelmann, Thomas: Enhancing Mobile Agents with Electronic Commerce Capabilities. In: Proc. 2nd International Workshop Cooperative Information Agents (CIA-98), Springer-Verlag LNCS/LNAI 1435, Juli 1998

Kunkelmann, Thomas; Vogler, Hartmut; Moschgath, Marie-Luise; Wolf, Lars: Scalable Security Mechanisms in Transport Systems for Enhanced Multimedia Services. In: Proceedings of 3rd European Conference on Multimedia Applications, Services and Techniques (ECMAST'98), Berlin, Mai 1998

Vogler, Hartmut; Kunkelmann, Thomas; Moschgath, Marie-Luise: An Approach for Mobile Agent Security and Fault Tolerance using Distributed Transactions. In: Proceedings of 1997 Int'l Conference on Parallel and Distributed Systems (ICPADS'97), Seoul, Korea, IEEE Computer Society, Dez. 1997, S.268-274

Vogler, Hartmut; Kunkelmann, Thomas; Moschgath, Marie-Luise: Distributed Transaction Processing as a Reliability Concept for Mobile Agents. In: Proceedings of 6th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'97), Tunis, Tunisia, IEEE Computer Society, Okt. 1997, S.59-64

Lebenslauf

Name:	Marie-Luise Moschgath
Geburtsdatum:	22. April 1966

Schule und Studiumg

1972-1976	Grundschule Dettingen
1976-1985	Voehlin-Gymnasium Memmingen, Abschluss: Abitur
1985-1993	Studium an der Friedrich-Alexander-Universität Erlangen-Nürnberg, Fachrichtung Informatik Abschluss: Diplom-Informatikerin

Berufliche Tätigkeit

März 1987 bis April 1987, Aug. 1991 bis Okt. 1991	Werkstudentin Siemens Forschungszentrum Erlangen
Nov. 1991 bis Apr. 1992, Juni 1992 bis Okt. 1992	Studentische Hilfskraft am Lehrstuhl für Fertigungsautomatisierung und Produktionssystematik, Erlangen
Apr. 1993 bis Juli 1993	Freie Mitarbeiterin der Softwarefirma FrankenData, Erlangen
Aug. 1993 bis Sept. 1995	Softwareentwicklerin bei Heitec Informationssysteme, Erlangen
Sept. 1995 bis Sept. 1999	Wissenschaftliche Mitarbeiterin am Informationstechnologie Transfer Office (ITO), TU Darmstadt
Okt. 1999 bis Sept. 2001	Wissenschaftliche Mitarbeiterin der Distributed Systems Group, ETH Zürich